4th International Conference on
Development and Learning and on Epigenetic Robotics
October 13-16, 2014. Palazzo Ducale, Genoa, Italy

A neural dynamics to organize timed movement: Demonstration in a robot ball bouncing task

Farid Oubbati, Mathis Richter, and Gregor Schöner
Institut für Neuroinformatik, Ruhr-Universität Bochum,
Universitätsstr. 150, 44780 Bochum, Germany
{farid.oubbati, mathis.richter, gregor.schoener}@ini.rub.de

***Abstract*—To address how different movement behaviors may be timed to sensory events while being flexibly organized in sequence, we propose a neural dynamic model of timed movement organization. Two layers of neural dynamics control the activation and de-activation of different elementary movements, while a third layer uses stable limit cycle oscillators to generate timed movement trajectories. Both the organization and the generation of the timed movements are coupled to online sensory information so that the system can compensate for perturbations by updating the movement trajectory while recovering the required movement timing. We formulate and demonstrate the approach in a robotic ball bouncing task. When the ball begins to fall, the robot arm moves to the interception point in a plane and initiates a rotatory motion of the racket timed such as to hit the ball with maximal velocity. When the ball is no longer falling or falling outside the reachable space, the robot moves the racket back toward baseline. A physics simulation is used to assess the model and demonstrate its capacity to handle perturbations of the ball trajectory.**

I. Introduction

Although they come naturally to us, activities like table tennis are complex and challenging. Within split-seconds we must perform and coordinate multiple tasks that include detecting the ball, estimating its future trajectory, preparing an action that will intercept the ball at the right time with the right orientation of the racket, and getting ready for the next interception. In addition, we need to control and update the movement trajectory of the racket to control the velocity vector at impact. All actions are continuously chained together and flexibly tuned to the environment.

Roboticists have studied problems of this kind not so much because of their practical importance, but because they exemplify core elements of autonomous action, in particular, the problem of timing a robot's action to external events. Specific robotic solutions to interception problems [1]–[4] are often very fast and accurate. The models typically address each catch as an isolated act, and flexible organization of sequences and their reorganization has not been a topic. Approaches based on learning movement primitives from human demonstration are adaptive in the sense that they learn from a given demonstrator [5]. During production, a learned primitive is selected or several are blended together [6]–[9]. The models produce periodic repetitions of timed acts, which may be updated to changing sensory estimates. Relatedly, periodic timed motor acts may be generated from nonlinear oscillators into which an effector system is coupled [10], [11]. Such systems may repeat timed actions that can be synchronized with sensed events, but are limited with respect to the complexity and heterogeneity of the timed actions.

We have previously developed an approach for how different discrete timed movements can be organized into sequences that are coordinated flexibly in response to time-varying perceptual input [12]. That approach was inspired by analogies with timed human movement [13], [14] and based on neural principles [15]. Timed discrete motor acts were generated from oscillators that are activated and deactivated based on sensory input [16], so that they are reliably timed relative to sensed discrete events while remaining coupled to ongoing, time varying sensory input. Previous robotic versions [17], [18] used this principle to organize initiation, updating, and termination of a single timed action, while our recent work [12] organized two actions. These were a movement toward an interception point and a hitting movement with a racket held by a robot arm that drives a ball up an inclined plane on which obstacles perturb the ball's trajectory.

In the present paper we show how the same framework can accomplish a ball bouncing task that involves a larger set of task variables. The ball is kept in the air by repeatedly hitting it with a racket in a way that controls the two-dimensional racket orientation as well as the three-dimensional racket location. We address how the approach handles perturbations to the ball by updating the planned robotic movements so that the timing of the hitting action remains correct.

Ball bouncing and juggling tasks have often been used in robotics to exemplify principles of timed movement, often with solutions from control theory [19], [20] or optimal control [21]. The behavioral organization of such tasks has been addressed through finite-state machine algorithms [8] and hierarchically organized dynamical systems [21]. Our approach is closer to the latter method, but uses continuous time throughout, from which the discrete events at which actions are initiated and terminated emerge through

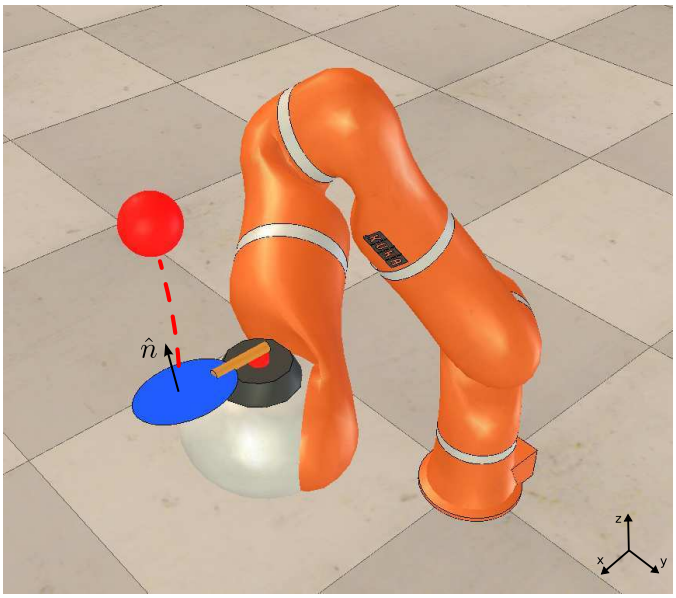


Fig. 1: Screenshot from the simulator showing the KUKA LBR+ arm holding a racket and performing the ball bouncing task.

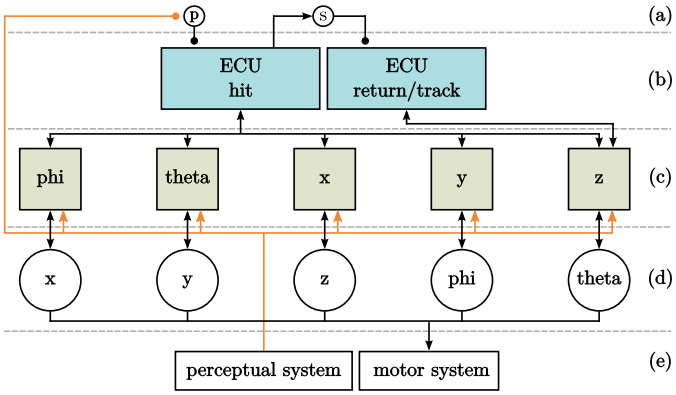


Fig. 2: Architecture that controls the robotic arm and organizes its behaviors in a ball bouncing task. It consists of (a) constraint nodes that encode the sequence on the higher level, (b) abstract ECUs (see Fig. 3) for the behaviors 'hit' and 'return/track', (c) modules of timed movement organization (see Fig. 4) for each movement variable x, y, z, ϕ, and θ, (d) timing dynamics (see Section III-C) for each movement variable, and (e) the sensory-motor system.

dynamical instabilities.

II. Task setting

In our simulation, a ball is bounced by a redundant, seven degree of freedom robotic arm (KUKA LBR+) holding a table tennis racket. The simulation is physically realistic and implemented in Matlab and v-rep[1] (see Fig. 1 for a screenshot).

When the ball is falling downward, the robot moves the racket toward the predicted hitting point with the ball along the x, y, z axes and orienting the racket at the same time. The hit is timed so that the racket intercepts the ball at maximum velocity and at an orientation ϕ, θ that will drive the ball toward the middle of the arena, keeping it in play.

The task highlights the following problems that our approach is addressing: The process of bouncing the ball requires timed movements that need to be sequentially activated and deactivated in time. The repetitive nature of the task requires that those timed movements be reinitiated for different trajectories of the ball. By introducing unpredictable perturbations to the ball, we can show that the system can quickly react to changes and is open to update the movement plan at any time.

III. Architecture

The architecture consists of four layers on top of a sensory-motor system (see Fig. 2 for an overview). All components within these four layers are dynamical systems (differential equations). Some are *neural dynamics* that have fixed points at an 'on' or 'off' state, which we use to activate and deactivate movements. Others are *neural oscillators* that have periodic solutions, which we use to time movements. These dynamical systems receive input from the perceptual system that is updated all the time dependent on input from the camera. The output of the differential equations is sent to the motor system and controls joint servo-controllers for the robot arm. All dynamical systems are time continuous; we solve the differential equations numerically in real time. Discrete events like starting to move, hitting, or moving back emerge from that time continuous set of differential equations through instabilities that we will explain. Although the architecture is inspired by neural models, it is designed by hand and has no learning so far. Many of its components are constructed from a principled approach, however, greatly reducing the number of parameters that need individual tuning.

Before we go into detail we will look at the individual components that make up the architecture.

A. Executive control units

The *behavioral organization* [22], [23] of the architecture, that is, the activation and deactivation of different motor actions, is based on *executive control units (ECU)*. Each ECU, depicted by the blue boxes in Fig. 2b and shown individually in Fig. 3, controls the downstream elements of the architecture that are associated with a motor action. Two motoric actions are generated: 'Hit' moves the end-effector toward the predicted hitting point of the ball along the x, y, and z axis and orients the paddle along ϕ and θ (see Fig. 1). The second motor action, 'return/track', moves the end-effector back toward a reference plane along

[1] v-rep is an open source simulator (coppeliarobotics.com).

the z axis while tracking the position of the ball along the x and y axes.

Each ECU is implemented through two activation variables, v_{int}, $v_{\text{CoS}} \in \mathbb{R}$ (Fig. 3). A sigmoidal function, $f(v)$, transforms negative activation levels into zero (i.e., the variable is 'off') and positive activation levels into one (i.e., the variable is 'on'), with a smooth interpolation around zero levels of activation. The *intention node*, $v_{\text{int}}(t)$, represents whether the ECU is activated and therefore impacts on any downstream parts of the architecture. The successful completion of a downstream motor act is represented by the *condition of satisfaction (CoS) node*, v_{CoS}.

The time courses of both nodes are generated from differential equations that receive sensory data as time-dependent input. For v_{int} we have

$$\tau \dot{v}_{\text{int}}(t) = -v_{\text{int}}(t) + h + S_{\text{int}}(t) + c_{vv} f(v_{\text{int}}(t)), \qquad (1)$$

which describes the rate of change $\dot{v}_{\text{int}}(t)$ of the activation of the intention node $v_{\text{int}}(t)$, given a resting level $h < 0$, a time-dependent external input $S_{\text{int}}(t)$, the self-excitation $f(v_{\text{int}}(t))$ of the node with a strength $c_{vv} \in \mathbb{R}_+$, and a time constant τ. We use an analogous equation for v_{CoS}.

Equations of this type are adapted from neural field models of cortical population activity [24] and form the mathematical core of the neural dynamics approach to cognitive robotics. This approach employs a time-continuous form of recurrent neural networks to model the evolution of population activity in the higher nervous system [25]. In neural dynamics, attractors are the functional states, and their instabilities induced by changing inputs bring about state changes.

The neural dynamics of Eq. 1 has an attractor at the resting level, $h < 0$, as long as there is no external input $S_{\text{int}}(t)$. At resting level, the sigmoided activation variable, $f(v_{\text{int}}(t))$, returns zero. With sufficiently strong excitatory input, the node becomes activated, generating positive output, which is stabilized by the self-excitation. Inhibitory input may destabilize that attractor so that activation falls back below zero. The stability properties of the attractors enable such systems to remain linked to sensory inputs while making detection, selection, or memory decisions.

The coupling of the two nodes sets up an ECU Fig. 3: The intention node receives external input from other ECUs and inhibitory input from its own CoS node, which in turn receives excitatory input from the intention node and from the sensory-motor system

$$S_{\text{int}} = -c_{\text{ic}} f(v_{\text{CoS}}) S_{\text{boost}}(t), \qquad (2)$$

$$S_{\text{CoS}} = c_{\text{ci}} f(v_{\text{int}}) C(t), \qquad (3)$$

where $c_{\text{ic}}, c_{\text{ci}} \in \mathbb{R}_+$ are weights, $S_{\text{boost}}(t)$ is an excitatory signal which activates the ECU, and $C(t)$ is input from a perceptual system that signals the state of the motor action. The idea is that the CoS node is activated when

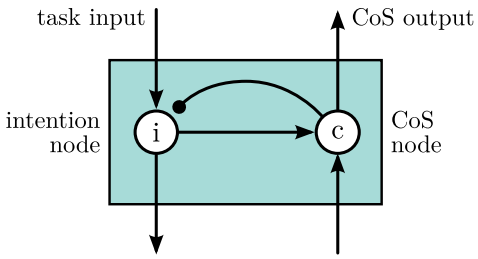


Fig. 3: Executive control unit (ECU) as part of our neural dynamic architecture. Excitatory connections are illustrated by normal arrows, inhibitory connections by lines ending in circles. See text for details.

sensory information signals that the intended motor action has been achieved. The CoS node then turns the intention node off and subsequently goes back to the resting state itself, as input from the intention node falls away. An ECU thus executes a motor action once until it is done. It may be reactivated by new input from other ECUs or the task level. The ECUs shown in Fig. 2b are directly activated by task input that comes from perception.

Coupling among ECUs organizes sequences of behavior. The ECU for 'hit' suppresses the ECU for 'return/track', so that they cannot be activated at the same time. Such competitive constraints are implemented by *suppression nodes* (illustrated by nodes marked with 's' in Fig. 2a), governed by an activation dynamics analogous to Eq. 1, which receives excitatory input from the intention node of one ECU (e.g., 'hit') and inhibits the intention node of the other ECU (e.g., 'return from hit'). Additionally to suppression between ECUs, we can express precondition constraints. For instance, the ECU for 'hit' should only be activated if there is perceptual information about the ball moving downward. The intention node of the ECU is inhibited by a *precondition node* (marked with 'p' in Fig. 2a) until that node is in turn turned off by perceptual information becoming available. Both precondition and suppression constraints will also show up between ECUs in the next lower level of the architecture, which we will explain now.

B. Timed movement organization

Layer (c) of the architecture (green boxes in Fig. 2c) organizes the actual phases of timed movements. It does so separately for the five movement variables, the x-, y- and z-coordinates of the end-effector and the orientation ϕ and θ of the racket. For each movement variable, three ECUs control the resetting of the motor coordinate frame to the initial state of the effector (*preparatory ECU*), the generation of a timed trajectory (*movement ECU*), and the maintenance of a fixation position that may move with moving input (*fixation ECU*). This is illustrated in Fig. 4. Each of these ECUs consists of the two activation variables illustrated in Fig. 3, except for the fixation ECU. It does not turn itself off but stays on until suppressed

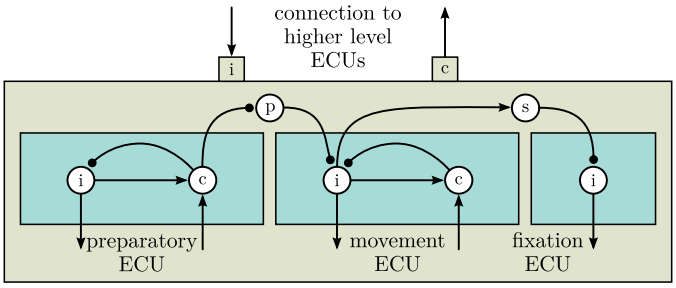


Fig. 4: A module for the *timed movement organization* consists of and organizes three ECUs that are part of a timed movement. Precondition constraints are illustrated as circles labeled 'p', suppression nodes as circles labeled 's'. Excitatory connections are illustrated by normal arrows, inhibitory connections by lines ending in circles. See text for details.

by the movement ECU via a suppression node. Because the coordinate frame needs to be reset before a movement takes place, a precondition node inhibits the intention node of the movement ECU.

The outputs of all intention nodes within this layer are used as weights that enable different contributions to another type of dynamical system, which we will now explain.

C. Timing

Layer (d) of the architecture (illustrated as larger circles in Fig. 2d) builds on neural oscillators to generate trajectories for the five movement variables, x, y, z, ϕ, and θ. That oscillators may drive timed movement is consistent with what we know about movement generation in nervous systems [26]. If the oscillators have stable limit cycles, then coupling among such oscillators or coupling to timed signals helps coordinates movements without perturbing the spatial paths of the movement much [27]. We use the Hopf normal form [28] as a mathematical formalization because it has a limit cycle attractor that is perfectly harmonic and can be analytically computed, so that all parameters have direct meaning. We write down the timing model for one movement variable, the x-component of the end-effector position, the other two being analogous:

$$H(x,k) = \begin{pmatrix} \lambda & -\omega \\ \omega & \lambda \end{pmatrix} \begin{pmatrix} x - r - x_{\text{init}} \\ k \end{pmatrix} - \left((x - r - x_{\text{init}})^2 + k^2\right) \begin{pmatrix} x - r - x_{\text{init}} \\ k \end{pmatrix}. \quad (4)$$

Because two variables are needed in a first order differential equation to make an oscillation, this equation includes an auxiliary variable, k, that has no other role. τ is the same time scale parameter mentioned before. The Hopf equation has a limit cycle attractor that is a harmonic function of amplitude, $r = \sqrt{\lambda}$ and frequency, ω. We choose λ so that $r = (x_{\text{target}} - x_{\text{init}})/2$ and have shifted the coordinate frame so that the limit cycle moves between x_{init} and x_{target}. The frequency, ω, is chosen so that the desired duration, T, of the movement emerges as half the cycle time, $T = \pi/\omega$.

The movement is continuously updated from online sensory information. Such updates are induced when the target location, x_{target}, shifts. In order for the system to reach the updated target location in the remaining time to impact, the movement speed of the end-effector must be adjusted. This demand is based on an analogy with human movement timing in which acceleration or deceleration is used to actively restore appropriate timing after perturbations [29], [30]. We implement the update mechanism by coupling into the Hopf oscillator a function that attracts the oscillator to the solution with the desired timing

$$C(x,k) = c_w \begin{pmatrix} x' - x \\ k' - k \end{pmatrix}, \quad (5)$$

$$x' = r(t)(1 - \cos(\omega t)), \quad (6)$$

$$k' = -r(t)\,\sin(\omega t) \quad (7)$$

(an alternative on-line updating rule is used in [12]). Herein, the primed variables are solutions to the Hopf equation that are initialized at the onset of the movement and take time varying sensory input about target location, $x_{\text{target}}(t)$ as a time varying parameter, $r(t) = (x_{\text{target}}(t) - x_{\text{init}})/2$. The parameter, c_w, is the coupling strength.

Typically, we want the oscillator to be active only for a single half-cycle in order to move the movement variable, x, from x_{init} and x_{target}. This is achieved by activating the Hopf oscillator through the movement ECU and then deactivating it once x has reached the target. Whenever the oscillator is not active, the fixation ECU activates a second contribution to the dynamics of the movement variable, creating a stable fixed point, x_{fix}, that is set to x_{init} initially and to x_{target} at the end of the movement. This second contribution makes sure that the movement variable is at the right start value before the movement begins, and remains at the final value after the movement has ended. In fact, for the movement in the x, y-plane, the fixed point is continuously computed from perception to be the position of the ball, so this fixation contribution essentially does tracking as well. In the complete equation

$$\tau \begin{pmatrix} \dot{x} \\ \dot{k} \end{pmatrix} = -a f(v_{\text{int,fix},x}) \begin{pmatrix} x - x_{\text{fix}} \\ k \end{pmatrix} + f(v_{\text{int,mov},x})\,(H(x,k) + C(x,k)), \quad (8)$$

the intention nodes for the fixation and movement ECU of the movement variable x are denoted by $v_{\text{int,fix},x}$ and $v_{\text{int,mov},x}$. Since the movement ECU suppresses the fixation ECU as illustrated in Fig. 4, the two contributions are not simultaneously active except for a short transition phase. a is a parameter that determines the relative strength of attraction to the fixed point.

Because the Hopf oscillator requires the initial position, x_{init}, of the movement variable to be known, the initial po-

sition must be stored before the movement is started. This purely internal operation is controlled by the 'preparatory ECU' through its intention node, $v_{\text{int,pre},x}$:

$$\dot{x}_{\text{init}} = f(v_{\text{int,pre},x})\,(-x_{\text{init}} + x_{\text{real}}), \tag{9}$$

through which x_{init} is updated to the current value of x_{real} when the preparatory ECU is 'on'.

D. Perceptual system

The neural dynamics of behavioral organization receive time varying input from the perceptual system (left box in Fig. 2e). Here, we trivialize the perception and use the known position of the ball from the simulator. In a previous and similar architecture, we have used a color-based segmentation to extract the position of a ball from a camera image [12]. The position of the ball provides input to a Kalman filter, which serves to extract time-varying estimates of the time-to-impact to a hitting plane at which interception is planned, as well as the velocity vector at the predicted interception point. That point is continuously updated whenever the ball is moving downward. It is stored in a neural activation field, which serves as a low-pass filter and is used to control the 'hit' ECU. The time-to-impact is used to control the 'hit' ECU as well, activating it whenever the time-to-impact falls below a threshold.

To keep the ball within range of the arm, we direct the ball toward the center of the hitting region with every hit. We derive the necessary normal vector $\hat{\boldsymbol{n}} = [n_x\,,\, n_y,\, n_z]^T$ of the racket from the predicted incoming velocity vector $\hat{\boldsymbol{v}}_{\text{i}}$ of the ball at the hitting point and its optimal outgoing velocity vector $\hat{\boldsymbol{v}}_{\text{o}}$ after the hit

$$\hat{\boldsymbol{n}} = \frac{\hat{\boldsymbol{v}}_{\text{o}} - \hat{\boldsymbol{v}}_{\text{i}}}{\sqrt{2\,(1 - \hat{\boldsymbol{v}}_{\text{i}}^T\,\hat{\boldsymbol{v}}_{\text{o}})}}. \tag{10}$$

From the racket's normal vector $\hat{\boldsymbol{n}}$ we compute the desired orientation of the racket along the two axes that we control:

$$\phi_{\text{des}} = -\arctan(\frac{n_y}{n_z}), \tag{11}$$

$$\theta_{\text{des}} = \arctan(\frac{n_x}{n_z}). \tag{12}$$

We control the arm such that the racket is at maximum velocity upon impact. In simulation, we implement the impact and hit of the ball by reversing its relative velocity with respect to the racket and adding the current racket velocity, factoring in energy loss.

E. Motor system

The motor system (right box in Fig. 2e) receives input from the dynamical systems that control the robotic arm. The dynamical systems described in Eq. 8 generate trajectories for the racket movement variables x, y, z, ϕ, and θ in task space that are defined in the coordinate system at the base of the robot. They are converted into joint angles $\boldsymbol{q}$ using the damped least squares (DLS) inverse method [31]

$$\dot{\boldsymbol{q}} = \boldsymbol{J}^T(\boldsymbol{J}\boldsymbol{J}^T + \lambda^2\,\boldsymbol{I})^{-1}\dot{\boldsymbol{p}}, \tag{13}$$

where $\boldsymbol{p} = [x, y, z, \phi, \theta, \psi]^T$ is the end-effector state vector in task space, $\boldsymbol{J}$ is the Jacobian matrix, $\boldsymbol{I}$ is the identity matrix, and $\lambda \in \mathbb{R}_+$ is a damping factor. The solution vector $\dot{\boldsymbol{q}}$ minimizes $\|\boldsymbol{J}\dot{\boldsymbol{q}} - \dot{\boldsymbol{p}}\|^2 + \lambda^2\|\dot{\boldsymbol{q}}\|^2$. For appropriate values of λ, the solution behaves well near singularities while providing a sufficiently fast convergence rate.

The joint angles $\boldsymbol{q}$ are then used to drive the joint servo-controllers of the robot arm.

IV. Evaluation & Results

We used the simulation environment to evaluate the performance of the system qualitatively in a number of situations that demonstrate core properties of the approach. The performance of the system is also presented in a supplementary video accompanying this paper.

We first demonstrate that the robot arm is able to bounce the ball. Fig. 5 shows trajectories of the ball and racket over time for an exemplary sequence of six consecutive hits. The model organizes all behaviors in time, generating a sequence of timed movements, and controls the racket to drive the ball toward the center of the hitting region. Fig. 6 shows more detailed time courses of elements of the model. The top panel shows whether a prediction for the hitting point is available, signaled by a activation peak in the perceptual field. Whenever such a prediction is available, the intention node of the 'hit' behavior (abbreviated by 'hit') is activated (second panel from the top). The bottom five panels show the movement variables x, y, z, ϕ, and θ over time. You can make out the six consecutive hits from the plot of the movement variable z. After each hit, the 'return/track' behavior is activated (abbreviated with 'r/t'), returning the end-effector to a fixed reference plane along the z-axis and to fixed reference orientations along ϕ and θ, but keeps tracking the position of the ball in x and y.

We now demonstrate that our model can flexibly react to unforeseen perturbations of the ball's trajectory. The initial pose of the robot arm and the initial position and velocity of the ball are the same as in the last experiment. Fig. 7 shows the trajectory of the ball and racket when we change the course of the ball while the arm is already moving toward the predicted hitting point. You can see that the robot moves toward the initially predicted hitting point but moves to the new prediction as soon as it is available. The model handles situations like this by adapting the sequence of timed movements and updating the movement parameters on the fly. Fig. 8 shows how this adaptation changes the movement of the racket. The speed profiles shown in Fig. 8 (right side) show sharp changes after the ball has been perturbed. The model is compensating for the perturbation and reaches the ball in time to perform the hit.

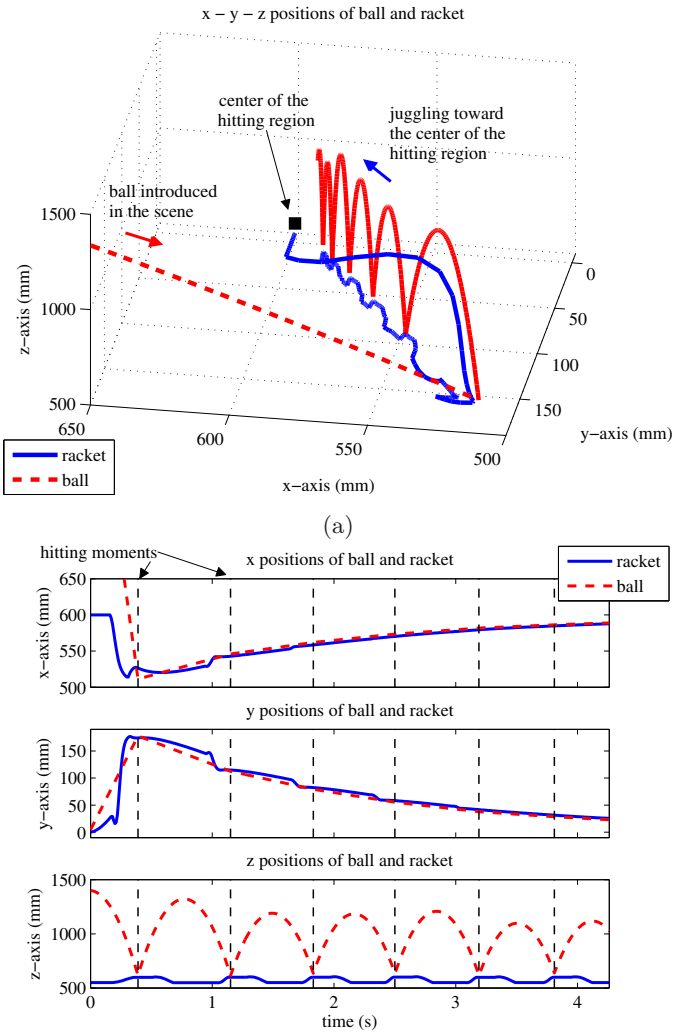


Fig. 5: Trajectories of the racket and the ball (a) in world coordinates and (b) separated into individual movement components over time. Plots show the first six consecutive hits of an exemplary trial. The racket is initially at the center of the hitting region.

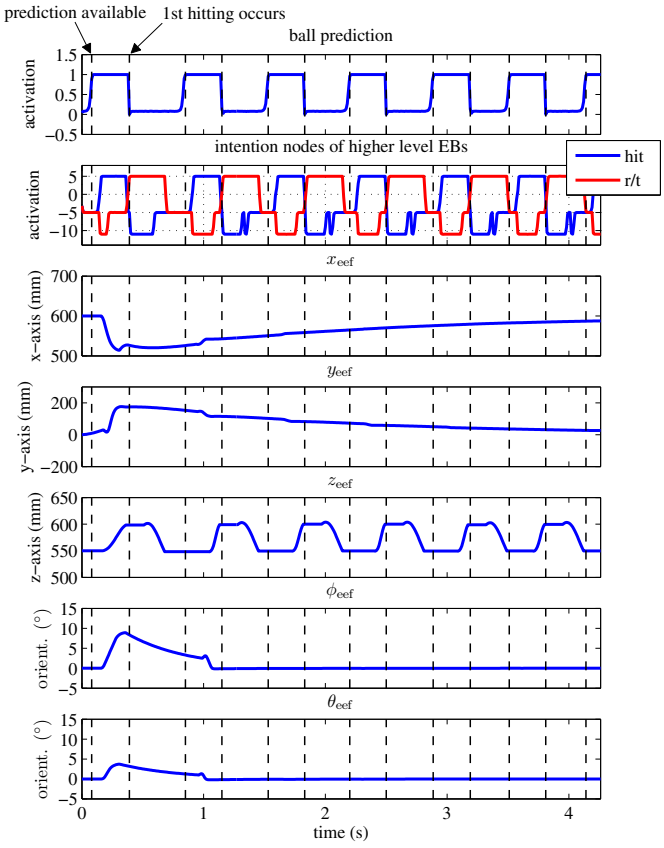


Fig. 6: Time courses of relevant variables and parameters of the architecture during the first six consecutive hits.

V. Conclusions

Movement planning and control are typically based on the mathematical frameworks of optimal control and control theory, while the sequential organization of different movements is most commonly organized algorithmically. Our approach, in part neurally inspired, addresses both the generation and the organization of timed movements through dynamical systems. Timing signals are generated by stable limit cycle oscillators, which are coupled to a neural attractor dynamics that performs executive control. Both dynamics are linked to online sensory information. As a result, the system is capable of responding to changes in the environment at any time.

We demonstrated the approach in a ball bouncing task

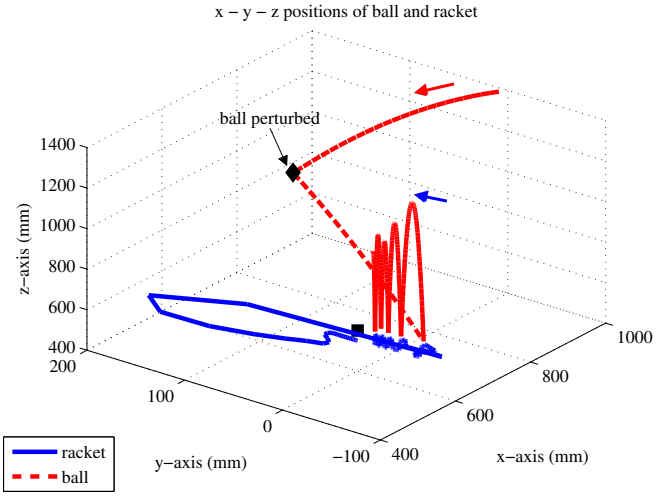
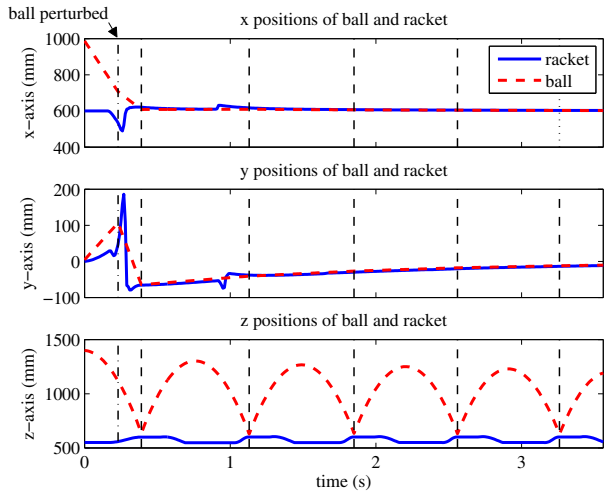
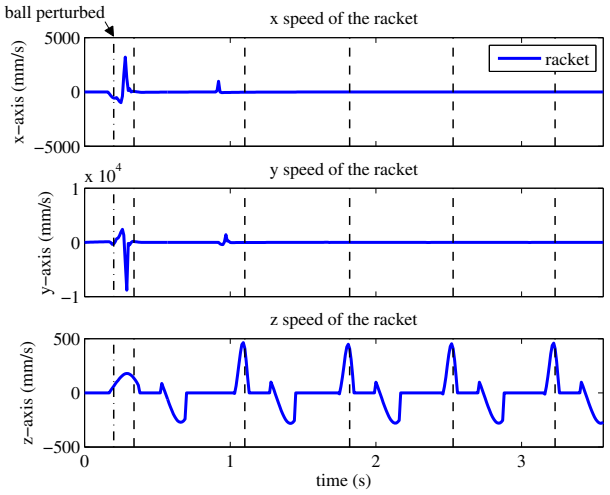


Fig. 7: Trajectories of the ball and the racket as the ball is perturbed during the first hitting movement.



(a)



(b)

Fig. 8: Trajectories of the racket and the ball over time for five consecutive hits after a perturbation of the ball. (a) Trajectories of the racket and ball over time, separated into individual movement components. (b) Speed profiles of the racket, showing an acceleration along x and y and a deceleration along z to compensate for the perturbation of the ball.

in which the robot moves a racket toward a predicted interception point and initiates a hitting motion of the racket just in time to hit the ball with maximal velocity. When the ball is perturbed, the time of initiation and the movement plan are updated to preserve the timing of the hitting motion relative to predicted time of impact.

Because the model is built from exactly solvable dynamic components, a Hopf normal form for the oscillator, and an Amari neural dynamics for behavioral organization, setting parameter values is easy. The neural dynamics is open to learning.

Acknowledgment

The authors acknowledge the financial support of the European Union Seventh Framework Programme FP7-ICT-2009-6 under Grant Agreement no. 270247—NeuralDynamics.

References

[1] W. Hong and J.-J. E. Slotine, "Experiments in hand-eye coordination using active vision," in *Experimental Robotics IV*, vol. 223. Springer, 1997, pp. 130–139.

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

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

[4] H. Li, H. Wu, L. Lou, K. Kühnlenz, and O. Ravn, "Ping-pong robotics with high-speed vision system," in *12th International Conference on Control Automation Robotics & Vision*, 2012.

[5] S. Schaal, A. Ijspeert, and A. Billard, "Computational approaches to motor learning by imitation," *Philosophical Transactions of the Royal Society (London) Series B*, vol. 358, pp. 537–547, 2003.

[6] S. Kim, E. Gribovskaya, and A. G. Billard, "Learning motion dynamics to catch a moving object," in *10th IEEE-RAS International Conference on Humanoid Robots*. IEEE Press, 2010, pp. 106–111.

[7] K. Mülling, J. Kober, and J. Peters, "A biomimetic approach to robot table tennis," in *International Conference on Intelligent Robots and Systems*. IEEE Press, 2010, pp. 1921–1926.

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

[9] S. Degallier, L. Righetti, S. Gay, and A. Ijspeert, "Toward simple control for complex, autonomous robotic applications: combining discrete and rhythmic motor primitives," *Autonomous Robots*, vol. 31, no. 2-3, pp. 155–181, May 2011.

[10] M. Buehler, D. E. Koditschek, and P. J. Kindlmann, "Planning and control of robotic juggling and catching tasks," *The International Journal of Robotics Research*, vol. 13, no. 2, pp. 101–118, 1994.

[11] A. Nakashima, Y. Sugiyama, and Y. Hayakawa, "Paddle juggling of one ball by robot manipulator with visual servo," in *Proceedings of the International Conference on Robotics and Automation*. IEEE Press, 2006.

[12] F. Oubbati, M. Richter, and G. Schöner, "Autonomous robot hitting task using dynamical system approach," in *IEEE Transactions on Systems, Man and Cybernetics (SMC)*, 2013, pp. 4042–4047.

[13] G. Schöner and J. A. S. Kelso, "Dynamic pattern generation in behavioral and neural systems," *Science*, vol. 239, pp. 1513–1520, 1988.

[14] W. H. Warren, "The dynamics of perception and action," *Psychological Review*, vol. 113, no. 1, pp. 358–389, 2006.

[15] S. K. U. Zibner, C. Faubel, I. Iossifidis, and G. Schöner, "Dynamic neural fields as building blocks of a cortex-inspired architecture for robotic scene representation," *IEEE Transactions on Autononous Mental Development*, vol. 3, no. 1, pp. 74–91, 2011.

[16] G. Schöner, "Dynamic theory of action-perception patterns: The time-before-contact paradigm," *Human Movement Science*, vol. 3, pp. 415–439, 1994.

[17] G. Schöner and C. Santos, "Control of movement time and sequential action through attractor dynamics: A simulation study demonstrating object interception and coordination," in *Proceedings of the 9th Intelligent Symposium On Intelligent Robotic Systems*, 2001, pp. 15–24.

[18] M. Tuma, I. Iossifidis, and G. Schöner, "Temporal stabilization of discrete movement in variable environments: an attractor dynamics approach," in *IEEE International Conference on Robotics and Automation*. IEEE Press, 2009, pp. 863–868.

[19] R. Ronsse, P. Lefevre, and R. Sepulchre, "Rhythmic feedback control of a blind planar juggler," *IEEE Transactions on Robotics*, vol. 23, no. 4, pp. 790–802, 2007.
[20] H. H. Rapp, "A ping-pong ball catching and juggling robot: a real-time framework for vision guided acting of an industrial robot arm," in *International Conference on Automation, Robotics and Applications (ICARA)*, 2011, pp. 430–435.
[21] R. Ronsse, K. Wei, and D. Sternad, "Optimal control of a hybrid rhythmic-discrete task: the bouncing ball revisited," *Journal of Neurophysiology*, vol. 103, no. 5, pp. 2482–2493, May 2010.
[22] Y. Sandamirskaya, M. Richter, and G. Schöner, "A neural-dynamic architecture for behavioral organization of an embodied agent," in *IEEE International Conference on Development and Learning (ICDL)*, 2011, pp. 1–7.
[23] M. Richter, Y. Sandamirskaya, and G. Schöner, "A robotic architecture for action selection and behavioral organization inspired by human cognition," in *IEEE/RSJ International Conference on Intelligent Robots and Systems*. IEEE Press, 2012, pp. 2457–2464.
[24] S.-i. Amari, "Dynamics of pattern formation in lateral-inhibition type neural fields," *Biological Cybernetics*, vol. 27, no. 2, pp. 77–87, 1977.
[25] G. Schöner, *Dynamical Systems Approaches to Cognition*. Cambridge, UK: Cambridge University Press, 2008, ch. Dynamical, pp. 101–126.
[26] U. Rokni and H. Sompolinsky, "How the brain generates movement," *Neural Computation*, vol. 24, no. 2, pp. 289–331, Feb. 2012.
[27] G. Schöner, "Timing, clocks and dynamical systems," *Brain and Cognition*, vol. 48, no. 1, pp. 31–51, 2002.
[28] L. Perko, *Differential Equations and Dynamical Systems*. New York, USA: Springer-Verlag, 2001.
[29] M. Jeannerod, "The timing of natural prehension movements," *Journal of Motor Behavior*, vol. 16, pp. 235–254, 1984.
[30] R. Bootsma and P. Van Wieringen, "Timing an attacking forehand drive in table tennis," *Journal of Experimental Psychology: Human Perception and Performance*, vol. 16, pp. 21–29, 1990.
[31] C. W. Wampler, "Manipulator inverse kinematic solutions based on vector formulations and damped least-squares methods," in *IEEE Transactions on Systems, Man and Cybernetics (SMC)*, 1986, pp. 93–101.

