Learning Motion Dynamics to Catch a Moving Object

Seungsu Kim, Elena Gribovskaya and Aude Billard

***Abstract*— In this paper, we consider a novel approach to control the *timing of motions* when these are encoded with autonomous dynamical systems (DS). Accurate timing of motion is crucial if a robot must synchronize its movement with that of a fast moving object. In previous work of ours [1], we developed an approach to encode robot motion into DS. Such a *time-independent* encoding is advantageous in that it offers robustness against violent perturbation by adapting on the fly the trajectory while ensuring high accuracy at the target. We propose here an extension of the system that allows to control the timing of the motion while still benefitting from all the robustness properties deriving from the time-independent encoding of the DS. We validate the approach in experiments where the iCub robot learns from human demonstrations to catch a ball on the fly.**

I. Introduction

Planning human-like robot trajectories for catching rapidly moving targets is a challenging task. It requires to consider two closely related problems: (1) predicting accurately the trajectories of the fast moving object; (2) and fast planning of precise trajectories for the robot's end-effector. Estimation of the dynamics of the moving object relies on accurate sensing which cannot always be ensured in robotics. This may lead to a frequent re-estimation of the target's location as both robot and object are moving. To compensate for such inaccurate sensing, one needs to be able to constantly and rapidly re-estimate the trajectory of the robot's arm. In this paper, we address both issues: estimating the motion of the moving object and replanning of the hand's trajectories so as to adapt to sudden temporal and spatial perturbation of the target.

A body of work has been devoted to autonomous control of fast movements such as catching [2] [3] [4] [5] [6] [7] and hitting flying objects [8] [9], or juggling [10] [11] [12]. Next, we briefly review these works according to (1) how they predict trajectories of moving objects and (2) how they generate the robot's motions [13]. To catch a moving object properly, prediction of the trajectory of moving objects is required. Hong *et al* [2] and Riley *et al* [5] model trajectories of the flying ball as a parabola, and subsequently recursively estimate the ball's trajectory through least squares optimization. Frese *et al* [4] and Park *et al* [7] also assume a parabolic form for the ball trajectories and predict the latter with Extended Kalman Filters [14]. Hong *et al* [2] generate trajectories of specific light-weighted objects using a generic aerodynamical model. To estimate the coefficient of this model, they use a wavelet network [15].

LASA Laboratory, EPFL, CH 1015 Lausanne, Switzerland {seungsu.kim; elena.gribovskaya; aude.billard} @epfl.ch

Such approaches can accurately estimate the trajectories. But they rely on determining in advance a model of the motion of the object. Here we estimate the dynamics of the moving object using our generic estimator of non-linear dynamical systems and based on the set of demonstrations.

To generate trajectories for catching moving objects, several works use polynomials [2] [3] [6] [8] to satisfy some boundary values on the trajectory. Zhang *et al* [3] used 5th order polynomial to match catching position, velocity and acceleration. Hong and Slotine [2] use 3^{rd} order polynomials to match the position/velocity of the end-effector and the ball. To accurately reproduce the catching motion, they decelerate the end-effector velocity along the initial path of the object after catching an object. Namiki *et al* [6] also used polynomial equation. The coefficients of the polynomial are found by resolving an optimization problem, where the sum of the torques and angular velocities are minimized so as to satisfy constraints on the initial and final position, velocity and acceleration of the end-effector. Senoo *et al* [8] developed a batting robot based on their high-speed vision system. They split the control of the robot's joint so as to control separately for high-speed swinging while allowing the remaining degrees of freedom are used for fast adaptation to perturbation. Again a 5^{th} order polynomial was used for trajectory generation in joint space.

Another approach to generate a trajectory is imitating human behaviors. Schaal and Atkeson [10] [11] implemented robot juggling tasks based on learning human behavior. They used locally weighted regression to represent a learned model of the task. Riely *et al* [5] use the point-to-point movement representation primitives by programmable pattern generators (PPGs) [16] which is based on human movements, to catch vertically falling objects. It can modify the trajectory on-line for a new target.

Along the same line of thought, the Dynamic Motor Primitive (DMP) offers a dynamical systems based representation of motion [17]. Recent work extended the original DMP formulation to allow a non-zero velocity at the target and was successfully used for hitting a moving target [9]. However, DMP as well as all the other approaches mentioned above are time-dependent. This makes these methods very sensitive to temporal perturbations. I.e., important changes in the duration of the movement that arise when the distance from the end effector to the target is reduced or extended, cannot be handled easily. A heuristic must be used to rescale in time the clock of the system. This problem has been largely overlooked in the literature sofar. In [13], we proposed a means to learn time-independent dynamical systems (autonomous dynamical systems) so as to become robust

to temporal perturbations. Here, we extend this work and address the problem of controlling the duration of the motion when encoded in an autonomous DS. Note that neither time-independent nor time-dependent dynamical systems can be explicitly controlled for the duration of the motion. Here we show how our controller can be used to control both time-dependent DS such as DMP and autonomous dynamical systems.

Though all the methods discussed above were successfully applied to the object catching, hitting or juggling, they are explicitly time-dependent and hence any temporal perturbation after onset of the motion would not be properly handled.

Specifically in this paper, we investigate the problem of discovering and imposing temporal constraints on motions encoded with non-linear dynamical systems [1]. We also demonstrate how our learning framework can be extended to learn motion of external objects with which the robot should synchronize. The accurate motion timing is highly important if a robot has to synchronize with external moving objects. However, the problem of imposing timing constraints on arbitrary non-linear DS has received so far little attention, since this encoding has been mainly applied to spatially constrained manipulation tasks rather than to tasks requiring explicit temporal coordination. Here, we investigate control of timing of a learned dynamical system so as to speed up or slow down the robot's motion and hence adhere to precise temporal constraints.

In previous work of ours, we addressed different aspects of encoding motions with dynamical systems, specifically: effective and smooth adaptation in the case of spatio-temporal perturbations [13], learning of asymptotically stable estimates [18], learning of position and orientation control for motion generation [1]. The present paper continues research in this direction and presents new results on *learning* motions of both the robot's *end-effector* and *external objects*. This highlights the ability of the system for continuous spatio-temporal adaptation and *synchronization*.

II. Dynamical systems with timing constraints

A. Autonomous dynamical system (DS)

To teach a new skill to a robot, a human demonstrates it several times. The demonstrated trajectories together with velocities are encoded with Gaussian Mixture Models (GMM) through Expectation-Maximization (EM) algorithm. The learned movements are represented by a first-order multivariate autonomous dynamical system [1] :

$$\dot{\xi} = \hat{f}\left(\xi\right) = \sum_{k=1}^{K} h_k\left(\xi\right)\left(\mu_k^{\dot{\xi}} + \Sigma_k^{\dot{\xi}\xi}\left(\Sigma_k^{\xi}\right)^{-1}\left(\xi - \mu_k^{\xi}\right)\right) \tag{1}$$

where $\xi, \dot{\xi}$ are the position and the velocity of the robot's end-effector respectively; K is the number of Gaussian components; μ_k and Σ_k are the mean and the covariance of a k^{th} Gaussian component. $h_k\left(\xi\right)$ gives a measure of the influence of the k^{th} Gaussian in generating the data point ξ; see [1] for details. Controlling point to point robot motion with an autonomous dynamical system given in [1] requires to ensure that the so-generated motion is asymptotically stable at the target of the motion. To verify the stability of our motion generator, we use the approach suggested in [18].

B. Timing controller

Controlling for the duration of the motion generated by the dynamical system described in Section II-A is not straightforward. The total duration can be estimated solely by running the system until convergence.

We are going to extend the model describe in Section II-A to allow one to modulate the speed of the motion generated by the dynamical system in Eq. (1).

While this could easily be achieved with a time-dependent dynamical system. Here we wish to preserve the time-independency of the motion generator as it provides appealing properties for on-the-fly replanning of the trajectory.

To provide a means of controlling the timing of the motion when generated by the autonomous dynamical system given in Eq. (1), we define a velocity multiplier λ. This multiplier modifies the original dynamics, by modulating the velocity mean $\mu^{\dot{\xi}}$ and the covariance $\Sigma^{\dot{\xi},\xi}$ as follows:

$$\tilde{\mu}_k^{\dot{\xi}} = \lambda \mu_k^{\dot{\xi}} \tag{2}$$

$$\tilde{\Sigma}_k^{\dot{\xi}\xi} = \lambda \Sigma_k^{\dot{\xi}\xi} \tag{3}$$

$$\dot{\xi} = \hat{\tilde{f}}\left(\xi\right) = \sum_{k=1}^{K} h_k\left(\xi\right)\left(\tilde{\mu}_k^{\dot{\xi}} + \tilde{\Sigma}_k^{\dot{\xi}\xi}\left(\Sigma_k^{\xi}\right)^{-1}\left(\xi - \mu_k^{\xi}\right)\right) = \lambda \hat{f}\left(\xi\right) \tag{4}$$

To allow for gradual and on the fly adaptation of the motion's duration so as to reach a position ξ^g in a given time T, we compute our multiplier at each time step as follows:

$$\lambda^{t_{i+1}} = \lambda^{t_i} + k_p\left(\hat{T}^{t_i} - T\right) - k_d\left(\hat{T}^{t_i} - \hat{T}^{t_{i-1}}\right) \tag{5}$$

where t_i is a time at the i^{th} controlling step, $t_{i+1} = t_i + \Delta t$, $t_0 = 0$; λ^{t_i} is the velocity multiplier, $\lambda^{t_0} = 1$; k_p and k_d are user defined proportional and derivative gains that control for the reactivity of the system. $\hat{T}^{t_i}$ is the *total* (from the onset to the offset) motion duration as computed at time t_i. $\hat{T}^{t_i}$ is estimated by integrating forward the following equation starting from $j = i$ till convergence to the attractor.

$$\xi^{t_{j+1}} = \xi^{t_j} + \lambda^{t_i} \sum_{l=1}^{L} \dot{\hat{\xi}}^{\{t_j + l\Delta t'\}} \Delta t' \tag{6}$$

To ensure stability of the new estimate in Eq. (4) and avoid trajectory distortions, we reduce the sampling rate for calculation to $\Delta t' = \Delta t / L$. However, the actual command is sent to the robot at the rate Δt. For the experiment, we set $\Delta t = 0.02$ and $L = 10$.

Algorithm 1 summarizes the steps followed during the reproduction.

The trajectories generated by the timing controller in Eq. (4) and the original controller in Eq. (1) follow the same Cartesian path, as the multiplier λ equally affects all components of the state vector ξ. Due to the same reason and the adaptive integration step in Eq. (6), the stability of the system in Eq. (4) is also not distorted.

Algorithm 1 Catching a ball

———————-Training———————————
$[\dot{\mathbf{x}}, \dot{\mathbf{o}}, \dot{\rho}] = \hat{f}(\mathbf{x}, \mathbf{o}, \rho) \Leftarrow$ learn the estimate of the dynamics of the robot's motion.

$\ddot{\mathbf{x}}_b = \hat{f}_b(\dot{\mathbf{x}}_b) \Leftarrow$ learn the estimate of the ball's motion.
$\lambda^{t_0} \Leftarrow 1.0$
———————-Motion Generation———————————
loop
  ———————-Predicting of the ball motion———————
  if (ball is detected) **then**
    $\mathbf{x}_b^{t_i} \Leftarrow$ from vision
    $[\mathbf{x}_b]^{t_i..t_N} \Leftarrow$ generate an estimate of the ball trajectory through $\hat{f}_b(\dot{\mathbf{x}}_b)$; see Sec. III-A.

    if (ball is catchable) **then**
      Determine the catching position and orientation ξ^g and the desired motion duration T; see Sec. III-B.
    else
      Stop and go back to the rest posture.
      Exit the loop.
    end if (ball is catchable)

  end if (ball is detected)
  ———————-Generating the robot motion———————
  $\dot{\xi}^{t_i} \Leftarrow \lambda^{t_i} \hat{f}(\xi^{t_i})$.
  $\hat{T}^{t_i} \Leftarrow$ estimate the reaching time by integrating Eq. (6) till $\|\xi^{t_j} - \xi^g\| < \epsilon$.
  $\lambda^{t_{i+1}} \Leftarrow$ update the timing constant through Eq. (5)
  $\theta^{t_i} \Leftarrow$ find the IK solution for $\xi^{t_{i+1}}$; see Sec. III-D
  Send θ^{t_i} to the robot and get the feedback from motors.
  if $(|\xi^{t_i} - \xi^g| < \epsilon)$ **then**
    Exit the loop.
  end if
end loop

C. DMP with timing controller

In contrast with autonomous DS, duration of motion generated with a time-dependent dynamical system, such as DMP, can be computed explicitly, if the time to target is known in advance before generating the trajectory. However, when the desired motion's duration to reach to the target is changed while in motion, one can no longer find an appropriate τ to control for the rest of the motion, as the relation is no longer linear (unless one stops the robot and starts the motion again from the location of the perturbation; such a stopping would be brutal and bound to prevent the robot to reach the target). Hence similarly to time-independent DS, a method is required to adapt gradually and on the fly to the motion's duration.

The timing controller presented in Section II-B can be used to control for the duration of movement when generated by DMP as well. DMP is composed of a linear 2nd order dynamical system, to which we refer to as f_2 and a modulating term f_m (estimated through LWR) [19] and is given by the following set of equations:

$$\dot{\xi} = \tilde{\tau}\left(f_2\left(\xi^g, \xi\right) + f_m\left(\xi^g - \xi^{t_0}, z\right)\right) \quad (7)$$
$$\dot{z} = -\tilde{\tau}\alpha_z z \quad (8)$$
$$\tilde{\tau} = \lambda\tau \quad (9)$$

Where τ is a time constant which is correspond to the inverse of the duration T^* of the learned motion, i.e. $\tau = 1/T^*$; The state $\xi = [\xi_1; \xi_2]$ consists of the position of each degree of freedom ξ_1 and its velocity ξ_2; $\xi^g = [\xi_1^g; 0]$ is the target position with zero velocity; α_z is a parameter defined by user.

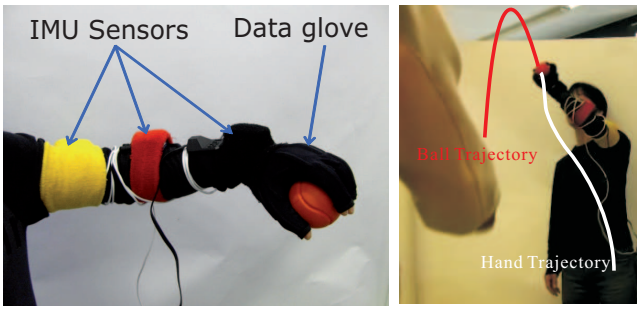


Fig. 1: *Left*: a motion capture set-up used to collect the training data. *Right*: the ball was thrown to the catcher from 3 meter distance.

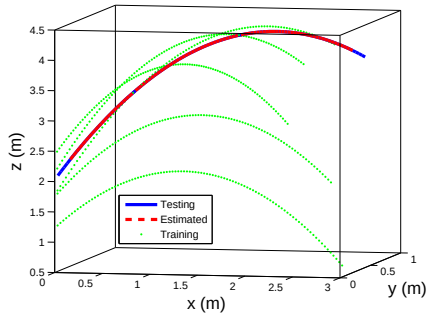
III. Experiment: Catching a flying ball

To validate the proposed timing controller, we conducted experiments where the iCub robot was required to catch a ball on the fly.

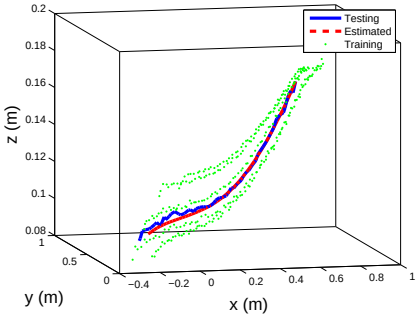
To obtain a training data set, a human provided forty demonstrations of different catching motions using a data glove and the X-Sens motion capture suit; see Fig. 1. The captured motions were mapped into the joint angles of the 7 degree of freedom (DOF) arm of the 53 DOF humanoid robot the iCub in real-time. This provided the teacher with immediate visual feedback of his actions and verify that the mapping resulted in the correct motion of the robot.

While capturing human demonstrations we did not record the ball's trajectories. Instead, we assumed that the ball's velocity vector was opposite to the palm's direction at the catching point. With the demonstrated trajectories, we trained the configuration of the robot's end-effector in the task space $\xi = [\mathbf{x}; \mathbf{o}; \rho]$ together with the corresponding velocities $\dot{\xi} = [\dot{\mathbf{x}}; \dot{\mathbf{o}}; \dot{\rho}]$. Where $\mathbf{x} \in \mathbb{R}^3$, $\mathbf{o} \in \mathbb{R}^3$, $\rho \in [0..1]$ are respectively the Cartesian position, the palm direction, and the degree of grasping (a normalized one-dimensional variable characterizing the degree of clench of the robot's hand; 1.0 corresponds to completely open hand ; 0.0 corresponds to the predefined grasping configuration).

For making a robot being able to catch a flying object, one should proceed to: 1) estimate the ball's dynamics to predict the timing of the robot's motion; 2) estimate the duration of



(a) The ball motion of the catching task observed in the simulator.



(b) The ball motion observed in the real-world *rolling* experiment. See 5.

Fig. 2: The dynamics of the ball in the simulated environment (a) and in the real experiment (b) is learned using our estimate of dynamical system. Training (green dot) and testing (blue solid) trajectories, superposed with the estimated one (red dashed).

the robot's motion and the end-effector configuration at the catching moment; 3) generate a task-space trajectory of the motion that satisfies the temporal and spatial constraints; 4) resolve the inverse kinematics to find a suitable joint angle configuration. We address these problems as follows.

A. Estimation of the ball's motion dynamics

The dynamic model of the ball motion can be learned using our dynamical system estimator described in Section II-A without attractors :

$$\ddot{\mathbf{x}}_b = \hat{f}_b\left(\dot{\mathbf{x}}_b\right) \tag{10}$$

where $\dot{\mathbf{x}} \in \mathbb{R}^3$ and $\ddot{\mathbf{x}} \in \mathbb{R}^3$ are the velocity and acceleration of a ball in cartesian space respectively.

Encoding with our generic DS provides an efficient way to model a dynamics of a moving object solely by observing examples of the object's displacement in space and without any prior information on the physical properties of the object such as its mass, density, etc.

To estimate the ball's dynamics, a ball was thrown 5 to 6 times with random velocities in the experimental environments; and its trajectories were recorded. Once learned, the system $\hat{f}_b$ is used to predict the ball trajectory given its position at the previous time step. Estimation of the ball's trajectory from the learned system $\hat{f}_b$ is depicted in Fig. 2.

B. Catch point determination

The robot starts tracking the ball when it is inside of a pre-defined 3D measuring region. The robot estimates the trajectory of the ball using the approach discussed in the previous section. To determine the catching configuration, the robot first verifies that the ball is catchable, by checking the intersection line of the estimated ball trajectory and the workspace of the robot arm. If the ball is catchable, the catching time and end-effector configuration are chosen so as to minimize the motion of the end-effector along the intersected path [4]. The estimated end-effector configuration at the target ξ^g is mapped into the attractor of the dynamical system given by Eq. (4). The desired reaching time (T) in Eq. (5) is assigned to the estimated catching time.

C. Task-space motion generation

The robot further starts to generate a motion of the end-effector and the fingers using the suggested approach. When the robot receives updated information of end-effector catching configuration and catching time, the timing controller gradually re-estimates the motion's duration by integrating the trajectory forward and properly adapting the velocity according to Eq. (4) - (6).

D. Inverse Kinematics

Finally, the damped least squares method [20] is used to convert the generated position and the palm's direction into joint angles.

For the fingers motion, we defined the two finger configurations: fully stretched ($q_1^{finger} \in \mathbb{R}^9$) and closed ($q_0^{finger} \in \mathbb{R}^9$). The trajectory of the 9 DOF of fingers was generated through the following equation with the learned degree of grasping ρ and the two finger configurations :

$$q^{finger} = \rho q_1^{finger} + (1-\rho)\, q_0^{finger}. \tag{11}$$

E. Experimental results

The result we have obtained so far in both simulated and real environments [1], confirm that the iCub endowed with the proposed DS and its timing controller manages to catch the ball on the fly successfully; see results in Fig. 4-6. Though the proposed algorithm generates the task space motion that should be further converted into joint angles, it is still sufficiently fast to generated motion in real-time at the frequency of 50 Hz.

[1] http://lasa.epfl.ch/~seungsu/humanoids2010.wmv

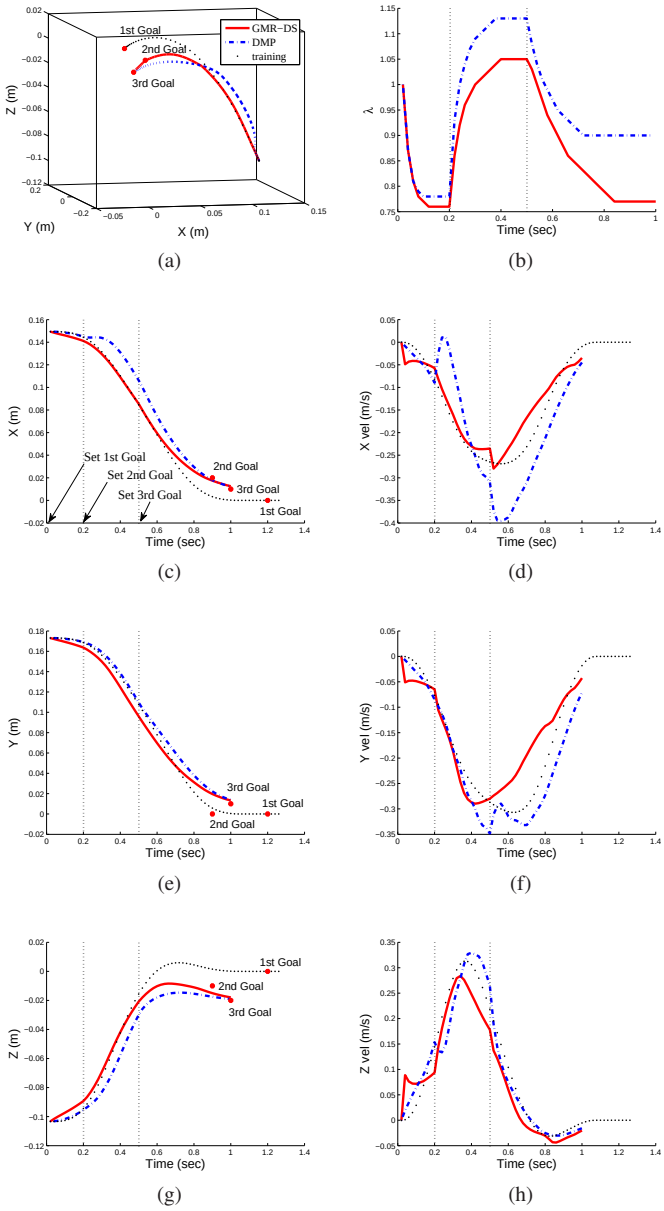


Fig. 3: We contrast adaptation to strong perturbations when using our timing control to adapt timing of either the DS (red solid line) or the DMP (blue dashed line) controllers. Both systems are trained on human demonstrations of catching motions (light hashed lines). Spatial and temporal perturbations were introduced by displacing the target twice at $t = 0.2$ sec and $t = 0.5$ sec. The timing controller gains were set to $k_p = 0.5$ and $k_d = 0.001$. (a, c-h) The new trajectories generated by DS and DMP reach the target accurately, while reproducing the demonstrated motion pattern adapting to spatial and temporal perturbations.

We also applied this method to control for motion duration in a time-dependent dynamical system encoding, defined by DMP; see Fig. 3. The controller gradually change λ when it receive more accurate target position and the motion's duration, so as to reach the target on time, whatever the motion is encoded by our DS or DMP. Since the change is gradual, it must be sufficiently quick to allow rapid adaptation to displacement. This depends on the gain parameters. These must be set so as to allow a reasonable acceleration peak. Note that by controlling only for the end-effector's position and by relying on an inverse kinematic controller for controlling for the joint, this may lead to too large acceleration peaks at the level of the joints and hence a conservative approach in the setting of the gains should be taken. In future work, we will exploit the generalized inverse kinematics method we developed in [21] for balancing a controller in joint and cartesian positions.

IV. Conclusion

In this paper we exploited the robustness of time-independent motion encoding through autonomous dynamical systems and developed a method to control for motion duration while remaining time-independent. The timing controller gradually speeds up or slows down the learned motion (DS and DMP), and hence adheres to precise temporal constraints. We validated the approach in the experiment where the robot iCub successfully catches a ball in motion.

V. Acknowledgement

This work was supported by EU Projects *First-MM* (FP7-ICT-248258) and *AMARSI* (FP7-ICT-248311).

References

[1] Elena Gribovskaya and Aude Billard, "Learning Nonlinear Multi-Variate Motion Dynamics for Real- Time Position and Orientation Control of Robotic Manipulators," in *Proceedings of IEEE-RAS International Conference on Humanoid Robots*, 2009.

[2] Won Hong and Jean-Jacques E. Slotine, "Experiments in hand-eye coordination using active vision," in *The 4th International Symposium on Experimental Robotics IV*, London, UK, 1997, pp. 130–139, Springer-Verlag.

[3] M. Zhang and M. Buehler, "Sensor-based online trajectory generation for smoothly grasping moving objects," in *Proceedings of the IEEE International Symposium on Intelligent Control*, 1994, pp. 16–18.

[4] U. Frese, B. Bauml, S. Haidacher, G. Schreiber, I. Schaefer, M. Hahnle, and G. Hirzinger, "Off-the-shelf vision for a robotic ball catcher," in *Proceedings of IEEE/RSJ Intl Conf. on Intelligent Robots and Systems*, 2001, vol. 3, pp. 1623–1629.

[5] Marcia Riley and Christopher G. Atkeson, "Robot catching: Towards engaging human-humanoid interaction," *Auton. Robots*, vol. 12, no. 1, pp. 119–128, 2002.

[6] A. Namiki and M. Ishikawa, "Robotic catching using a direct mapping from visual information to motor command," in *Proceedings of IEEE Intl Conf. on Robotics and Automation*, sept. 2003, vol. 2, pp. 2400 – 2405 vol.2.

[7] Ga-Ram Park, KangGeon Kim, ChangHwan Kim, Mun-Ho Jeong, Bum-Jae You, and Syungkwon Ra, "Human-like catching motion of humanoid using evolutionary algorithm(ea)-based imitation learning," in *Robot and Human Interactive Communication, IEEE International Symposium on*, 27 oct. 2009, pp. 809 –815.

[8] T. Senoo, A. Namiki, and M. Ishikawa, "Ball control in high-speed batting motion using hybrid trajectory generator," in *Proceedings of IEEE Intl Conf. on Robotics and Automation*, 15-19 2006, pp. 1762 –1767.

[9] Jens Kober, Katharina Mulling, Oliver Kromer, Christoph H. Lampert, Bernhard Scholkopf, and Jan Peters, "Movement templates for learning of hitting and batting," in *Proceedings of IEEE Intl Conf. on Robotics and Automation*, May 2010, pp. 853–858.

[10] Stefan Schaal and Christopher G. Atkeson, "Open loop stable control strategies for robot juggling," in *Proceedings of IEEE Intl Conf. on Robotics and Automation*, 2-6 1993, pp. 913 –918 vol.3.

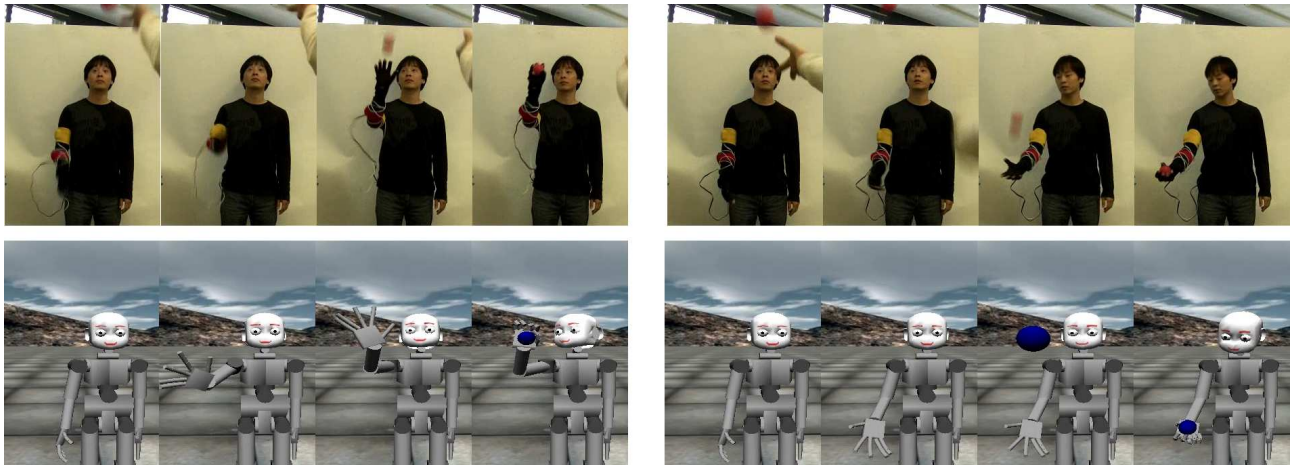


Fig. 4: Top: A human teacher demonstrates a ball catching motion starting from different configuration of the hand and palm. The demonstrations are encoded with an autonomous DS, Eq. (4). Bottom: The simulated iCub robot uses the learned DS to generate a motion (Cartesian position of the end-effector, palm orientation and clench) for catching a ball.

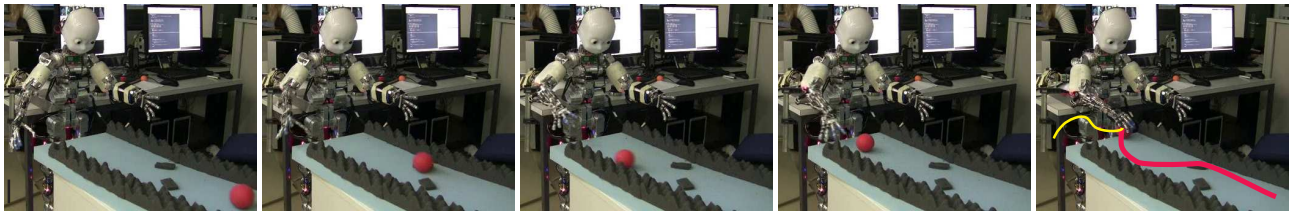


Fig. 5: The iCub has to catch the ball rolling down the slope before the latter falls down. During learning, the robot observes the motion of the ball rolling along the free slope. During reproduction, several obstacles are set that deform the ball's motion. The robot can roughly predict the ball's trajectory. However it still should continuously adapt both spatially and temporally to accommodate itself to the perturbed ball's dynamics.

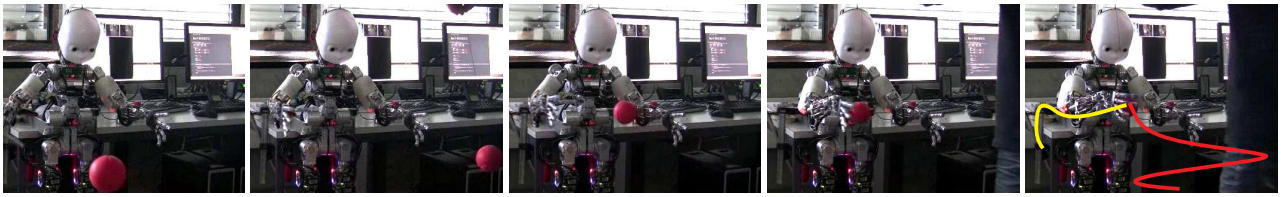


Fig. 6: The iCub has to catch a ball that bounces on a wire attached to a human hand. Again, the robot learns the motion of the ball, however, due to variability in the motion of a human pulling the wire, the robot should continuously readapt the motion while approaching the ball.

[11] Stefan Schaal, Dagmar Sternad, and Christopher G. Atkeson, "One-handed juggling: A dynamical approach to a rhythmic movement task," *Journal of Motor Behavior*, vol. 28, pp. 165–183, 1996.

[12] A.A. Rizzi and D.E. Koditschek, "Further progress in robot juggling: solvable mirror laws," in *Proceedings of IEEE Intl Conf. on Robotics and Automation*, 8-13 1994, vol. 4, pp. 2935–2940.

[13] Elena Gribovskaya, Seyed Mohammad Khansari-Zadeh, and Aude Billard, "Learning Nonlinear Multivariate Dynamics of Motion in Robotic Manipulators," *International Journal of Robotics Research*, 2010.

[14] A. L. Barker, D. E. Brown, and W. N. Martin, "Bayesian estimation and the kalman filter," *Computers and Mathematics with Applications*, vol. 30, no. 10, pp. 55 – 77, 1995.

[15] Mark Cannon and Jean-Jacques E. Slotine, "Space-frequency localized basis function networks for nonlinear system estimation and control," *Neurocomputing*, vol. 9, no. 3, pp. 293 – 342, 1995, Control and Robotics, Part III.

[16] Stefan Schaal and Dagmar Sternad, "Programmable pattern generators," in *Intl Conf. on Computational Intelligence in Neuroscience*, 1998, pp. 48–51.

[17] A.J. Ijspeert, J. Nakanishi, and S. Schaal, "Trajectory formation for imitation with nonlinear dynamical systems," in *Proceedings of IEEE/RSJ Intl Conf. on Intelligent Robots and Systems*, 2001, vol. 2, pp. 752 –757 vol.2.

[18] Seyed Mohammad Khansari-Zadeh and Aude Billard, "BM: An Iterative Method to Learn Stable Non-Linear Dynamical Systems with Gaussian Mixture Models," in *Proceedings of IEEE Intl Conf. on Robotics and Automation*, 2010.

[19] A.J. Ijspeert, J. Nakanishi, and S. Schaal, "Movement imitation with nonlinear dynamical systems in humanoid robots," in *Proceedings of IEEE Intl Conf. on Robotics and Automation*, 2002, vol. 2, pp. 1398–1403.

[20] C. Wampler, "Manipulator inverse kinematic solutions based on vector formulations and damped least-squares methods," in *IEEE Transactions on Systems, Man and Cybernetics*, 1986, vol. 16, pp. 93–101.

[21] M. Hersch and A. Billard, "Reaching with Multi-Referential Dynamical Systems," *Autonomous Robots*, vol. 25, no. 1-2, pp. 71–83, 2008.