Learning Motion Dynamics to Catch a Flying Object

Seungsu Kim, Elena Gribovskaya and Aude Billard

I. Introduction

In this abstract, we consider a novel approach to control *the timing of motions* when these are encoded with autonomous dynamical systems. Accurate motion timing is highly important if a robot has to *synchronize* its movements with the dynamics of a moving object. In previous work of ours [1], we developed an approach to learning motion dynamics from demonstrations performed by a human user. Encoding motions as autonomous dynamical systems(DS) provides an efficient way to generate and adapt motions to external perturbations, while ensuring high accuracy at the target.

Catching an object on the fly requires very accurate estimate of the dynamics of motion of the object. This depends on accurate sensing which cannot always be ensured in robotics. One thus needs to quickly-adapt the planned trajectory so as to catch the object on time when receiving a novel (more accurate) estimate of the object's motion. DS encoding offers a very powerful way to recompute on-the-fly trajectories, while ensuring that both the dynamics is correctly regenerated.

Here, you investigate the ours of DS to encode control of the motion's duration so as to speed up or slow down a motion during reproduction and therefore to adhere to temporal constraints. We validate the proposed method in an experiment where the i-Cub robot learns to catch a ball on the fly.

II. Algorithm Overview

A motion representation is learned as a first-order autonomous dynamical system:

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

where ξ defines the configuration of the robot's end-effector in the task space $\xi = [\mathbf{x}; \mathbf{o}; \rho]$, $\mathbf{x} \in \mathbb{R}^3$, $\mathbf{o} \in \mathbb{R}^2$, $\rho \in [0..1]$ are respectively the Cartesian position, the palm direction, and the degree of grasping (a normalized one-dimensional variable characterizing the clench of the robot's hand); $\dot{\xi} = [\dot{\mathbf{x}}; \dot{\mathbf{o}}; \dot{\rho}]$ are the corresponding velocities. The demonstrated trajectories together with velocities are encoded with Gaussian Mixture Models (GMM); the estimate $\hat{f}(\xi)$ is further built with Gaussian Mixture Regression (GMR) as the expectation of the conditional probability; see [1] for details.

To provide a means of controlling timing along a motion, we design a controller which allows for gradual adaptation of the motion duration following Eq.1, for as to satisfy synchronization constraints and to reach the target ξ^* in a given time T. It control the velocity multiplier λ using a form of PD controller:

$$\xi^{t_{i+1}} = \xi^{t_i} + \lambda^{t_i} \sum_{l=1}^{L} \dot{\hat{\xi}}^{\{t_i + \frac{\Delta t}{L} l\}} \frac{\Delta t}{L} \tag{2}$$

$$\lambda^{t_{i+1}} = \lambda^{t_i} + k_p \left(\hat{T}^{t_i} - T\right) - k_d \left(\hat{T}^{t_i} - \hat{T}^{t_{i-1}}\right) \tag{3}$$

where t_i is a time at i^{th} controlling step, $t_{i+1} = t_i + \Delta t$, $t_0 = 0$; λ^{t_i} is a velocity multiplier, $\lambda^{t_0} = 1$; k_p and k_d are the proportional and derivative gains respectively; $\hat{T}^{t_i}$ is an estimated motion duration starting from the beginning of motion at time t_0 as calculated at time t_i; It can be estimated by calculating Eq. 2 iteratively until the distance between ξ^t and the attractor ξ^* is smaller than user defined tolerance. When the learned DS given by Eq.1 is modulated with the multiplier λ^{t_i}, the resulting trajectory might be depart from that original computed by the DS. To reduce a negative effect of a too big integration step, we integrate the dynamical law $\hat{f}$ L times, before sending an actual command to the robot; see Eq. 2.

III. Experiment: Catching a Flying Ball

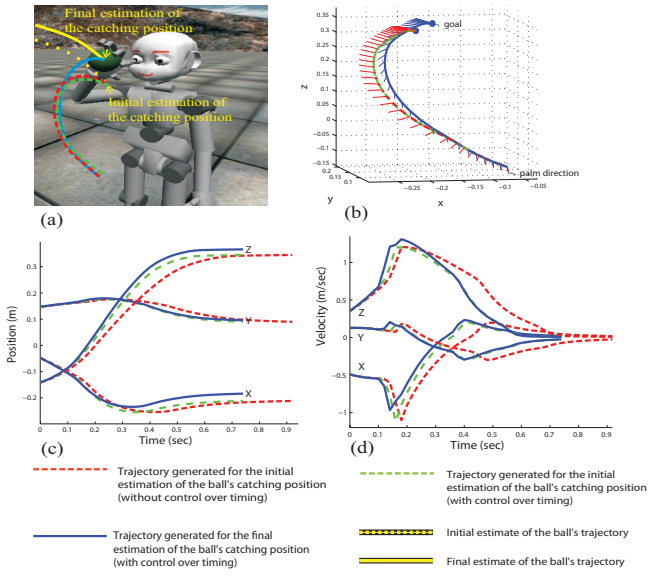

Fig. 1: The robot gradually estimates the ball's trajectory and re-estimates the catching position and motion duration. (a),(b) If the robot relied on the initial estimate of the ball's trajectory, then it would missed the ball. Gradual re-estimation of the ball's ballistics allows for on-line adaptation of the robot's motion in both time and space. (c),(d) Note, the duration of motion generated with the original dynamics Eq.1 is longer than with the improved system Eq.1-3.

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

We have validated the proposed controller in an experiment where the iCub robot catches a ball on the fly. To obtain a data-set for training a model of the motion, a human demonstrated forty different catching motions using a data glove and X-Sens motion capture suit. The captured motions were mapped into iCub joint angles in real-time, so the teacher could get an immediate visual feedback of his actions.

For making a robot being able to catch a flying object, one should resolve several problems, namely: 1) estimate the ball's ballistics to predict the timing of the robot's motion; 2) estimate the duration of the robot's motion and the end-effector configuration at the catching moment; 3) generating a task-space trajectory of the motion that satisfies temporal and spatial constraints; 4) resolving the inverse kinematics to find a suitable joint angle configuration. We tackle these problems as follows.

First, the ball's motion is modeled according to the Newtonian mechanics with the air drag, and a trajectory of the ball is estimated using Kalman filter [2]. Second, the catching time and position are chosen as to minimize the motion of the end-effector [3], the palm direction vector is defined so as to have a direction opposite to the ball's velocity vector at the moment of catching. The estimated end-effector configuration at target is mapped into the attractor of the motion dynamics Eq.1. The robot further starts to generate an end-effector trajectory in the task-space. It gradually re-estimates the motion duration by integrating the trajectory forward and adapts the velocity according to Eq.3-2. Finally, the task-priority redundancy control [4] is used to convert the generated positions into the joint angles. We assign the highest priority to the hand position and the palm direction.

The experimental results, we have obtained so far in a simulator[1], confirm that the iCub performed a catching motion with the proposed dynamical controller manages to catch the ball. However, the iCub in simulator couldn't catch the ball in most case because of the limitation of the simulation, even though it perform the catching motion on time. You can see the simulation result at this web link[2]. We are currently implementing the experiments on the actual physical robot.

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 9th IEEE-RAS International Conference on Humanoid Robots*, 2009.

[2] 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.

[3] 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 *Intelligent Robots and Systems, 2001. Proceedings. 2001 IEEE/RSJ International Conference on*, 2001, vol. 3, pp. 1623–1629 vol.3.

[4] S. Chiaverini, "Singularity-robust task-priority redundancy resolution for real-time kinematic control of robot manipulators," *Robotics and Automation, IEEE Transactions on*, vol. 13, no. 3, pp. 398–410, Jun 1997.

[1]The simulator contains a physical model of the world and, therefore, allows for realistic simulations

[2]http://www.youtube.com/ksrobot