Oubbati, F., Richter, M., Schöner, G. (2013). Autonomous robot hitting task using dynamical system approach.
IEEE International Conference on Systems, Man, and Cybernetics (SMC 2013), pp. 4042-4047.

Autonomous Robot Hitting Task Using Dynamical System Approach

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

***Abstract*—We propose a model that autonomously generates and flexibly organizes sequences of timed actions. The timing of the movements is controlled by non-linear oscillators. Their activation and deactivation is organized by a hierarchical neural-dynamic architecture. We demonstrate the features of our model in an exemplary robotic task where the manipulator arm keeps hitting a ball up an inclined plane. The autonomous generation of movement sequences is tightly coupled to visual sensory information about the ball motion and able to adapt, on-line, to perturbations introduced in the ball trajectory. The performance of the proposed model is evaluated and the reactions to different perturbations are discussed.**

***Index Terms*—timed motor acts, attractor dynamics, behavior sequences.**

I. Introduction

Humans exhibit natural skills in all kinds of ball games and racket sports where it is crucial to coordinate multiple actions in time and adapt the movements, on-line, to a quickly changing environment. For example, a typical table tennis player needs to produce in a fraction of time a hole sequence of actions that starts by tracking and predicting the ball trajectory, initiating 'at the right time' a timed movement to hit the ball while adapting, on the fly, the movement parameters to changing perceptual information until executing the hit, and finally returning to an *awaiting stage* to be ready for the next interception. These actions are continuously chained together and may be reproduced, at any time, in a new sensorial context with new movement parameters.

In fact, this was and remains a tremendous challenge for roboticist, because this kind of problems exemplify core elements of autonomous actions. Many robotic demonstration of ball batting, ball juggling, or robot table tennis often rely on fast and accurate algorithms that fail to adapt on-line to changing sensory information (see e.g., [1], [2]). Other approaches reproduce movements learned from human demonstrations (e.g., for catching a moving object [3]). Although able to adapt the movements to a changing perception are unable to generate distinct sequence of movements. Non-linear oscillators has been also used to generate periodic timed actions (e.g., [4]) that can be fully synchronized with sensed events but such systems generate, essentially, a single motor act in rhythmic fashion and so, limited with respect to the complexity of the timed actions. In previous robotic implementations, timed motor acts were generated from oscillators and stabilized against sensory input [5], [6].

In this paper, we show, in a simplified hitting task scenario, how a set of different timed actions can be organized in sequences and coordinated autonomously using a behavioral organization architecture that is sensitive to timing.

II. Robotic Scenario

An overview of our experimental setup for the ball hitting task is shown in Fig. 1. The hitting task involves the eight degrees of freedom (DoFs) robot arm CoRA and a colored rubber ball (of radius 3 cm and weight 66 g) rolling on an inclined plane placed in front of the robot. The robotic arm holds a small racket (10.5 cm in diameter) that is used to hit the ball. The robot is equipped with a vision system that tracks and predicts the ball trajectory. The hitting on the approaching ball occurs at a virtual hitting line "just in time" driving the ball back up the inclined plane.

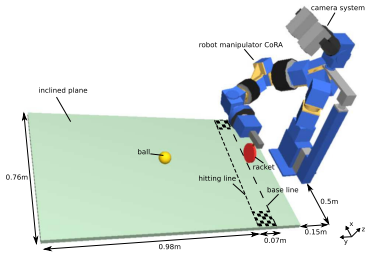


Fig. 1. Graphical overview of the experimental setup.

Ideally, the ball is hit continuously and kept in play or on the inclined plane at all times. The hitting region is constrained by safety limits set on both sides of the hitting line (marked with a checkerboard pattern in Fig. 1) to prevent the real robot from colliding the inclined plane left and right borders.

During task execution, different measures of the ball motion are continuously monitored or updated by the robot's vision system. These include a prediction of the ball landing position along the hitting line, which specify the ball hitting point (x_{hp}). In addition, the hitting movement is initiated only if the x_{hp} is tested to be inside the reachable hitting region of the robot by the parameter $b_{\text{reachable}} \in [-1, 1]$. The last measure expresses the time needed for the ball to reach the hitting line and refered here as the time-to-impact (t_{tim}). These parameters control the initiation of the hitting movement when the x_{hp} time is within a t_{tim} criterion and the ball is inside the robot's hitting region.

A. Task Movements Description

The task movements are designed to accommodate the hardware setup while respecting the manipulator's limitations

and workspace constraints. As the approaching ball becomes reachable ($b_{\text{reachable}} = 1$), a timed movement is initiated and brings the racket positions x_{eef} and y_{eef} to the predicted x_{hp}. When the ball t_{tim} reaches a time threshold, a second timed movement moves the racket azimuth orientation ϕ_{eef} and hit the ball.

Once the ball is hit, the racket ϕ_{eef} is brought to the initial orientation while y_{eef} position moves back to the base line. Simultaneously, a tracking movement is executed along the base line with the aim to be as close possible to the predicted x_{hp} for the next ball hitting. While in the tracking movement, the robot is ready to start another timed movements sequence to hit the ball whenever the vision system signals a predicted x_{hp} and t_{tim} satisfying the task conditions.

The timed movements sequence is interrupted if the ball falls out of the inclined plane after a miss, thus no more detected by the vision system, or if the ball is reflected before a hit occurs. Furthermore, these timed movements are sequenced autonomously (see Section III) allowing a flexible re-initiation able to accommodate any sudden change in the ball perceptual information.

B. Ball Tracking and Prediction

The robot's cameras module tracks the ball on the inclined plane by means of a color based tracking process (see Fig. 2). Based on the position measurements and on a discrete model of the ball motion, a linear Kalman filter is used to estimate the ball velocity and predict the ball motion. The hitting point x_{hp} is computed as the intersection between the ball heading vector and the hitting line while the time-to-impact t_{tim} is approximated using the motion model.

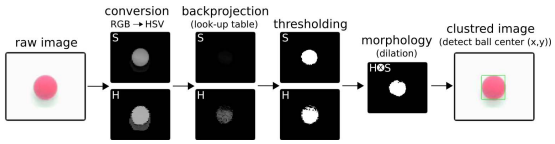


Fig. 2. Overview of the color-based segmentation process.

C. Inverse Kinematics

CoRA is composed of a series of *roll* and *pitch* joints (see Fig. 3). This special structure allows the use of a closed from solution for the inverse kinematics. Such a solution is always preferable for real time control of robots.

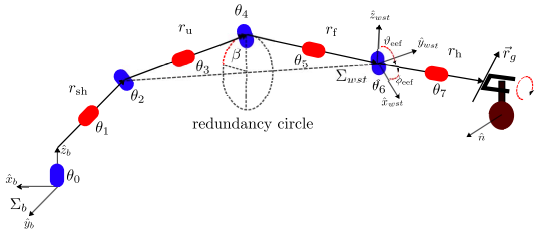


Fig. 3. CoRA arm configuration with the relevant coordinate systems.

To respect CoRA's work space constraints during hitting, the end-effector elevation ϑ_{eef} is set to $0°$ while the elbow posture, defined through the elbow angle β, is set to $102.5°$.

Given the desired racket positions x_{eef} and y_{eef}, the wrist vector $\vec{r}_{wst}$ can be computed and from which the joint angles $\theta_{i=0..4}$ are determined using a straight forward solution.

Having the robot's maximum speed limitation and to achieve higher speed in the timed hitting movements, the manipulator hand segment $\vec{r}_{\text{h}}$ is controlled alone to perform the actual hit so that only two joints, θ_5 and θ_6, are involved. The desired azimuth ϕ_{eef} controls separately the hand segment $\vec{r}_{\text{h}}$ orientation using the formula

$$\vec{r}_h = R_z^{\phi_{\text{eef}}} \cdot (\hat{e}_x l_{\text{h}}), \tag{1}$$

where l_{h} denotes the hand segment length, R_z is a rotation matrix around z-axis of a coordinate frame Σ_{wst} attached to the wrist, and $\hat{e}_x \in \mathbb{R}^3$ represents the x-axis unit vector. The orientation of the hand segment $\vec{r}_{\text{h}}$ permits to compute the joint angles θ_5 and θ_6.

To continuously ensure that the racket is correctly oriented during the hitting, the normal unit vector $\hat{n}$ must be kept always parallel to the inclined plane. The joint θ_7 is computed from the gripper vector $\vec{r}_g$ given by

$$\vec{r}_g = \vec{r}_h \times (R_x^{\alpha} \cdot \hat{e}_z), \tag{2}$$

where α defines the plane inclination, R_x is a rotation matrix about the world frame Σ_b x-axis, and $\hat{e}_z \in \mathbb{R}^3$ represents the z-axis unit vector.

III. Movements Trajectories Generation

To implement the task movements described in Section II-A, We propose a model able to generate autonomously sequences of timed movements.

A. Timed Movement Dynamics

To generate the trajectory of a timed movement, a dynamical system is used. The dynamical system for the pair of timing variables $(x\,,\,y)$ combines two regimes of operation controlled by the "neurons" $c_{\text{post}}, c_{\text{hopf}} \in [0,1]$ and permits the end-effector to start a movement from a postural state toward a target position within a desired movement duration. The time course of the variables x and y is governed by

$$\tau \begin{pmatrix} \dot{x} \\ \dot{y} \end{pmatrix} = -c_{\text{post}}\, a \begin{pmatrix} x - x_{\text{post}} \\ y \end{pmatrix} + c_{\text{hopf}}\, H(x,y) + \eta, \tag{3}$$

where the timing variable x defines the end-effector state at any time (here, the end-effector x_{eef}, y_{eef} positions and azimuth orientation ϕ_{eef}) and y is an auxiliary variable. These states are characterized by a time scale τ. In the postural regime ($c_{\text{post}} = 1$; $c_{\text{hopf}} = 0$), the timing variable x relaxes to the fixed point attractor x_{post} with a strength set by the term $a > 0$. Through the Hopf term $H(x,y)$, the oscillatory regime ($c_{\text{post}} = 0$; $c_{\text{hopf}} = 1$) stabilizes a periodic solution along a limit cycle attractor. The term $H(x,y)$ specifies the normal form of the Hopf bifurcation and is modified as follows

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

Where $r = \frac{1}{2}(x_{\text{target}} - x_{\text{init}})$ is the oscillator radius which defines the harmonic trajectory range of the timing variable x. In phase space, the Hopf cycle is shifted along the x-axis by r and the initial movement state x_{init}, so that the variable x smoothly rises from x_{init} to the movement target x_{target} during the oscillatory regime. The parameter $\lambda = r^2$ sets the oscillator radius and the oscillator intrinsic frequency ω defines the movement full cycle time $T = \frac{2\pi}{\omega}$.

The dynamics in Eq. 3 are augmented by a Gaussian white noise term, η, that guarantees escape from unstable states and assures robustness to the system.

B. Movement Time Adaptation

The target location x_{target} of a timed movement is usually a varying prediction based on low level and often noisy sensory information. Therefore, to ensure the correct timing, the movement must be parametrically updating while it is in execution. While x_{target} can be updated in Eq. 4, the cycle time T is adapted to accelerate or decelerate the movement accordingly so that the overall execution time is stabilized. Assuming a linear time to space relationship during the oscillatory regime, T is set to satisfy the ratio

$$\frac{T}{2d_{\text{init}}} = \frac{t_{\text{tim}}}{d(t)}. \tag{5}$$

where d_{init} is the initial distance to the target as the movement starts, and $d(t) = |x_{\text{target}} - x|$ is the remaining distance to the target.

C. Neural Dynamic Architecture for Behavioral Organization

In terms of movements generation, the timed movements described in in Section II-A represent timed behaviors. In order to initiate and terminate the different behaviors at appropriate instants in time, a form of behavioral organization is required. For that purpose, a neural dynamic architecture is built upon a framework for behavioral organization previously introduced in a grasping task [7]. The framework is based on Dynamic Field Theory (DFT) which is a variant of the attractor dynamics approach to embodied cognition [8].

Within DFT, behaviors are considered as *elementary behaviors* (EB) and modeled in a common way based on elements of DFT. An EB consists of two parts, *intention* and a *condition of satisfaction (CoS)* each of which represented by a dynamical node and a dynamical neural field (DNF) pair. The intention node, when active, models the intention to execute the EB. The intention field encodes the EB target location and permits the agent to execute the behavior. The CoS field receives inputs both from the intention field (representing the behavior target) and from the sensory system (describing the current state of the agent). If the two inputs overlap, a peak forms in the CoS field signaling the successful completion of the EB.

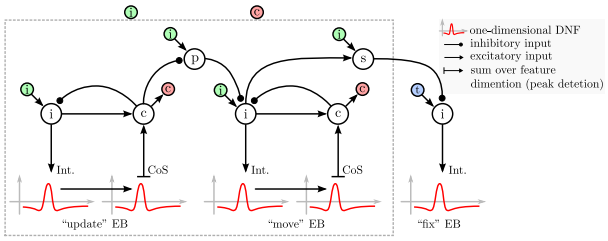


Fig. 4. A movement module is used to generate the timed movement behavior. The "update" and "move" EBs can be activated or deactivated through the colored intention 'i' and CoS 'c' nodes while the "fix" EB is always active by task input 't'. Behavioral constraints are set through a supression 's' and preconiditon 'p' nodes.

The peak activates the CoS node, which in turn inhibits the intention node, switching off the EB.

1) Movement Module: The two dynamic regimes described in Section III-A express two behaviors. (1) A movement behavior where the end-effector executes the timed trajectory during the oscillatory regime. (2) A fixation behavior that stabilizes the end-effector at a postural state after a movement. Furthermore, we need to update the initial state x_{init} of the movement before starting the behavior. The update process allows to start the behavior from the current real end-effector state x_{real} (read from the hardware sensory) and so, start the timed movement from any position along the movement dimension. The update can be performed by an update behavior.

These three EBs are integrated in a *movement module* (see Fig. 4). To generate a timed movement behavior, the movement module must be activated. First, the "update" EB updates the initial state of the movement through the intention node output (i.e., the sigmoided activation) $c_{\text{update}} = 1$ by the single attractor dynamical system

$$\dot{x}_{\text{init}} = -c_{\text{update}}\, a(x_{\text{init}} - x_{\text{real}}), \tag{6}$$

where $a > 0$ sets the attraction strength. The initial oscillator radius r_{init} in Eq. 5 is memorized similarly. Then, the "move" EB switches the timed movements dynamics in Eq. 3 from the postural to the oscillatory regime by the intention node output $c_{\text{hopf}} = 1$ (while $c_{\text{post}} = 0$) to generate the timed behavior. Moreover, the "move" EB intention field $u(x,t)$ encodes the movement target location x_{target} which can extracted by reading the peak position [9] using

$$\dot{x}_{\text{target}} = -\left(\int \sigma(u(x,t))dx\right) x_{\text{target}} + \int x_{\text{target}} \times \sigma(u(x,t))dx, \tag{7}$$

where $\sigma(.)$ is a sigmoid function. Finally, the "fix" EB intention node $c_{\text{post}} = 1$ (while $c_{\text{hopf}} = 0$) switches back the dynamics to the postural regime when the target state is reached or if the movement module is deactivated. Structurally, the "fix" EB does not have a CoS node and field since it caracterizes a postural state. To ensure the correct sequencing of the EBs, behavioral constraints modeled as dynamical nodes are set. A precondition constraint is set between the "update"

and "move" EBs so that a state update is always performed before a movement starts. Additionally, a suppression constraint prevents the "fix" EB from being active if the "move" EB is being executed.

The movement module together with the timed movement dynamics in Eq. 3 are designed to have a standard structure allowing a general use for the different components that define the end-effector state.

2) Sequential EBs: The dynamics of each movement component described in Section II-A is governed by a separate movement module. These timed movement behaviors are modeled by *sequential* EBs that should activate or deactivate the movement modules to initiate or terminate the timed behaviors (respectively). To achieve that, a hierarchical architecture is proposed where the higher level sequential EBs connect to and control the lower level movement modules as shown in Fig. 5. To implement the ball hitting task, four sequential EBs are defined

- "Move to ball" EB: brings the end-effector x_{eef} and y_{eef} positions toward the predicted hitting point x_{hp}.
- "Move hand forward" EB: executed by the end-effector azimuth orientation ϕ_{eef} and moves the robot hand segment to perform the hit.
- "Move hand backward" EB: brings ϕ_{eef} back to the initial orientation after a hit or a miss.
- "Move to base line" EB: moves the end-effector y_{eef} position back to the base line.

Note that the "Move to ball" and "Move to base line" EBs share the use of the y_{eef} movement module as is the case for the "Move hand forward" and "Move hand backward" EBs of the ϕ_{eef} movement module. A characteristic that illustrates the inherent flexibility of the architecture.

The sequential EBs are coordinated with the ball perceptual information and organized in time by setting behavioral constraints. A suppression constraint is set between the the "Move to ball" and "Move to base line" EBs so that the end-effector always moves to the base line if no ball prediction is flagged. Similarly, a suppression constraint between the "Move hand forward" and "Move hand backward" EBs ensures that the robot hand segment returns to the default orientation if the ball time is above a threshold. Moreover, using an additional suppression constraint, we ensure that the "Move to base line" EB is initiated only if the ball hitting is accomplished by the "Move hand forward" EB. Once at the base line with a retracted hand, the fixation behaviors get active and keep the robot arm at that configuration waiting for next ball hit. Due to our hardware speed limitation, the postural state of the x_{eef} movement module is made to track the ball along the base line so that the end-effector starts the "Move to ball" EB from the closest position when a new ball prediction is provided.

In the neural architecture, the robotic task is modeled as a dynamical task node. The task node stays always active during the execution of the robotic task and provides activation inputs to all intention nodes of the sequential EBs as well as the nodes implementing the behavior constraints. Furthermore, the task node activates by default the postural states in all movement modules to ensure the dynamics to be in the postural regime unless a movement is initiated.

D. Perception and Motor Systems

The proposed neural dynamic architecture is completely autonomous in the sense that all timed movement behaviors are initiated or terminated through on-line coupling to sensory information about the ball motion. The coupling is achieved by introducing perceptual constraints in the form of two precondition nodes that inhibit the intention nodes of the "Move to ball" and "Move hand forward" EBs. These two precondition nodes express perceptual conditions about the ball prediction and are active by default.

The first precondition node connects to a perceptual neural field that is defined over the inclined plane width dimension (i.e., its x-axis) and encodes the ball hitting point x_{hp} prediction. The perceptual field forms a peak activation centered at x_{hp} if that point lies inside the reachable hitting region of the robot (i.e., $b_{\text{reachable}} = 1$), see Fig. 1. The peak activation inhibits the precondition node which leads to start the "Move to ball" EB toward x_{hp}.

The second precondition node gets inhibited when the ball time-to-impact t_{tim} reaches a threshold leading to initiate the "Move hand forward" EB to hit the ball. The threshold time t_{thr} controls the initiation time of the "Move hand forward" EB and is variable to maximize the movement speed at the hitting point while respecting the hardware speed limitation. t_{thr} is computed using the formula

$$t_{\text{thr}} = \frac{T_{\text{hit}}}{2\pi} \arccos\left(\frac{2(\phi_{\text{des}} - \phi_{\text{init}})}{\zeta} + 1\right), \tag{8}$$

where T_{hit} is a fixed cycle time chosen to accommodate the robot hardware limitation in speed, $\zeta = 10^\circ$ is the movement range of the azimuth orientation ϕ_{eef}, $\phi_{\text{init}} = 95^\circ$ is the initial azimuth orientation, and ϕ_{des} is the desired azimuth orientation of the racket at the hitting point to direct the ball toward the top middle of the inclined plane after a hit (see Section III-E).

On the robotic arm side, the trajectories in Eq. 3 for the racket movement components (x_{eef}, y_{eef} and ϕ_{eef}) are generated in the inclined coordinate system and transformed to a world reference frame centered at the robot base. The corresponding joint angles are computed through the inverse kinematics transformation and fed to the robotic arm that executes autonomously the desired movements.

E. Stroke Movement

After every hit, the ball should be directed toward the inclined plane middle top with the aim to facilitate the next ball interception and achieve the 'keep the ball always in play' goal. To derive the required racket normal vector $\hat{n}$ so that the ball direction after a hit $\hat{v}_{\text{o}}$ is correctly oriented, we can use the reflection law

$$\hat{n} = \frac{\hat{v}_{\text{o}} - \hat{v}_{\text{i}}}{\sqrt{2 \cdot (1 - \hat{v}_{\text{i}}^T \cdot \hat{v}_{\text{o}})}}, \tag{9}$$

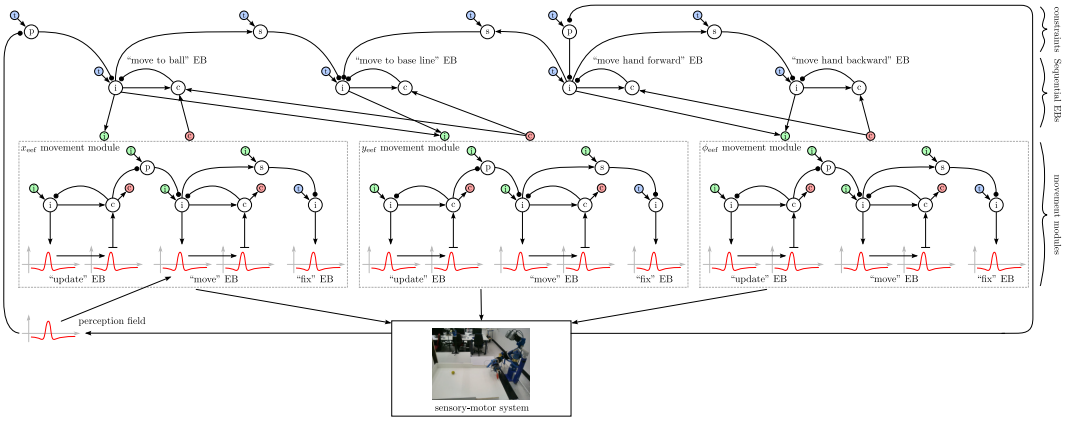


Fig. 5. The neural dynamic architecture for timed movements generation and behavioral organization.

where $\hat{v}_{\text{i}}$ is the predicted velocity of the ball at the hitting point. From $\hat{n} = (n_x\,,\, n_y)^T$ we can compute the desired racket orientation $\phi_{\text{des}} = \arctan n_y/n_x$ at the hitting point. To obtain the correct orientation ϕ_{des}, the cycle time of the "Move hand forward" EB is adpated using Eq. 5 with $d(t) = |\phi_{\text{des}} - \phi_{\text{eef}}|$.

IV. Evaluation & Results

In this section, we will illustrate the core properties of the proposed model both in a physically realistic Matlab simulation of the complete setup and in a hardware implementation.

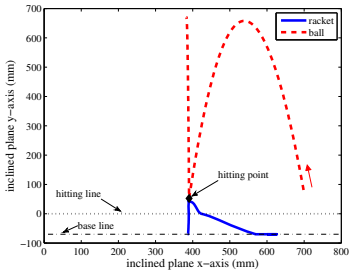


Fig. 6. Trajectories of the ball and the racket in a successful ball hit.

From simulation, Fig. 6 shows that the robot was able to hit the ball successfully. The detailed time courses of the relevant variables and parameters are shown in Fig. 7. When the ball becomes reachable for hitting at t ≈ 2.56 s (i.e., $b_{\text{reachable}}$ set to 1), the intention node of the 'move to ball (mtb)' EB turns on and drives the end-effector x_{eef} and y_{eef} positions toward the predicted x_{hp}. Then, as the ball approaches, the current time-to-impact becomes smaller then the variable threshold about t ≈ 3.95 s at which the intention node of the 'move hand forward (mhf)' EB gets activated and starts the end-effector ϕ_{eef} movement that hits the ball at t ≈ 4.42 s. Once the hitting occures, the ball is driven backup and becomes no more reachable (i.e., $b_{\text{reachable}}$ set to -1). Finally, the intention nodes of the 'move hand backward (mhb)' and 'move to base line (mtbl)' EBs switch on and drive the end-effector ϕ_{eef} and y_{eef} back to their initial postures.

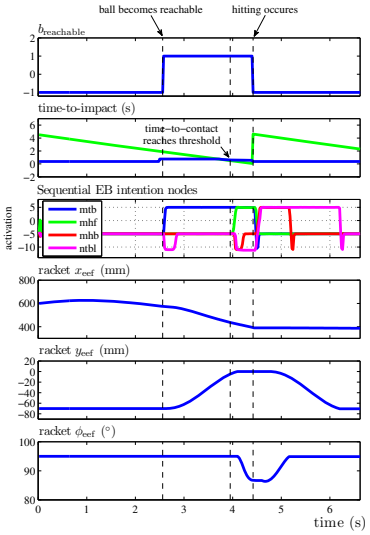
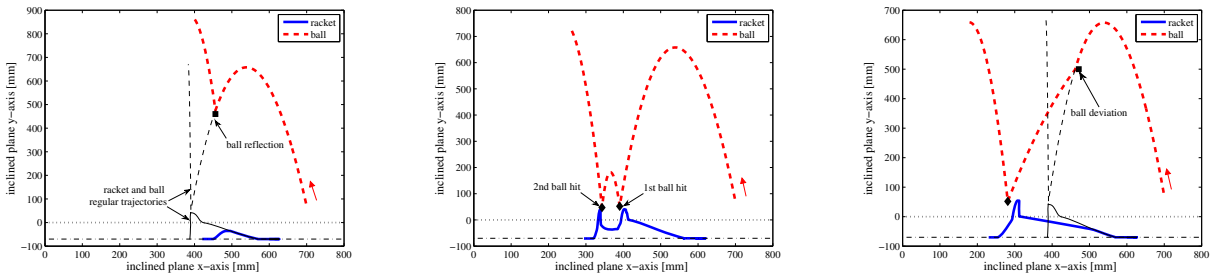


Fig. 7. Trajectories of the relevant parameters and variables for the autonomous successful ball hit and return to the base line. The top three plots represents ball reachability test parameter, time-to-impact and time courses of the sequential EB intention nodes. The bottom three plots illustrate the timed end-effector x_{eef}, y_{eef} and ϕ_{eef} trajectories.

In Fig. 8(a), we can see how the model allowed the robot to abort the hitting movements sequence when a ball reflection occurred during the robot movement. The robot was also able to re-initiate a supplementary hitting sequence when the ball became again reachable after a hit with a non-sufficient momentum as can be seen in Fig. 8(b). In the case the the ball is deviated as in Fig. 8(c), the robot was able to accelerate the movement and successfully hit the ball.

In the hardware implementation , the robot was able to keep the ball in play and perform several consecutive hitting sequences as shown in Fig. 9 and in TABLE I where some statistical data are gathered for 25 trials. Fig. 10 shows snapshots of the robot manipulator during a successful ball hit.



(a) (b) (c)

Fig. 8. Reactions of the robot to different kinds of perturbations during the racket movement.

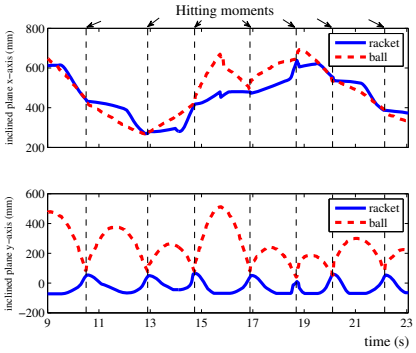


Fig. 9. Trajectories of the real robot and the ball for seven successive hits.

TABLE I
PERFORMANCE OF THE SIMULATED ROBOT ON SUCCESSIVE BALL.

Plane incl. (°)	Success rate (%)	Mean (Hits)	Average dev. ball⇆racket (mm)
2	94.32	18.9	28.2
4	93.56	14.52	35.11

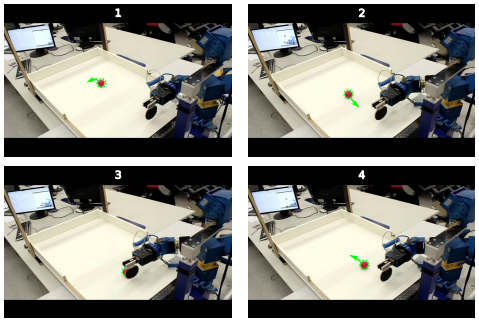


Fig. 10. Snapshots of the robot manipulator during a ball hit. (1) Tracking movement before initiating the hitting sequence. (2) Initiating the hitting movements sequence through the "Move to ball" EB. (3) Executing the hitting movement by the "Move hand forward" EB. (4) Returning to the base line by initiating the "Move hand backward" and "Move to base line" EBs.

V. CONCLUSION

We evaluated the core properties of the proposed model both in simulation and in a hardware implementation. The results illustrate how the system is able to react appropriately in response to external perturbations on the ball by initiating, aborting, or re-initiating inactive behaviors at any time. Furthermore, the system may also update movement parameters such as the amplitude and movement time on the fly when such perturbations shift the timing and spatial constraints. We are currently exploring the possibilities to extend the model to other robotic applications like catching or juggling and we will also transfer the implementation to a new robotic platform, the KUKA lightweight arm, which permits much faster movements and is, therefore, a more suitable platform for such applications.

ACKNOWLEDGMENT

The authors acknowledge the financial support of the European Union Seventh Framework Program FP7-ICT-2009-6 under Grant Agreement no. 270247—NeuralDynamics. This work reflects only the authors' views; the EC is not liable for any use that may be made of the information contained herein.

REFERENCES

[1] T. Senoo, A. Namiki, and M. Ishikawa, "Ball control in high-speed batting motion using hybrid trajectory generator," in *Proceedings of the International Conference on Robotics and Automation*, ICRA 2006, (Orlando, USA), pp. 1762–1767, IEEE Press, 2006.

[2] L. Hailing, W. Haiyan, L. Lei, K. Kuehnlenz, and O. Ravn, "Ping-pong robotics with high-speed vision system," Aug. 2012.

[3] S. Kim, E. Gribovskaya, and A. Billard, "Learning motion dynamics to catch a moving object," in *10th IEEE-RAS International Conference on Humanoid Robots*, (Nashville, TN, USA), IEEE Press, Dec. 2010.

[4] S. Schaal, D. Sternad, and C. G. Atkeson, "one-handed juggling: a dynamical approach to a rhythmic movement task," no. 2, pp. 165–183, 1996.

[5] G. Schöner and C. Santos, "Control of movement time and sequential action through attractor dynamics: A simulation study demonstrating object interception and coordination," *9th Intelligent Symp On Intelligent Robotic Systems*, 2001.

[6] M. Tuma, I. Iossifidis, and G. Schöner, "Temporal stabilization of discrete movement in variable environments: an attractor dynamics approach," vol. 1, pp. 863–868, 2009.

[7] 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 IROS*, IEEE Press, 2012.

[8] G. Schöner, *Dynamical Systems Approaches to Cognition*, ch. Dynamical, pp. 101–126. Cambridge, UK: Cambridge University Press, 2008.

[9] 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.