Autonomous generation and on-line updating of sequences of timed robotic actions: an attractor dynamics approach

Dissertation
zur Erlangung des Grades eines
Doktor-Ingenieurs
der Fakultät für
Elektrotechnik und Informationstechnik
an der Ruhr-Universität Bochum

Vorgelegt von

Farid Oubbati

aus Algerien

Bochum, 2014

Date of Submission: April 29^{th}, 2014

Date of Examination: July 1^{st}, 2014

Gutachter: Prof. Dr. Gregor Schöner

Zweitgutachter: PD. Dr. Rolf Würtz

Acknowledgements

I would like to express my gratitude to my advisor Prof. Dr. Gregor Schöner for his guidance and support during this project. I would also like to thank PD Dr. Rolf P. Würtz for supporting this thesis as a joint examiner. Thanks go also to my colleagues of the Autonomous Robotics Group, with whom I enjoyed to work.

I am very grateful to the German Academic Exchange Service (DAAD) for funding my research. During four years, I have enjoyed my scholarship. . . Thanks a lot.

Bochum, April 21^{th}, 2014

To my family and friends.

Abstract

In everyday life, humans are very skillful in generating complex action sequences that are timed in coordination with a changing environment and highly adaptive to the current perceptual context. This ability appears clearly in tasks ranging from making coffee, driving a car, to playing table tennis or airhockey. Understanding how these behavioral patterns are generated and providing solutions to emulate this ability is a key challenge in autonomous robotics.

In this thesis, I address how different movement behaviors may be timed to sensory events while being sequentially and flexibly organized. In this context, I propose a dynamical model for sequence generation of timed movements. The system is formulated within the dynamical systems approach, using its tools and concepts that account for timing and coordination in human movements. The model consists of two parts: a hierarchical neural dynamics architecture for behavioral organization and timing dynamics for movements generation. The neural dynamics architecture is based on elements from the dynamic field theory (DFT) which is a variant of the attractor dynamics approach to cognition. The timed movements are generated using a framework that combines fixed point attractors to define postural states and stable limit cycles for timed behaviors. This formulation ensures stability and robustness of the system while flexibility is provided through bifurcation properties of the dynamics. Both the generation and organization of movements are tightly coupled to time-varying sensory information and autonomously react to perturbation either by updating movement parameters or by flexibly adapting the sequence of behaviors.

The core properties of the model are assessed in simulation by two robotic tasks, catching and hitting a ball, and in a hardware implementation of a robotic hitting task. Both from simulation and hardware tests, the obtained results demonstrate that the system is able to execute successfully the robotic tasks and illustrate the features of the model in handling different perturbation scenarios. In addition, the tests show the characteristics of the dynamical systems approach in addressing this kind of problems in robotics.

Keywords: Attractor dynamics, dynamic field theory (DFT), autonomous robotics, timed movement, behavioral organization.

Zusammenfassung

Im alltäglichen Leben sind Menschen sehr geschickt darin, komplexe Handlungssequenzen zu generieren, die mit einer veränderlichen Umwelt zeitlich koordiniert und hochgradig an den aktuellen Sensorkontext angepasst sind. Diese Fähigkeit wird im Spektrum verschiedenster Aufgaben offensichtlich, vom Zubereiten von Kaffee über das Steuern von Automobilen, bis hin zum Tischtennisspielen. Zu verstehen, wie diese Verhaltensmuster generiert und emuliert werden können, ist eine zentrale Fragestellung im Bereich der autonomen Robotik.

In dieser Arbeit behandle ich, wie verschiedene Bewegungsverhalten zeitlich mit sensorischen Ereignissen koordiniert und gleichzeitig sequentiell und flexibel organisiert werden können. In diesem Kontext schlage ich ein dynamisches Modell für Sequenzgenerierung und zeitlich koordinierte Bewegungen vor. Für die Formulierung dieses Systems nutze ich dabei den dynamische-Systeme-Ansatz und die daraus stammenden Konzepte und Werkzeuge, um den zeitlichen Ablauf und die Koordination menschlicher Bewegungen abzubilden. Das Modell besteht aus einer hierarchischen, neuro-dynamischen Architektur für Verhaltensorganisation, sowie Timing-Dynamiken für die Bewegungsgenerierung. Der neurodynamische Teil der Architektur basiert auf Elementen aus der dynamische-Felder-Theorie (DFT), einer Variante des Attraktordynamik-Ansatzes für die Modellierung Kognition. Die zeitlich koordinierten Bewegungen werden von einem Framework generiert, das Fixpunkt-Attraktoren kombiniert, um posturale Zustände zu definieren, und stabile Grenzzyklen für zeitlich koordinierte Verhalten erzeugt. Diese Formulierung stellt die Stabilität und Robustheit des Systems sicher, während Flexibilität durch die Bifurkationseigenschaften der Dynamiken erreicht wird. Sowohl die Generierung als auch die Organisation der Bewegungen sind eng an die zeitlich veränderlichen Sensorinformationen gekoppelt und reagieren autonom auf Störungen, entweder, indem sie Bewegungsparameter aktualisieren, oder, indem sie die Verhaltenssequenz flexibel anpassen.

Die Kerneigenschaften des Modells werden sowohl in Simulationen zweier robotischer Anwendungen, dem Fangen und Schlagen eines Balls, als auch in einer Hardwareimplementierung des Letzteren evaluiert. Sowohl die Ergebnisse der Simulationen als auch der Tests auf der echten Hardware zeigen, dass das System in der Lage ist, die Aufgaben erfolgreich auszuführen. Weiterhin illustrieren die Ergebnisse die Eigenschaften des Systems in verschiedenen Perturbationsszenarien, und heben die Nützlichkeit des dynamische-Systeme-Ansatzes in robotischen Problemstellungen hervor.

Schlagwörter: Attraktordynamik, dynamische-Feld-Theorie (DFT), autonome Robotik, zeitlich koordiniertes Bewegungsverhalten, Verhaltensorganisation.

Contents

List of Figures

List of Tables

Chapter 1

Introduction

1.1 Preamble

In everyday life, humans and other animals generate behavioral patterns that are tightly coordinated with the environment, in the service of achieving a specific goal. On one hand, this ability implies the organization in time of different elements of the body to produce the required pattern of movements. On the other, it implicates perception, such that information about the surroundings lead to the selection of the appropriate action to be performed and adapted to the environmental conditions.

For instance, humans excel in all kinds of ball games and racket sports where it is crucial to coordinate multiple actions in time and adapt the movements online to a quickly changing environment. In a table tennis game, a player is able to generate and coordinate, almost instantaneously, a set of different tasks that start by tracking and predicting the ball trajectory, initiating the required movements to hit the ball, and returning to a resting posture to be ready for the next set of actions. In addition, the generated movements should adapt and continuously compensate for perturbations in order to successfully perform the hit. Furthermore, a hitting movement is generally constrained with a correct timing and racket orientation at impact. Typically, these actions are continuously chained together and may be reproduced at any time in a new sensorial context with new movement parameters.

In performing such tasks, humans exhibit two complementary attributes: stability and flexibility. Stability implies consistent movement patterns that resist perturbations and are reproducible in different sensorial contexts. In addition, these movement patterns are non-rigid but flexible and adaptive to environmental constraints like timing and coordination. Understanding how such behavioral patterns are generated and organized in time has been a theme for decades, in particular, defining the nature of the interaction between perception and the generated actions. Different research disciplines including psychology, cognitive science, neuroscience, and robotics addressed these issues using different concepts and tools. Those studies were conducted at many levels of analysis and resulted in different theories of motor control.

An effective approache to human movement is the dynamical systems approach (Kelso, 1995; Schöner and Kelso, 1988). If we emphasize the fact that these behavioral patterns result from cognitive processes that are environmen-

tally embedded, corporeally embodied, and neurally embrained, the dynamical systems approach offers an integrated framework to describe cognition as a dynamical phenomenon in a dynamical world (Van Gelder et al., 1999; Thelen and Smith, 1994). The central insight of dynamical systems theory is that behaviors are represented as stable states of a dynamical system that unfolds continuously in time. Moreover, such a representation provides the system with the necessary flexibility through instabilities (or bifurcations).

In this thesis and within the dynamical systems framework, I propose a dynamical model for sequence generation of timed movements. The model presents a systematic approach to the organization of timed actions in which both the individual movement components as well as the sequencing mechanism are tightly coupled to changing sensory information. The performance of the model is assessed both in simulation and in a hardware implementation of a ball hitting task executed by a robot manipulator.

1.2 Motivations for the thesis

From coffee making to playing pinball machine or table tennis, humans excel at generating complex action sequences that are often coordinated with the environment. Moreover, these actions need to be timed, flexible, and adaptive to the current perceptual conditions. Being able to define the processes behind these behavioral patterns is a key challenge both in cognitive science and in autonomous robotics. Indeed, enabling robots to exhibit resembling behaviors will open the doors to more direct human-robot interactions and will considerably increase robots autonomy in natural environments.

In the present thesis, I address how different movement behaviors may be timed to sensory events while being sequentially and flexibly organized. The main contribution of the work is a dynamical model to generate sequences of timed movements. The mechanism comprises a hierarchical neural dynamics architecture for behavioral organization and timing dynamics for movements generation. Both organization and generation of the movements are coupled online to sensory information so that the system adjusts to perturbations both by varying movement parameters, and by activating or deactivating different movement components.

For this work, different tools and concepts from the dynamical systems theory are used. The dynamical systems approach is a theoretical framework within which the embodied view of cognition can be formalized and designed. The approach defines the quantitative variables that describe a system as attractor states of continuous time dynamical systems. This formulation provides dynamical models with an intrinsic stability and robustness against perturbations and flexibility features through bifurcations (Van Gelder et al., 1999; Schöner, 2008). With these properties, the framework can account for many empirical evidences in human movements, notably, movement timing, coordination, and the neural and sensory couplings involved.

To demonstrate the core properties of the model, two robotic applications are considered. A simplified catching task in simulation permits to show the functioning and properties of the proposed architecture. An extension of the

model to a more complex application with a hardware implementation is also addressed. Here, the task is a blend of playing pinball machine and airhockey, in which a robot arm keeps hitting a ball back up an inclined plane. In both applications, perturbation scenarios that affect the movement timing and the behavioral organization are treated. Even though my perspective is a pure engineering one, these demonstrations represent a proof of the applicability of the dynamical systems concepts in these kind of tasks. Tasks that imply high speed movements and a large degree of adaptation and flexibility.

1.3 Main goals of the work

The main goals of the present work can be summarized as follows:

1. Inside a framework for timed movement generation, develop mechanisms to stabilize total movement time by adapting the movement parameters in face of perturbations.

2. Design and implement an architecture to organize in time and space timed behaviors in coordination with sensed events. The architecture is based on elements from the dynamic field theory (DFT) (Schöner, 2008).

3. Illustrate the core features of the proposed model in simulated robotic tasks.

4. Evaluate the performance of the model in a hardware implementation.

1.4 Contributions

A preliminary implementation of the robotic hitting task has been published as a refereed contribution in (Oubbati and Schöner, 2013). The application of the dynamical model for sequence generation of timed movements to a robotic hitting task was published as an abstract (Oubbati et al., 2013b), and as a refereed proceedings paper (Oubbati et al., 2013a).

1.5 Outline of the thesis

The remainder of the dissertation is laid out in the following chapters. Chapter 2 provides a review of the related literature and an introduction to the principles and tools of the dynamical systems approach. Chapter 3 presents the dynamical model for timed movements sequence generation and illustrates its features in a simulated catching task. Chapter 4 applies the proposed model to a robotic hitting task both in simulation and in a hardware implementation. Chapter 5 covers my conclusions and further work that would improve on the model.

Chapter 2

Background and methods

2.1 Sequencing timed motor acts: state of the art

Human life is full of activities that require generating a well ordered series of complex actions, that are yet stable and highly adaptive. Moreover, these actions often need to be timed and coordinated with the environment. These aspects appear clearly in many sport activities like table tennis, pinball or airhockey games where humans can exhibit their skills in terms of speed, dexterity, and coordination while keeping very flexible against perturbations.

Building movement generation systems able to emulate these capacities was and is still a tremendous challenge in the robotics field. Roboticists studied problems of this kind because they exemplify core elements of autonomous action, in particular, the problem of timing robot's actions to external events. Providing such skills to robots will certainly increase their performance and abilities to work autonomously in human environments.

To solve these problems, several solutions have been proposed and demonstrated in a variety of robotic scenarios. Typically, tasks like ball hitting, juggling, or catching in which robots interact with moving objects are the most widely chosen to address these issues. In the following, I will go through some literature where similar problems are addressed using different approaches. Broadly speaking, these approaches can be classified into classical (or algorithmic approaches), learning based approaches , and dynamical systems approaches.

2.1.1 Classical approaches

Tasks like ball catching, juggling, or hitting have been extensively addressed by various researchers. For instance, robot juggling is a well studied problem and has been realized using different approaches. In earlier implementations, Aboaf et al. (1989) presented a task level learning method to juggle a single ball in three dimensions with an aiming mechanism. A class of mirror control algorithms introduced by Buehler et al. (1994) allowed a one degree of freedom robot to juggle a ball in two-dimensional space, where a symmetry between the ball and paddle motions is continuously kept. More recently, Nakashima et al. (2006) proposed an extended version of the Buehler et al. (1994) algorithm to perform a juggling task by solving the ball regulation problems. The ball was

tracked by a stereo vision system and predicted using a discrete motion model. With an industrial robot, Rapp (2011) implemented a Ping-Pong ball juggling system using a simple PID controller. A Kalman estimator and a discrete time model were used to predict the ball trajectory. The juggling task was even demonstrated using a Quadrocopter (Mueller et al., 2011) with a near-hover feedback controller (Bouabdallah et al., 2004).

A robot catching flying objects was also treated. Baetz et al. (2009) presented a non-prehensile catching method for a basketball on a plate hold by an industrial robot. The method decomposed the task into a contact selection problem, based on mechanical joint limits and a dynamical manipulability measure (Yoshikawa, 1985), and a balancing problem, based on a force/torque feedback control. The ball was predicted using recursive least squares. Using a theme park type animatronic humanoid robot, Kober et al. (2012) developed a test bed for a throwing and catching game using a finite-state machine algorithm.

Ball interception and hitting scenarios were also frequently considered. In a Ping-Pong robotic application, Hailing et al. (2012) used a high speed stereo vision system to track and predict the ball trajectory, and a PID controller for a redundant manipulator to perform the hitting tasks. In the same context, Senoo et al. (2006) proposed a hybrid trajectory generator in joint space for high-speed batting motions. They split the control of the robot's joints so as to control separately for high-speed swinging while the remaining degrees of freedom are used for hitting.

Similar tasks were also addressed as optimal control problems. Optimal control is a set of techniques for determining the control signals for a system with given dynamics that will minimize an objective or cost function while satisfying specified constraints (Kirk, 1970). Lampariello et al. (2011) formulated a ball caching task as a non-linear optimization problem with the mechanical energy of the manipulator as a cost function. The method was only tested in simulation with a suitably initialized local planner and without minimizing the cost function. A biologically inspired trajectory generator is proposed by Muelling et al. (2010) for a robot table tennis application. Inspired from the movement stages observed in human players, comfort postures are selected in joint space and cost functions are defined on these postures. The model produced a single stroke movement at a time on a ball predicted using a Kalman filter and a dynamical model.

Generally, these kinds of applications are used to probe real-time control problems in robotics. Even though fast and accurate, the path planning methods are most of the time based on heuristics and ad-hoc approaches that severely suffer from the lack of flexibility against environmental variations. For example, perturbation cases in the ball predictions were rarely considered and reorganizing actions in reaction to some changing events was not a topic. Moreover, timing was always seen from the prism of real-time planning and not as a result of coordination with external objects.

2.1.2 Learning based approaches

Learning from external agents tasks like ball catching, juggling or table tennis was also explored. Learning control means improving a motor skill by repeatedly

practicing a task. In imitation learning, robots are taught to perform a task by observing a set of demonstrations provided by a teacher (human or robot). Demonstrations to a robot may be performed in different ways; back-driving the robot, teleoperating it using motion sensors, or capturing a task via vision sensors. The learning process consists of extracting the relevant information from the demonstrations and encoding this information in a motion model that can be used to reproduce the task.

Different approaches and techniques have been applied in a variety of robotic tasks. To teach a robot how to juggle a devil stick, Schaal and Atkeson (1994) proposed a memory-based model by means of a locally weighted regression (Atkeson et al., 1997). The particular formulation of the approach allowed a real time implementation using training data from an inverse model. Using a similar method, (Matsushima et al., 2005) demonstrated a robot table tennis task by learning three maps representing the ball states before and after a hit as well as the desired ball state in the opponent court. These data were fed to a polynomial based trajectory generator for the robot effector. Neither real time coordination with the ball nor perturbations were treated.

Ijspeert et al. (2002) and Schaal et al. (2003) suggested to use dynamical movement primitives to represent both discrete and rhythmic movements. The framework allowed the representation of arbitrarily shaped movements through the primitive's policy parameters that are estimated by locally weighted regression on training data. The framework was demonstrated in several tasks like tennis swings. However, the dynamics are time-dependent, thus very sensitive to temporal perturbations, and modifying the learned global shape of the movement in response to perturbations or varying target predictions was not trivial. In the same context, Kober et al. (2010) proposed a modified version of the canonical form introduced by Ijspeert et al. (2002) to allow an online updating of the movement's target and final speed. The new version was demonstrated in a robot table tennis application. The evaluation showed that the dynamics could be generalized to different targets. Nevertheless, the stroke movements were often not precise which shows that the robot could not sufficiently adapt to varying ball predictions. Moreover, perturbation scenarios during movements were not addressed at all.

In a robot catching task, Kim et al. (2010) considered movement timing and coordination with a moving object. The demonstrated trajectories using a data glove were encoded in an autonomous dynamical system with Gaussian Mixture Models and Expectation-Maximization algorithm (Gribovskaya and Billard, 2009). The motion dynamics were tuned using a timing controller. The robot could adapt to changes in ball predictions while keeping the temporal structure of the movement stable. However, the adaptation was too slow to use real flying or rolling objects. Moreover, adapting the task autonomously so as to abort or restart the catch in response to some sensed events was not considered. Using a similar approach, Khansari-Zadeh et al. (2012) extended the previous formulation of the autonomous dynamical system to model robot motions with a desired velocity at the target. The modified version was demonstrated in robot experiments of playing minigolf. The approach could be generalized to different initial positions of the ball and the hole. In addition, online adaptation to perturbations in the ball or the hole positions was demonstrated. However, an

upper bound for the maximum value of perturbations has to be set and timing the movements was not considered. The same approach have been used in many other applications like a pick-and-place task (Gribovskaya and Billard, 2009) or a coordinated hand-arm grasping (Shukla and Billard, 2011).

As opposed to classical planning algorithms, learning approaches brought to robots skillful manipulation capacities similar in flexibility and precision to that displayed by humans. In addition, the proposed approaches were able to adapt to variations in the sensory information and timing could be considered as a constraint. Nevertheless, the adaptation often fails in the face of large perturbations particularly when high speed movements are to be produced. Moreover, learning a sequence of movements that can be (re)organized in coordination with some perceptual events does not seem to be trivial.

2.1.3 Dynamical systems approaches

The dynamical systems theory was also employed to address similar tasks. This use was mainly motivated by the proven properties of the approach in endowing robots with flexibility in movements generation while ensuring stability and robustness against perturbations.

In one of the first attempts, Schöner and Santos (2001) proposed a two-layer model that can produce both stable oscillations and stationary states. The top layer was composed of switching dynamics (see Section 2.2.2 for more details) that, in coupling to sensory information, control the switching between these two regimes. The lower layer was responsible to generate the stationary states through fixed point attractors and timed movements via a stable limit cycle. The model allowed the implementation of a sensor-driven initiation and termination mechanism of discrete timed movements. A demonstration in a simulated ball interception task was proposed. The model could produce sequences of timed movements that were coordinated and flexibly organized according to sensed events. However, stabilizing the movement time by updating the parameters of the dynamics so that to accelerate or decelerate the movement in response to perturbations was not treated. More recently, Santos and Ferreira (2009) used an identical framework to implement a catching task executed by a Puma arm on a ball rolling on an inclined table. Here also, integrating a consistent mechanism to stabilize movement time in face of noisy sensory information was not a topic. The proposed model represents the basic structure of the timed movement dynamics presented in Section 3.1.

In the context of movement timing and using a similar approach, Tuma et al. (2009) formulated an on-line adaptation rule to stabilize total movement time against disturbances. The rule was demonstrated on a mobile robot that while reaching for a target should avoid obstacles. In this formulation, heading dynamics controlled the movement direction and the oscillator dynamics generated velocity profiles. The adaptation rule permitted to continuously update the oscillator radius that represents the current speed of the robot. The system adapted the robot's speed so that to accelerate or decelerate if the distance to the target is increased or decreased (respectively) during, for example, obstacles avoidance. Nevertheless, generating sequences of several timed movements, like going back to a homebase after reaching a target and back again, was not considered.

In a different context, Degallier et al. (2011) presented a dynamical framework able to generate combinations of discrete and rhythmic motor primitives. The approach is based on the concept of central pattern generators (CPGs) observed in invertebrate and vertebrate animals (Ijspeert, 2008). The system was applied to interactive drumming and infant crawling in a humanoid robot. The framework modeled stable oscillatory movements around time-varying offsets for each degree of freedom but could be tuned to generate discrete or rhythmic movements separately. The proposed CPG system is a network of coupled dynamics to obtain discrete and rhythmic behaviors that are coordinated by ensuring fixed time relationships. Different mechanisms were defined to handle both spacial and temporal perturbations. Timing was introduced as a coordination constraint, notably, to stabilize phase relationships between movements while playing music notes in the drumming task. The generated patterns were a set of repetitive rhythmic movements combined with shift cycles between the drum's pads. Having a stable temporal structure for the generated patterns or a mean to reorganize the behaviors so as to abort and restart the task in accordance with some sensed information were not treated.

The dynamical systems approach has been also used in several applications where a well ordered sequence of movements need to be produced. In a simulated grasping task, Luksch et al. (2012) presented a neural architecture for movement generation that is based on a hierarchical organization of dynamical systems. Different properties like failure recovery and adaptation of the behavior sequence in response to sensorial information or a prediction layer were demonstrated. In the same context, Richter et al. (2012) introduced a neural-dynamic framework for behavioral organization, in which the action selection mechanism is tightly coupled to the agents sensory-motor systems. The architecture is composed of several elementary behaviors organized into a sequence using behavioral constraints and online perceptual information. The viability of the approach was shown in a grasping task for a humanoid robot. In both systems, timing the behaviors and their organization in coordination with sensed events was not addressed.

As can be observed, the problems of sequencing motor behaviors and considering temporal constraints in the movement generation in presence of perturbations are most of the time addressed separately. Moreover, developing a mechanism able to reorganize the sequence of behaviors in coordination with perceived information is not a trivial task. The work presented in this thesis addresses these specific problems. Using the dynamical systems approach, I have built a model to generate sequences of timed movements. Both the individual movement components and the behavioral organization are coordinated with perceptual information and adapt to perturbations. The model's core properties are assessed in simulation and in a hardware implementation of a robotic ball hitting task.

2.2 Dynamical systems theory

The dynamical systems theory is a confederation of research efforts bound together by the idea that natural cognition is a dynamical phenomenon that can

be best understood in dynamical terms, and guided by the belief that dynamics provide the right tools for understanding cognitive processes (Thelen and Smith, 1994; Van Gelder, 1998). The central insight of dynamical systems theory is that a behavior can be understood geometrically, that is, as a matter of position and change of position in a space of possible overall states of the system that unfolds continuously in time (Van Gelder, 1998). The behavior can then be described in terms of attractors, transients, stability, hysteresis, coupling, bifurcations, and so forth features found in dynamical systems (Schöner, 2008; Van Gelder et al., 1999).

In this section, I will review some core concepts of the dynamical systems theory. In particular, I will focus on the different methods used for behavior generation and organization as an approach to autonomous robotics. Some of these tools will be used along this thesis.

2.2.1 Generation of behaviors

The dynamical system approach to autonomous robotics has been first introduced by Schöner et al. (1995) where they have proposed to represent the task relevant behaviors as attractor state variables of dynamical systems. Moreover, they have illustrated stability as a central characteristic of these concepts and demonstrated flexibility through a form of instabilities (or bifurcations) in the behavioral dynamics of the robot.

The qualitative changes in a state variable, called non-equilibrium phase transitions, express the diversity of the behavioral patterns that can be generated. These transitions occur in the control parameters space. The control parameters are collective variables that describe the dynamics of the system. The emergent patterns at a bifurcation are said to be self-organized because the control parameter that induces the changes does not contain information about the new pattern. The emerging behavioral pattern is only caused by specific interactions between the numerous subsystems that are involved in the behavior generation (Schöner and Kelso, 1988).

The approach has been applied to implement various navigation tasks for autonomous vehicles (Bicho et al., 1998; Monteiro and Bicho, 2002; Soares et al., 2007). A typical issue for navigation is planning a collision-free path during a target acquisition task. To describe the behaviors that this movement represents, the heading direction, ϕ, and velocity, v, of the robot are generally employed as the behavioral variables. The actual behaviors are generated in time as the governing solutions of dynamical systems (or behavioral dynamics) that define the rate of change of the behavioral variables as a function of their current values

$$\begin{aligned} \dot{\phi} &= \frac{d\phi}{dt} = f(\phi), \\ \dot{v} &= \frac{dv}{dt} = f(v), \end{aligned} \tag{2.1}$$

with $\phi(t)$ and $v(t)$ specifying the on-going behaviors at any time t. The behavior's dynamics are designed to be dissipative with asymptotically stable fixed points (or attractors). The attractors in Eq. 2.1 serve to define the behavioral variables so that the system is ensured to be at all times in a stable state.

For example, if we assume the robot's velocity to be constant ($\dot{v} = 0$), a simple behavioral dynamics for target acquisition can be defined by

$$\dot{\phi} = -\lambda_{\text{tar}} \sin(\phi - \Psi_{\text{tar}}), \tag{2.2}$$

where $\lambda_{\text{tar}} > 0$ sets the attraction strength of the robot's heading variable, ϕ, toward the target direction specified by the attractor Ψ_{tar} in world coordinates. Given that the target direction may vary as the robot moves, the dynamics of Eq. 2.2 ensures that the heading variable, ϕ, sits always on or close to the attractor.

In the case where an obstacle is present in the robot's way to the target and if we assume that its distance from the robot d_{obs} can be continuously estimated using sensory inputs, we can modify the behavioral dynamics by adding an unstable fixed point (also called repellor) at the obstacle direction Ψ_{obs} as follows

$$\dot{\phi} = -\lambda_{\text{tar}} \sin(\phi - \Psi_{\text{tar}}) + \lambda_{\text{obs}}(\phi - \Psi_{\text{obs}}) \exp\left(\frac{(\phi - \Psi_{\text{obs}})^2}{\sigma}\right), \tag{2.3}$$

where $\lambda_{\text{obs}} > 0$ sets the repulsion strength and $\sigma > 0$ defines the angular range of the repellor. The functional form of the repulsive term is biased by the distance to the obstacle through $\lambda_{\text{obs}} = f(d_{\text{obs}})$ and $\sigma = g(d_{\text{obs}})$. This implies that the strength and angular range of the repulsion becomes stronger as the robot gets closer to obstacles (Figure 2.1).

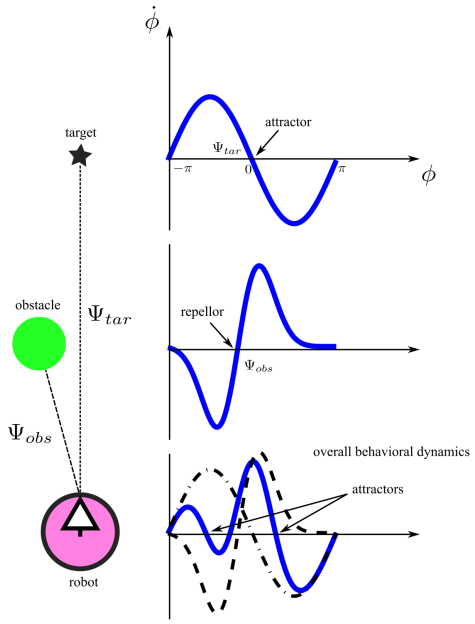


Figure 2.1: Behavioral dynamics for mobile robot navigation. Overall phase portrait of the heading dynamics (right) results from a target and obstacle contributions for a robot navigation task (left). The behavioral dynamics illustrate the emergence of two attractors. This implies an internal decision to be made by the system that depends on the current state of the heading variable ϕ.

When multiple N obstacles are present in the scene, their repulsive contributions can be simply summed and added to Eq. 2.2 to form the overall behavioral

dynamics

$$\dot{\phi} = -\lambda_{\text{tar}} \sin(\phi - \Psi_{\text{tar}}) + \sum_{i=1}^{N} \lambda_{\text{i}}(\phi - \Psi_{\text{i}}) \exp\left(\frac{(\phi - \Psi_{\text{i}})^2}{\sigma_{\text{i}}}\right). \tag{2.4}$$

The repulsion angular range of each obstacle assigns the acting field of each contribution and decides when these contributions merge together to form a single repellor. The formed behavioral dynamics can produce many complex behaviors (see (Althaus, 2003) for examples) where internal choices of movement direction through bifurcations can be taken depending on the actual internal state of the system (heading direction) and the surrounding environment (obstacles in the scene) which provides the system with an inherent flexibility. The sensed distances to obstacles represent the principal control parameters responsible of this flexibility in the behavioral dynamics. The same principles has been also extended to control robot manipulators (Iossifidis and Schöner, 2006; Ellekilde and Christensen, 2009; Reimann et al., 2011). For example, Iossifidis and Schöner (2006) used the previously illustrated behavioral dynamics to define the end-effector heading directions (azimuth, ϕ, and elevation,θ) in 3D space for target acquisition tasks in an unstructured environment observed by an operator or an external visual sensor (see Figure 2.2).

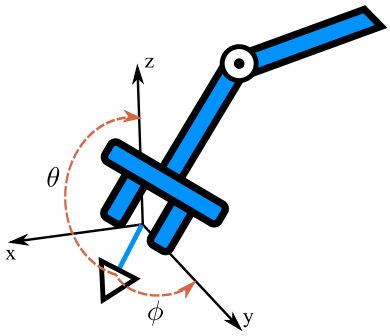


Figure 2.2: Behavioral dynamics for robot manipulator. Two heading directions (azimuth, ϕ, and elevation, θ) define the behavioral variables of a robot manipulator in a reaching task.

2.2.2 Organization of behaviors

Nowadays, autonomous robots are called to perform increasingly complex tasks in naturalistic physical environments. Moreover, robots should be able to react flexibly and change strategies in order to adapt to a dynamically changing surrounding. This requires mechanisms for decision making and behaviors organization that are tightly coupled to sensory information from the continuously changing environment.

The autonomous organization of behaviors is a crucial problem in robotics. Three major requirements have to be fulfilled by any mechanism meant to solve this problem. First, flexibility during behavioral organization is necessary so that changes can be brought into the original sequence of behaviors in response to variations in the sensorial context. Second, sequence decisions must be stabilized against noisy and fluctuating sensory input. Finally, the mechanism should

present a framework that is able to integrate, simultaneously, various sources of sensory information about the environment with an internal knowledge about the required logical order of the behaviors, their logical interplay and the historical state of the system. The dynamical systems theory provides the right tools to address these issues.

Competitive dynamics for behavioral organization

Complex tasks can be regarded as an ensemble of behaviors, called *elementary behaviors* (abreviated EB), that are fused inside the same behavioral dynamics and organized in a certain sequence or concurrently activated to accomplish the intended objective (Steinhage and Schöner, 1998; Schöner and Dose, 1992).

A neural dynamics architecture proposed by Large (1997); Schöner and Dose (1992) is capable of fusing all these constraints and representations into a single framework in the form of dynamical states. For a robotic task composed of N elementary behaviors $\mathrm{EB_i}$, $\mathrm{i} \in \{1, ..., N\}$, a task level competitive dynamics is defined by

$$\tau \dot{u}_\mathrm{i} = \alpha_\mathrm{i}(u_\mathrm{i} - u_\mathrm{i}^3) - \sum_{\mathrm{i} \neq \mathrm{j}}^{N} \gamma_{\mathrm{j,i}} u_\mathrm{j}^2 u_\mathrm{i} + \eta, \tag{2.5}$$

where u_i represents the *neuronal* activation of the the elementary behavior $\mathrm{EB_i}$; $\alpha_\mathrm{i} \in [-1, 1]$ is the competitive advantage of $\mathrm{EB_i}$; τ controls the time scale of the dynamics. The first two terms of the non-linear dynamics represent a supercritical pitchfork bifurcation (Strogatz, 1994). A single attractor $u_\mathrm{i} = 0$ is formed for $\alpha_\mathrm{i} < 0$. For $\alpha_\mathrm{i} \geq 0$, the fixed point $u_\mathrm{i} = 0$ gets destabilized and two attractors $u_\mathrm{i} = 1$ and $u_\mathrm{i} = -1$ form which implies the use of the absolute value $|u_\mathrm{i}|$ as the activation weight of the associated behavior (1 for activated, and 0 for deactivated). The last part of the dynamics in Eq. 2.5 is a competitive term where $\gamma_{\mathrm{j,i}}$ is the competitive interaction with $\mathrm{EB_i}$ of each of the remaining elementary behaviors $\mathrm{EB_j}$, $\mathrm{j} \in \{1, ..., N\}$ given that $j \neq i$. The competitive term destabilizes any formed attractor when the competitive interaction $\gamma_{\mathrm{j,i}}$ of at least one of the interacting behavior $\mathrm{EB_j}$ and its neuronal activation u_j are different from zero. $\eta \sim N(0, n)$ is a Gaussian white noise term with a mean 0 and variance n to guarantee escape from unstable states and assure robustness to the system.

In the absence of any competitive interaction, the architecture deploys the neuronal variable u_i to activate or deactivate the behavior with the largest competitive advantage α among the behaviors composing a complex robotic task. α may encode external sensory information and/or internal behavioral constraints like, for example, a behavioral *precondition* to express the activation conditions of a particular behavior. On the other hand, γ expresses a behavioral *suppression* mainly to prevent undesired simultaneous activation of particular behaviors.

Using these principles, the architecture implements competitions and corporations among these dynamical variables by stabilizing or destabilizing particular patterns of activation. Since the underlying variables are stable states, the behavioral organization is guaranteed to be stable. Moreover, the required flexibility is ensured via controlled instabilities dictated by the actual sensorial context and some internal states. The same form of the non-linear dynamics or some variants

of it has been used in various robotic applications (Steinhage and Schöner, 1997, 1998; Ellekilde and Christensen, 2009).

During the present work, an early implementation of the ball hitting task, presented in Chapter 4, has been realized using similar competitive dynamics (Oubbati and Schöner, 2013). The project consisted of building a mechanism for timed movements generation via oscillatory dynamics demonstrated in a ball interception task. The competitive dynamics permitted to switch between the different task movements. The mechanism was able to generate reliably the correct behavioral sequences based on internal constraints and external sensory information about the rolling ball. The approach was evaluated in simulation and successfully brought onto a real manipulator. However, the implementation severely suffered from different hardware limitations and the complexity of the resulting architecture.

Dynamic Field Theory for behavioral organization

In Section 2.2.2, I have introduced a neural dynamics architecture as a first step toward solving the problem of behaviors organization in robotics. Although successful in many applications, the particular mechanisms used in these earlier architectures rendered the design of these systems quite complex.

To address this issue, it is not sufficient to build dynamical mechanisms that activate or deactivate elementary behaviors based on some external sensory information and internal rules or constraints. In nature, cognitive agents are able to generate sequences of behaviors in a very smooth and robust fashion to accomplish particular tasks. Furthermore, each action is executed while respecting constraints arising from the environment, the agent's embodiment and the behavioral history. The dynamical systems approach offers a theoretical framework called *Dynamic Field Theory* to understand such cognitive processes where embodiment and situatedness are essential elements.

Dynamic Field Theory (or DFT) is a variant of the attractor dynamics approach to embodied cognition. The theory is based on a description of the dynamics of neuronal activation introduced by Amari (1977) and Wilson and Cowan (1973) to model pattern formation in neural tissues. DFT reflects the hypothesis that strong recurrent interactions in local populations of neurons form a basic mechanism for cortical information processing. DFT has found widespread applications in many areas that include perception of motion, visuo-spatial cognition and motor planning (Erlhagen and Bicho, 2009; Schöner, 2008).

In the following, I will give a brief introduction to the essential concepts and principles of DFT. After that, I will present a behavioral organization mechanism based on elements from DFT and suitable for robotics.

Dynamic Field Theory. Within the DFT framework, the states of a behaving agent are described by continuous activation functions defined over behaviorally relevant parameter spaces of different dimensionality. As described by Amari (1977), an activation function represents a dynamic neural field (or DF) that evolves in time according to the dynamic equation

$$\tau \dot{u}(x,t) = -u(x,t) + h + S(x,t) + \int \omega(x-x')f(u(x',t))dx' + \eta, \tag{2.6}$$

where $u(x,t)$ is the DF activation; $h < 0$ is the DF resting level; t is the current time; x is the space parameter of the behavior; $S(x,t)$ is the external input to the DF; $\eta \sim N(0,n)$ is a Gaussian white noise with a mean 0 and variance n to guarantee escape from unstable states and robustness to the dynamics; τ is the relaxation constant of the dynamics; $f(.)$ is a sigmoidal non-linearity used to shape the DF output from 0 for a negative argument to 1 for a positive argument and may take the form

$$f(x) = \frac{1}{1+e^{-\beta(x-x_0)}}, \tag{2.7}$$

with β defining the steepness, and x_0 the inflection point of the sigmoid output function. The last term of Eq. 2.6 describes the homogeneous lateral interaction within the DF activation where the output is convolved with a Gaussian interaction kernel given by

$$\omega(x-x') = c_{\text{exc}} \exp\left(-\frac{(x-x')^2}{2\sigma^2}\right) - c_{\text{inh}}. \tag{2.8}$$

The interaction kernel decomposes the lateral influence into two aspects, a local excitation of strength $c_{\text{exc}} \in \mathbb{R}$ over a range specified by σ and a global or mid-range inhibition set by $c_{\text{inh}} \in \mathbb{R}$. Nearby field locations are assumed to mutually excite each other, driving up activation, while distant locations are assumed to mutually inhibit, driving down activation. The sigmoidal non-linearity $f(u(x,t))$ implies that only sufficiently activated locations contribute to the interaction. Other forms of interaction kernel are possible.

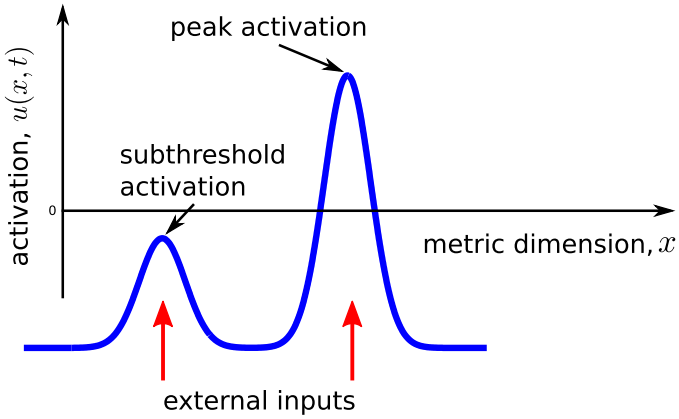


Figure 2.3: The activation pattern of a dynamic field $u(x,t)$ over a metric dimension x at time t. Within DFT, a peak activation represents an attractor state described by its location. A sub-threshold activation is an input-driven attractor representing graded information. This figure is in part adapted from (Schöner, 2008).

The DFT core idea is to define, via dynamic fields, neural representations over continuous metric dimensions. The metric dimension x reflects the behavioral space parameters and may span a visual space, object features, or movement

parameters over which the activation function $u(x,t)$ is defined. Thus, representing metric information as dynamical state variables requires a second dimension. The extra dimension is graded and shows the activation extent such like the strength, certainty, or intensity of a given location along the metric dimension (Figure 2.3). Depending on the feature space, the behavioral space parameters may be encoded along a single dimension or more (Figure 2.4).

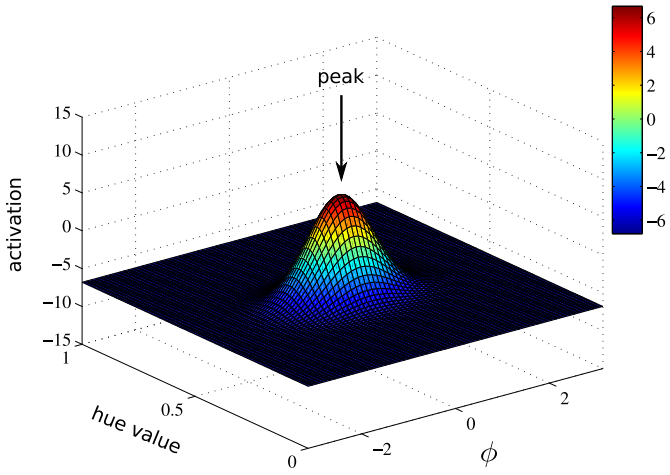


Figure 2.4: An example of a dynamic field defined over a two-dimensional feature space: hue value and heading direction ϕ. The DF receives, as an input, the robot's camera image after a segmentation process. Such a DF can be used as a perceptual system for a robot tracking a colored object. The robot's heading direction ϕ can be extracted from the DF by reading the peak position defined by the detected hue value. The color bar depicts the activation level of the dynamic field.

In the absence of external input $S(x,t)$, the field activity stabilizes at an attractor state set by the resting level $h < 0$. Inputs to the field activation may come from sensorial information or other dynamic fields and can be localized to specify a particular value along the metric dimension or homogeneous to affect the overall activation level.

When a sub-threshold and localized input (preshape) is introduced, the activation pattern forms an input-driven attractor in which the contribution of the neuronal interaction is negligible (due to the non-linearity $f(x)$). In many applications, this regime can be exploited to represent, for instance, ambiguous prior information about the dimension spanned by the field by encoding probability of choices. It can also be merely used to filter external inputs.

If the input strength is sufficiently increased, the neuronal interaction, driven by the bell shaped interaction kernel $\omega(x-x')$ and amplified by the stabilization term $-u(x,t)$, permits to elevate the sub-threshold activation level. During this process, the system is brought through a bifurcation, referred as *detection instability*, to stabilize a localized peak activation beyond the inducing input level (the system is bistable). The peak solution is stabilized against decay by the local excitatory interaction and from diffusion by the global inhibitory interaction. The peak stays stable even if the localized input is fluctuating, reduced, or slightly

shifted as shown in Figure 2.5 for a one-dimensional field. Within DFT, peaks of activation represent the elementary units of representation.

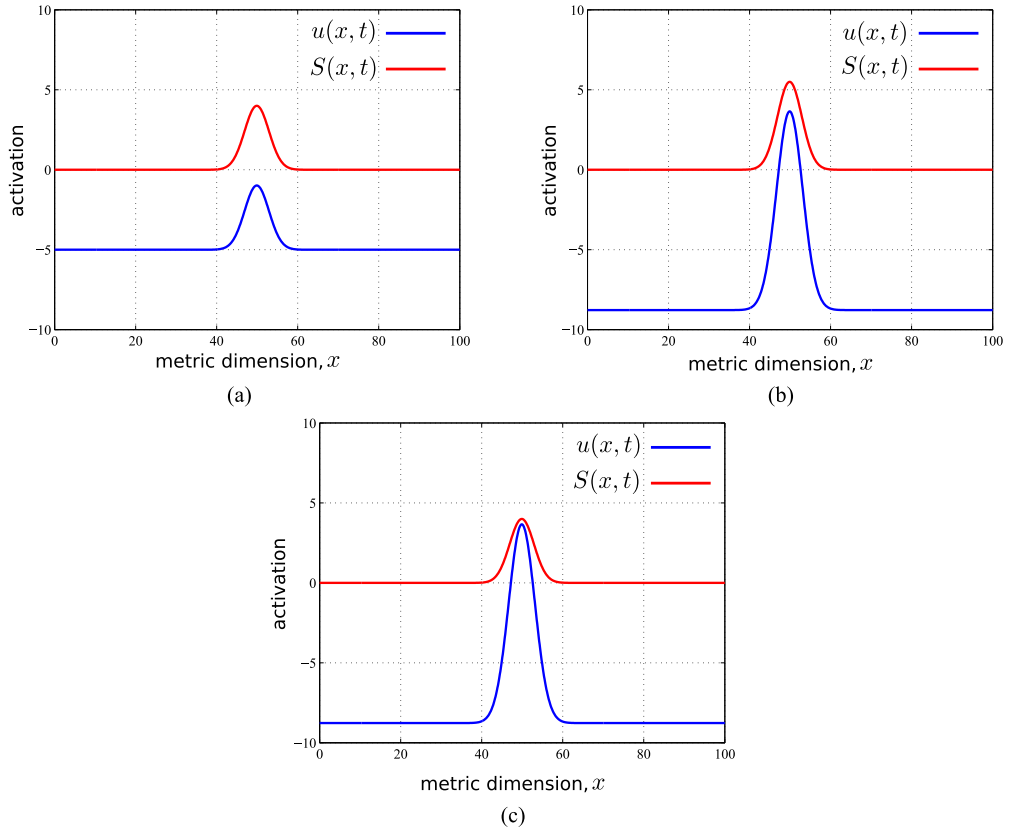


Figure 2.5: Activation patterns of a dynamic field $u(x,t)$ with different level of external input $S(x,t)$. (a) A sub-threshold input results in an input driven sub-threshold attractor. (b) If the input is increased, the field goes through a detection instability and a peak activation forms. (c) Even if the input is decreased to the level of (a), the peak stays stable.

These stabilized patterns of activation are used to build elementary cognitive functions like *decision making* mechanisms (see (Schöner, 2008; Sandamirskaya and Schöner, 2008; Erlhagen and Bicho, 2009) for a review about these mechanisms and their use). One form of decision making is *detection decision*. By elevating the activation level of an inhomogeneous stream of sub-threshold localized inputs in a DF (for example, by increasing the resting level $h < 0$), only the sub-threshold with the highest input will form a self-stabilized peak leading to select a particular location on the feature dimension. This instability may be used to implement methods of categorization or learning processes. If the general level of activation and the local excitation are sufficiently high, the DF interaction pattern may sustain a localized peak of activation even if the initial input is completely removed. Such sustained patterns of activation may represent a form of metric working memory of some behavioral history.

Another form of decision making is *detection selection* among multiple localized inputs. When two metrically close inputs are introduced into the field dynamics, a single broader peak forms at an averaged location between the two.

On the other hand, when the inputs are distant enough, a single peak activation, centered around one of the inputs, forms. This choice may be influenced by fluctuations, noise or prior activation history.

Stability properties of the field dynamics permit to define meaningful states of the system as attractors while the different dynamic regimes and instabilities allow the implementation of decision making and abstract cognitive mechanisms. Therefore, the DFT framework offers a suitable and robust interface between noisy and highly fluctuating information at a lower sensory-motor layer, and higher layer cognitive functionalities.

The level of abstraction to represent activity in the cortical tissue may vary from an entire subsurface of the cortex to few or even a single neuron. In fact, activity patterns in a DF might be represented by the dynamics of single neurons or nodes. The activity of those nodes is governed by similar dynamics to the dynamics of a neural field and is given by

$$\tau \dot{u}(t) = -u(t) + h + S(t) + c_{\text{exc}} f(u(t)) + \eta. \tag{2.9}$$

Similarly to Eq. 2.6, the self excitation term $f(u(t))$ of strength $c_{\text{exc}} \in \mathbb{R}_{\geq 0}$ provides a bistable dynamics to the dynamical node. When the input $S(t)$ pushes the node activity $u(t)$ beyond the threshold, the system enters a detection instability (modulated by $f(u(t))$) and the positive activity is sustained against possible fluctuations of the input. If a sufficiently large self excitation is applied, the node will maintain its positive activation even if the input ceases. The node can be brought again to the inactive state by an inhibitory input. A dynamical node can be considered as a neural field of dimensionality zero.

Coupling between several neural fields, even of different dimensionality, is possible and permits to propagate positive activations between different representations or information resources as in the context of scene representation (Zibner et al., 2011) or object detection and position estimation (Richter et al., 2012). This property is due to the stability of the attractor states so that the qualitative dynamics stays invariant and no unpredictable behaviors can be produced.

In practice, coupling between a dynamical field $u_{\text{i}}(x,t)$ and other $u_{\text{j}=1\ldots\text{N}}(x,t)$ fields within an architecture is generally performed via the sum of the field's outputs shaped by the sigmoidal non-linearity of Eq. 2.8. The connections strength can be tuned by synaptic weights $c_{\text{j,i}} \in \mathbb{R}_{\geq 0}$ as follows

$$S_{\text{i}}(x,t) = \sum_{\text{i}\neq\text{ j}}^{N} c_{\text{j,i}} f(u_{\text{j}}(x,t)). \tag{2.10}$$

The sigmoidal non-linearity ensures that only supra-threshold activities is propagated to be used as excitatory or inhibitory inputs in the destination field. Projecting a field's positive activation can be realized from lower to higher dimensional fields or the opposite along a shared dimension. As an example, a higher dimensional field can provide supra-threshold activation (by summing up the activation along the extra dimension(s)) and switch on a dynamical node so that the later can play the role of a *peak detector*. Dynamical nodes of this type are used in the DFT framework for behavioral organization described in the next Section.

DFT based framework for behavioral organization

Sandamirskaya and Schöner (2011) and Richter et al. (2012) explored how the DFT framework can be extended to integrate a behavioral organization mechanism with rules that can connect with perceptual, cognitive, and motor processes. Expressing these rules using DFT elements makes it possible to consider behavioral constraints, in selecting the most appropriate next action, emerging from perceptual inputs, past behavioral states and the particular embodiment of the agent (i.e., motor and sensory systems). Such a mechanism will certainly increase the autonomy of robots situated in natural environments while the intrinsic stability of states of the DF dynamics will provide the system with the necessary robustness.

Using the notation of Steinhage and Schöner (1998), a task can be considered as an ensemble of elementary behaviors (EBs) that should be executed in a specific order and manner to accomplish an objective. For example, the task of grasping an object can be subdivided into several subtasks that may be perceptual like detecting or tracking an object, or motor oriented like moving the end-effector or closing the gripper.

During a behavioral sequence generation and since the behaviorally meaningful states in DFT are attractors, we need a dynamical mechanism to destabilize the current state in order to proceed to the next one. This problem was addressed by implementing in DFT concepts from the theory of intentionality (Searle, 1983) like the *intention* and *condition of satisfaction* in performing actions. The argument behind this conceptual choice is that every action-related state of a system's dynamics needs a correspondent condition of satisfaction (CoS) state, which signals accomplishment of the current action and triggers a sequential transition. For the grasping example, the first action of a cognitive agent will start with the intention to detect the object and estimate its position. Once the object is detected (first action's CoS fulfilled), the second action should initiate the end-effector movement toward it with the intention to be as close as possible to the grasping point. As soon as the end-effector reaches the object (second action's CoS fulfilled), the last action will close the gripper with the intention to execute the grasp with a certain strength or a specific closing angle (last action's CoS fulfilled).

Following these concepts, an elementary behavior (EB) is modeled so that it comprises both, a higher level representation of the intention and CoS of an action, as well as a low level sensory-motor representation that receives inputs directly from robot's sensors. Using this scheme, architectures of several EBs, as well as different behavioral constraints between these EBs, can be implemented using DFT components.

Elementary behavior. In an EB (see Figure 2.6), the intention and CoS states of an action are modeled using two dynamical nodes, intention (i) and CoS (c). The state of the intention node determines if the EB is active or inactive (for example, the intention to reach for an object in a grasping task). When the EB is active, the state of the CoS node determines whether the behavior has been completed (the object has been reached). To ensure that the CoS node can turn on only if the EB is active, the intention node provides a sub-threshold

excitatory input to the CoS node. However, this input is not sufficient to activate the CoS node and additional activation describing the current behavioral state and the target is needed. After completing the behavior, the CoS node turns on and inhibits the intention node. The intention and CoS nodes are coupled to two dynamical fields, the intention and CoS fields, that represent graded information about the EB intention and CoS. The intention and CoS fields play the interface role between the higher level representation of a behavior and the low level sensory-motor systems.

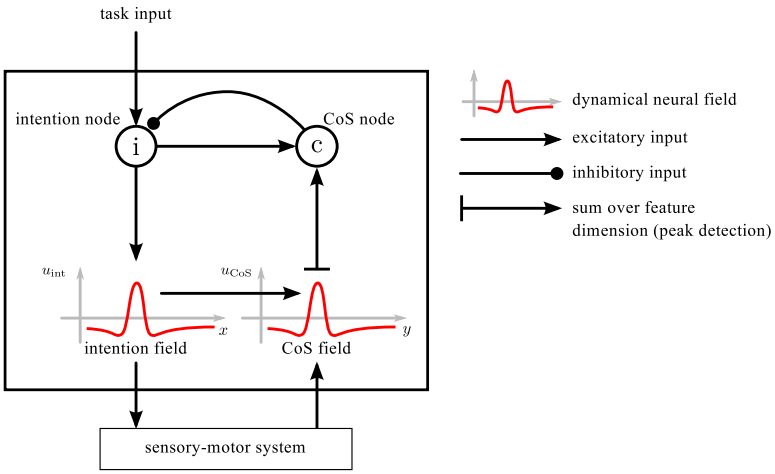


Figure 2.6: DFT-based model of an elementary behavior (EB). This figure is in part adapted from (Richter et al., 2012).

The intention field permits to encode, over a metric feature x of any dimension, the behavior intention (for example, the intended movement target of the end-effector in the grasping task) that is forwarded as movement parameters to the dynamics that drive the effector behavior. In addition, the behavior intention is propagated to the CoS field as a sub-threshold activation. Depending on the nature of the EB, the behavioral target may be received from a perceptual system or from another EB.

On the other hand, the CoS field represents the current perceptual state of the behavior being executed (in our example, the current end-effector position that can be acquired as a feedback from the hardware); and it is used to signal the accomplishment of the behavior when the action target and the current perceptual states overlap. Depending on the type of the sensory input, the CoS field may share the same metric feature y and dimension with the intention field or be defined differently. For instance, if the end-effector is tracked by some visual system during the movement instead, the CoS field should be defined over a perceptual feature dimension.

Executing our exemplary grasping task requires a set of EBs that are coupled together through behavioral constraints to form an architecture and sequentially activated to generate the necessary actions. In the resulting architecture, the task is modeled as a dynamical node that activates, through excitatory task inputs, the intention nodes of all EBs involved, as well as, the respective constraints.

To illustrate how the DFT based behavioral organization mechanism works, I will take the execution example of one EB.

To trigger the behavior's start, the task dynamical node (t) is activated by some external input $I > |h|$ according to

$$\tau \dot{u}_{\mathrm{t}}(t) = -u_{\mathrm{t}}(t) + h + I + c_{\mathrm{exc,t}} f(u_{\mathrm{t}}(t)) + \eta. \tag{2.11}$$

Once activated, the task excitatory input $f(u_{\mathrm{t}}(t))$, with a strength set by the synaptic weight $c_{\mathrm{t,i}} \in \mathbb{R}_{\geq 0}$ such that $c_{\mathrm{t,i}} > |h|$, turns on the EB intention node described by

$$\tau \dot{u}_{\mathrm{i}}(t) = -u_{\mathrm{i}}(t) + h + c_{\mathrm{exc,i}} f(u_{\mathrm{i}}(t)) + c_{\mathrm{t,i}} f(u_{\mathrm{t}}(t)) + c_{\mathrm{CoS,i}} f(u_{\mathrm{CoS}}(t)). \tag{2.12}$$

After accomplishing the behavior, the intention node gets deactivated by an inhibitory input $f(u_{\mathrm{CoS}}(t))$ coming from the CoS node that is tuned by $c_{\mathrm{CoS,i}} \in \mathbb{R}_{\leq 0}$ so that $|c_{\mathrm{CoS,i}}| > (c_{\mathrm{t,i}} - |h| + c_{\mathrm{exc,i}})$.

When the intention node gets activated, a homogeneous boost $f(u_{\mathrm{i}}(t))$ is provided to the EB intention field and to the CoS node. Combined with a localized sub-threshold activation $S_{\mathrm{perc,int}}(x,t)$ encoding the target of the action, the intention field forms a peak activation governed by

$$\tau \dot{u}_{\mathrm{int}}(x,t) = -u_{\mathrm{int}}(x,t) + h + S_{\mathrm{int}}(x,t) + \int \omega(x-x') f(u_{\mathrm{int}}(x,t)) dx' + \eta, \tag{2.13}$$

where

$$S_{\mathrm{int}}(x,t) = c_{\mathrm{i,int}} f(u_{\mathrm{i}}(t)) + c_{\mathrm{perc,int}} S_{\mathrm{perc,int}}(x,t). \tag{2.14}$$

The connection weights $c_{\mathrm{i,int}}, c_{\mathrm{perc,int}} \in \mathbb{R}_{\geq 0}$ are tuned so that the intention field forms a peak only if the behavior target was defined and the intention node is turned on.

The intention peak activates the movement dynamics that starts driving the effector to execute the action. The interaction kernel $\omega(x-x')$ consists of a single excitatory mode with additional global inhibition to prevent multiple peaks to form in the case where additional sensory inputs are provided to the intention field (an action should have only one intention at a time along the feature dimension).

The localized peak activation in the intention field represents the behavior intention and is propagated to the CoS field as a sub-threshold excitatory input $f(u_{\mathrm{int}}(x,t))$. Combined with the progressing sub-threshold input representing the current perceptual state of the behavior $S_{\mathrm{perc,CoS}}(y,t)$, the CoS field activation dynamics is described by

$$\tau \dot{u}_{\mathrm{CoS}}(y,t) = -u_{\mathrm{CoS}}(y,t) + h + S_{\mathrm{CoS}}(y,t) + \int \omega(y-y') f(u_{\mathrm{CoS}}(y,t)) dx' + \eta, \tag{2.15}$$

where

$$S_{\mathrm{CoS}}(y,t) = T_{\mathrm{int,CoS}} \diamond c_{\mathrm{int,CoS}} f(u_{\mathrm{int}}(y,t)) + c_{\mathrm{perc,CoS}} S_{\mathrm{perc,CoS}}(y,t). \tag{2.16}$$

The weights $c_{\text{int,CoS}}, c_{\text{perc,CoS}} \in \mathbb{R}_{\geq 0}$ are set so that only an overlap between the intention and the perceptual inputs can induce a peak in the CoS field. A peak in the CoS field signals the successful completion of the EB by indicating that the behavior target state has been reached. If the intention and CoS fields do not share the same feature dimensions, a match mapping process $T_{\text{int,cos}}$ via a matching operation ($\diamond$) is required for the input coming from the intention field.

The EB CoS node plays the role of a peak detector for the CoS field and gets activated through the integral of the thresholded activation $f(u_{\text{CoS}}(y,t))$ as follows

$$\tau \dot{u}_{\text{c}}(t) = -u_{\text{c}}(t) + h + c_{\text{exc,c}} f(u_{\text{c}}(t)) + c_{\text{i,c}} f(u_{\text{i}}(t)) + \int c_{\text{CoS,c}} f(u_{\text{CoS}}(y,t)) + \eta. \quad (2.17)$$

The connection parameters $c_{\text{CoS,c}}, c_{\text{i,c}} \in \mathbb{R}_{\geq 0}$ are chosen so that the CoS node turns on only if the intention node is currently activated $f(u_{\text{i}}(t))$ and there is a peak in the CoS field.

When the CoS node is activated, a cascade of instabilities in the elementary behavior starts. The intention node turns off and ceases to excite the intention field; which goes through a reverse detection instability. Consequently, the CoS field looses its activation and turns off the CoS node. After deactivating the task node, all elements of the EB are now inactive and the action has been accomplished.

Behavioral constraints. To execute a complex task like grasping an object, several EBs are required. The set of EBs comprises the behavioral repertory of the agent (detect an object, move end-effector toward the object, ...). To execute the task, the activation of these EBs has to be organized in time. This is done by encoding behavioral constraints on the sequential activation of EBs. The constraints may relate to the embodiment of the agent or can be task specific.

Task specific constraints implement simple rules of behavioral organization by coupling EBs via dynamical nodes that can be activated or deactivated. These constraints are conceptually similar to those introduced in Section 2.2.2 and involve a precondition, suppression and competition between EBs.

A precondition constraint (p) ensures that EB_1 can be activated only if EB_0 has been successfully completed as shown in Figure 2.7. The precondition node is activated by the task input and inhibits the intention node of EB_1. In turn, the precondition node is inhibited by the CoS node of EB_0. Thus, when the CoS node is activated, the intention node of EB_1 is dis-inhibited. As a result, EB_0 and EB_1 are activated sequentially. Alternatively, a precondition constraint can also model a dependence of an EB to some perceptual information and is referred as a perceptual constraint.

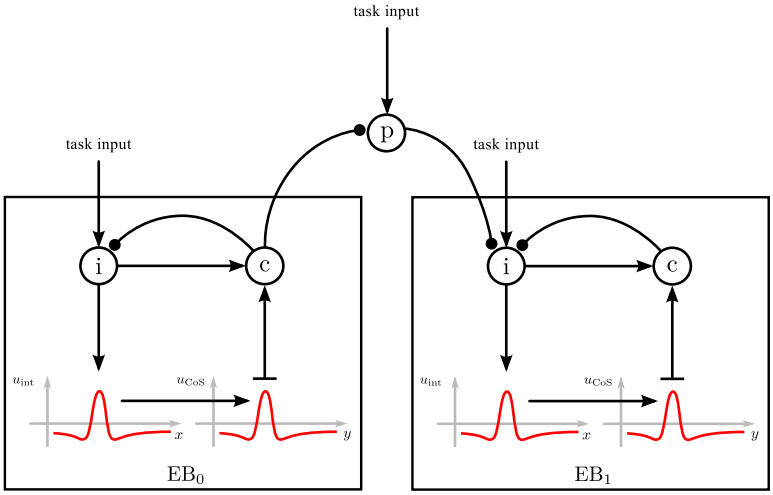


Figure 2.7: A precondition constraint between two elementary behaviors EB_0 and EB_1. This figure is in part adapted from (Richter et al., 2012).

A suppression constraint (s) prevents EB_1 to be or to become active while EB_0 is being executed as illustrated in Figure 2.8. Similarly to the precondition node, the suppression node inhibits the EB_1 intention node. However, the suppression node is activated both by the task input and input from the EB_0 intention node and is released as soon as the EB_0 is completed or terminated.

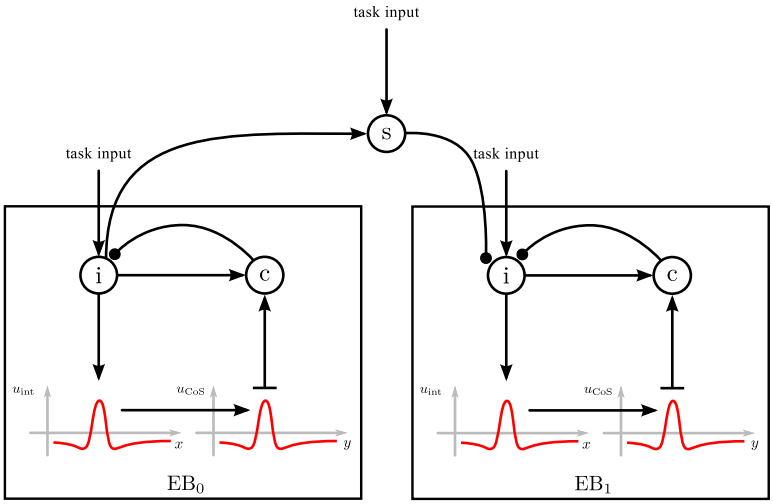


Figure 2.8: A suppression constraint between two elementary behaviors EB_0 and EB_1. This figure is in part adapted from (Richter et al., 2012).

Using two suppression nodes, mutual suppression between two EBs can be used to implement competition as depicted in Figure 2.9. In this configuration, the risk of simultaneous active states of EB_0 and EB_1 may happen but can be reduced by tuning the dynamics time scales.

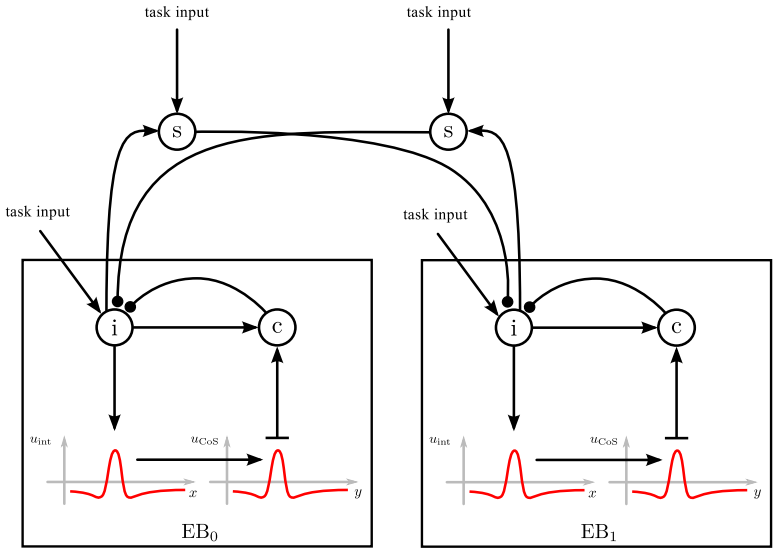


Figure 2.9: A competition constraint between two elementary behaviors EB_0 and EB_1. This figure is in part adapted from (Richter et al., 2012).

The mathematical formulation of the dynamical nodes follows the general mathematical description of the nodes that compose an EB.

Within the same framework, embodied constraints can also be implemented. An example may arise when an action depends on the current state of the system like deciding to open or close the gripper of an agent.

Based on the DFT framework, a robotic architecture for action selection and behavioral organization has been developed and successfully tested in a grasping task performed by a humanoid robotic platform (Richter et al., 2012). In the present work, the DFT framework for behavioral organization will be extended to the problem of generating and organizing timed movements in coordination with sensed events. The resulting architecture will be tested in simulation with different robotic tasks and in a hardware implementation of a robot ball hitting scenario.

2.3 Timed motor acts: dynamical systems perspective

Motor control is the process by which humans and animals use their neuromuscular system to activate and coordinate the muscles and limbs involved in the performance of a motor skill (Rosenbaum, 2009). One major issue in studying human motor control is how do we control the sequencing and timing of our behaviors. In every day life, generating movements in the right order is important, but generating the movements with the right timing can be even more so. People can, for example, perform very well in batting an incoming ball, playing musical instruments or dancing with a partner. Exemplary tasks that require a certain sense of synchronization (or tempo), hence, a capacity to generate actions just "at the right time".

Timing is defined in terms of the stability of the temporal structure of an action: A timed movement takes place in time and compensates for perturbations

by either holding up or advancing the movement so as to restore as much as possible the movement time (Schöner, 2002; Bootsma and Van Wieringen, 1990). Timed movements are actions whose temporal form is reproducible and stable in the face of perturbations as attested by the invariant velocity profile of the end-effector (Morasso, 1981).

Most commonly, timing is studied in rhythmic movements like dancing, walking or running. Nevertheless, stable temporal relationships and reproducible movement times occur also during discrete motor acts. For instance, if a hand is slowed down when reaching for an object, the hand opening is slowed as well, even though, no functional reasons justify that (Jeannerod, 1984). This implies a form of coordination that tends to keep stable relative timing between the different movement components in face of varying environmental conditions or perturbations. In addition, coordination is a very important aspect in action-perception schemes (Warren, 2006; Goodale and Humphrey, 1998; Bootsma and Van Wieringen, 1990). Indeed, humans and other animals are able to generate behavioral patterns that are tightly coordinated with the environment in order to achieve specific goals like catching or hitting moving objects. Behavioral patterns that are yet stable and highly adaptive.

The dynamical systems theory offers a suitable and unified framework to account for these experimental observations. In the following, I will give a brief description of the dynamical concepts and tools used to analyze and understand these phenomena.

2.3.1 Timing dynamics

The dynamics approach focuses on the ability of the nervous system to maintain persistent patterns of behavior in complex varying environments. Thus, the ability to immunize these behaviors from the the stochastic nature of the environment and uncertainties in percepts and acts. The stability of a behavioral pattern is then a central requirement for meaningful and efficient behavior outcomes. Stability is a central concept in the dynamical systems theory.

The idea that motions of biological systems may be governed by dynamical systems has been a recurrent topic of discussion in human motion science and biology. The dynamical systems approach defines the relevant variables of an observed system as attractor states. This formulation ensures a stable performance for the underlying states while providing them with the necessary flexibility and adaptation characteristics to changing conditions and constraints (see Section 2.2.1 for more details).

By exploiting these concepts, dynamic timing models have proven their capacity to account for absolute timing, relative timing and coordination in general (Schöner, 2002). From the dynamical systems theory point of view, all timing models are limit cases of limit cycle attractors (Schöner, 2002). Limit cycle attractors are stable self-sustained non-linear oscillators (Strogatz, 1994). In other words, these systems oscillate even in the absence of external periodic forcing and are stable periodic solutions to which the system is attracted from nearby states. If the oscillator state is slightly perturbed from the cycle orbit, it resists and returns back to the original cycle demonstrating it's stability property. This is an essential property to realize any inter-oscillator coupling (Schöner, 2002). In

dynamic timing models, the intrinsic dynamics of the oscillator set the temporal order of the movement and the coordination between movements is captured in terms of relative timing between the states of coupled oscillators.

In general, this kind of dynamic timing models use exactly solvable nonlinear oscillators that contain a bifurcation to a limit cycle. This choice is guided by the need to have complete control over the timing properties and stable states of the dynamics via temporal-space control parameters (i.e., the frequency/cycle time and radius). This will permit to show how the timing properties of discrete or rhythmic movements can be described by the pattern dynamics and discuss the relation of the pattern variables to the observed effector movements (Schöner, 1990). Limit cycle attractors represent essential elements in the dynamical systems toolbox to model, among others, human timed behaviors and movements coordination (Schöner, 1990; Schöner and Kelso, 1988). Indeed, experimental observations on reproducible amplitude-frequency relationships and on relaxation after mechanical perturbations of the movements are accounted for by the radial stability of limit cycle oscillators (Schöner, 2002).

In such kind of modeling, the commonly used limit cycle oscillator is the Hopf oscillator. A Hopf oscillator is a two-dimensional dynamical system with the simplest polynomial formulation containing a bifurcation. This instability occurs from a fixed point to a limit cycle as a parameter crosses a critical value. In a differential equation, a Hopf bifurcation typically occurs when a complex conjugate pair of eigenvalues of the linearized flow at a fixed point becomes purely imaginary (see Section 3.1 for an example of application). In addition, the Hopf oscillator is completely solvable. Another type of limit cycle oscillator is inspired by the neuronal structure and properties of the brain, the Amari oscillator (Amari, 1977; Schöner, 2002). However, the main inconvenience of the Amari oscillator as a model for movement trajectories is the difficulty to control parametrically the limit cycle time.

These ideas and concepts have found a large resonance as a theory of development that describe the behaviors of complex physical and biological systems. For example, in a theoretical modeling,Haken et al. (1985) and Kelso (1984), succeeded to explain, in terms of the relative phase transitions, the human's interlimb coordination patterns by means of coupling nonlinear oscillatory processes. Using a similar concept, the coordination of phase and antiphase rhythmic movements could also be captured in a framework that reproduced the spontaneous phase transitions as the frequency is varied (Schöner, 2002; Schöner and Kelso, 1988). Schöner (1990) demonstrated that the same conceptual framework can account for the coordination of discrete movements if a mechanism is postulated that switches a limit cycle to some postural state after a single (or even a half) cycle based on behavioral information. In particular, Schöner (1990) showed that in a coordination pattern between two effectors, if one effector is slowed down or accelerated, the other one may change its time course to reduce the discrepancy in relative timing. In a dynamical analysis of a paddle-juggling system, Schaal et al. (1996) demonstrated that a juggling pattern can be characterized and evaluated in terms of the stability properties of a nonlinear system of coupled oscillators.

These dynamical models for discrete and rhythmic movements generation lie under the nomination of central pattern generators (Ijspeert, 2008). Other

approaches based on dynamical systems theory can also be used to generate coordinated discrete or rhythmic movements (see (Degallier and Ijspeert, 2010) for a review and application examples).

The work presented in this thesis builds on some of these principles to address the problem of generating and organizing sequences of timed movements in coordination with sensed information. In particular, I exploit a framework for discrete timed movements generation introduced by Schöner and Santos (2001) to build the timing dynamics inside my model. The Movement planning combines stable fixed points to define postural states and limit cycle attractors to model the timed movement phases.

2.3.2 Action-Perception dynamics

The empirical field of action-perception systems is aimed to describe the correlation between a behavior performed by an agent (human or animal) and the perceptual information from its environment that drives this behavior. In nature, these patterns of behavior can be observed at different levels and in a variety of tasks, in which the action is initiated and steered by some external sensory information. Such tasks include, for instance, batting a ball in a baseball game, catching a flying boomerang, manipulating an object, and avoiding obstacles during locomotion. In performing this kind of actions, humans and animals exhibit stable and highly flexible behavioral patterns. This are two properties that can be captured inside the dynamical systems approach.

In action-perception systems, the perceptual information is generally described as a variable that expresses a temporal constraint from the task space. This variable can be computed as the ratio between a measure related to an approaching object like a distance, an angle, or a diameter and the rate of change of this measure. This variable finds its origin in biology and is generally computed as the expansion rate in the retina of the observed object as it approaches (Hancock and Manser, 1997; Hecht and Savelsbergh, 2004; Tresilian, 1991). The obtained time measure provides an estimate of the remaining time for the object to reach the observer. This time estimate is commonly called τ and permits to obtain a *time-to-contact* judgment (Hecht and Savelsbergh, 2004; Hancock and Manser, 1997; Bootsma and Van Wieringen, 1990). Eventually, this estimate may then be used to trigger an action (for example, an avoidance or an interception) when the former reaches a critical value. However, this time estimate is based on too many assumptions, like a constant velocity approach, to be precise. In reality, this is rarely the case.

Actually, the problem of which information is used in order to decide when to trigger a movement has not been yet solved. It is likely that humans use a perceptually specified time-to-contact measure to control the timing and adapt their movements during execution (Katsumata and Russell, 2012). Hence, the operational timing hypothesis implies that humans have to initiate their movements when a time-to-contact reaches a certain critical value.

Within the dynamical systems approach, an agent and its environment are described as two dynamical systems that are informationaly and mechanically coupled to execute a set of behaviors (Warren, 2006; Schöner and Jeka, 1998). An argument supported, to some extent, by Gibson (1998). These ideas permit

to unify the two conceptual visions about action-perception systems. A vertical conception where a central action planner generates the behaviors according to some perceptual control information, and a more horizontal one in which behaviors emerge by direct coupling between sensorial information and parameters specifying the actions (Warren, 2006).

Dynamical models for action-perception systems couple the perceived timing of environmental events to the dynamics describing the actions (Schöner, 1994; Warren, 2006). In this context, the perceived timing plays the role of behavioral information and it is used to trigger the different behaviors (Schöner, 1990). The stability property of the approach permits to maintain particular states in face of various types of perturbations to which the system is continuously exposed, while instabilities through bifurcation provide the system with the necessary flexibility. Moreover, integrating the timing constraint in generating the behaviors can be conceptually accounted for. These characteristics ensure a stable action-perception coupling and an invariant relative timing between the movement and the perceptual event, i.e., coordination.

Experimental studies on humans and animals in several action-perception scenarios led to develop several dynamical models as prediction tools for the obtained observations. Examples of scenarios include: posture analysis in moving visual environment (Schöner, 1991), plummeting gannets (Lee and Reddish, 1981), landing flies (Wagner, 1982), hitting a falling ball (Bootsma and Van Wieringen, 1990; Lee et al., 1983), and ball juggling by a racket (Dijkstra et al., 2004; Siegler et al., 2003; Sternad et al., 2001; Schaal et al., 1996). For instance, an action dynamics with two postural states for the initial and final positions of the system, and a limit cycle for the actual action that links the two states, could account for the timed movement of the wing of a diving gannet as it approaches the water, and for the flight motor of a landing fly as it approaches a landing surface (Schöner, 1994). Behavioral dynamics developed by Warren (2006) permitted to analyze the passive and active stabilization regimes in a ball juggling and object balancing scenarios. In all cases, the visual information flow, that expresses a time measure of distance, is coupled to the action dynamics as a behavioral information used to initiate and steer the action.

The movement model proposed in this work is a dynamical action-perception system. In robot ball catching and hitting scenarios, the developed timing dynamics and the behavior organization mechanism permit to generate sequences of timed actions directed at a moving ball. Both the timed movements and the behavior organization mechanism are tiedly coupled to perceptual information about the ball.

Chapter 3

Dynamical model for sequence generation of timed movements

In robotics, the autonomous sequencing of timed motor acts is not a trivial task, particularly when online coupling to low level and often noisy sensory information is needed to initiate, steer, and terminate the different movement phases. Moreover, an autonomous agent is supposed to operate in time-varying environments where it needs to adapt the movements and react autonomously to external disturbances either by activating, aborting, or reactivating the appropriate behavior in coordination with sensed events.

In this chapter, I present a dynamical model for timed movements sequence generation. The proposed model permits to generate and flexibly organize sequences of timed behaviors. The design integrates non-linear oscillators for timed movements generation and a hierarchical neural dynamics architecture for organizing in time and space the timed behaviors. The autonomous generation of movement sequences is tiedly coupled to sensory information and adapts online to external perturbations.

To demonstrate the core properties of the model, a robotic simulation of a ball catching scenario is used. For this task, single dimension movements are executed by an end-effector to catch a free-falling ball before moving back to a base station. Although very simplified, the task is sufficient to illustrate the inherent features of the model. Extending the model to a more complex robotic task is considered in Chapter 4.

This chapter is structured as follows. I will start by presenting the dynamics of a timed movement in Section 3.1 followed by an introduction of different mechanisms for temporal stabilization of movements in Section 3.2 In Section 3.3, an intuitive description of the hierarchical neural dynamics architecture is provided. The robotic catching task is illustrated in Section 3.4. Results of the robotic simulation are presented in Section 3.5 and discussed in Section 3.6.

3.1 Dynamics of timed movement

The dynamics of a timed movement is a modified version of the canonical form proposed by Schöner and Santos (2001). It consists of a dynamical system for a pair of timing variables $(x\,,\,y)$. Although only x is used to define the relevant task variable at any time, the variable y is an auxiliary needed to represent the

oscillatory solution. The dynamical system combines two regimes of operation and permits an end-effector to start a timed movement from a postural state toward a target location within a specified time duration. The time course of the variables x and y is governed by

$$\tau \begin{pmatrix} \dot{x} \\ \dot{y} \end{pmatrix} = \underbrace{-c_{\text{post}}\, a \begin{pmatrix} x - x_{\text{post}} \\ y \end{pmatrix}}_{\text{postural state}} + \underbrace{c_{\text{hopf}} H(x, y)}_{\text{oscillatory regime}} + \eta. \tag{3.1}$$

The dynamical system can be operated in two separate dynamic regimes by controlling the "neurons" $c_{\text{post}}, c_{\text{hopf}} \in [0, 1]$. In practice, the sequential activation of the neurons is controlled by noisy signals defined by sigmoidal nonlinearities, thus with a tendency to have values either close to 0 or close to 1.

In the postural regime, $(c_{\text{post}} \approx 1; c_{\text{hopf}} \approx 0)$, the timing variable x relaxes to the fixed point attractor x_{post} with a strength set by the term $a > 0$. The time scale of the dynamics is controlled by τ that can be tuned to overcome delays during the switch between the dynamic regimes and so, accommodate real time implementations. The dynamics are augmented by Gaussian white noise $\eta \sim N(0, n)$ with mean 0 and variance n. The noise signal guarantees escape from unstable states and assures robustness to the system.

The oscillatory regime, $(c_{\text{post}} \approx 0; c_{\text{hopf}} \approx 1)$, stabilizes a periodic solution along a limit cycle attractor defined by the normal form of a Hopf bifurcation (Strogatz, 1994). The Hopf oscillator has many interesting properties, among which: (a) it has a unique periodic solution that is globally stable, (b) this solution can be found analytically and is a perfect sine, and (c) the frequency and the amplitude are explicit parameters that can be tuned and modified online. The Hopf term is given by

$$\begin{aligned} H(x, y) = & \begin{pmatrix} \lambda & -\omega \\ \omega & \lambda \end{pmatrix} \begin{pmatrix} x - r - x_{\text{init}} \\ y \end{pmatrix} \\ & - \left((x - r - x_{\text{init}})^2 + y^2 \right) \begin{pmatrix} x - r - x_{\text{init}} \\ y \end{pmatrix}, \end{aligned} \tag{3.2}$$

where the oscillator intrinsic frequency ω defines the movement cycle time $T = \frac{2\pi}{\omega}$ and the amplitude of the Hopf contribution is specified by $\lambda = r^2$ where $r = \frac{x_{\text{target}} - x_{\text{init}}}{2}$ is the oscillator radius. In normal case, to generate a discrete timed movement, the oscillatory regime is activated during a half cycle only.

In phase portrait, the Hopf cycle is shifted along x-axis by the Hopf radius r and initial state x_{init}. This modifications introduce a variable offset along the movement dimension spanned by the timing variable x. The new system permits to define a modulated harmonic trajectory for the timed movement so that x smoothly rises from the current x_{init} toward the target location x_{target} during the oscillatory regime.

If taken individually, the Hopf oscillator described in Eq. 3.2 represents the simplest form containing a bifurcation from a fixed point to a limit cycle attractor as the parameter λ is varied. When $\lambda < 0$, the system has a globally attractive fixed point at $(r + x_{\text{init}}, 0)$. A supercritical Hopf bifurcation occurs when $\lambda > 0$. The bifurcation destabilizes the fixed point and leads to a system with a

stable periodic solution (attractor limit cycle), a sine of amplitude $r = \sqrt{\lambda}$ and frequency ω. If it happens that the system is initially at the unstable fixed point when the bifurcation occurs, a noise term is generally added to the dynamics to guarantee the escape from the unstable state and the convergence to the attracting limit cycle. In our case, the special arrangements in the dynamics ensure that the system is always in an attractor state in both operational regimes. Thus, the noise term in Eq. 3.1 is an extra security measure.

3.2 Temporal stabilization of movements

Humans generate flexible sequences of movements that are highly adaptive. Moreover, these actions tend to have a stable temporal structure. Indeed, experimental evidences showed that the generated movements take place in time and compensate for perturbations by either accelerating or decelerating the movement so as to restore as much as possible the overall movement time (Schöner, 2002; Bootsma and Van Wieringen, 1990).

The timed movement dynamics defined in Eq. 3.1 permits to generate discrete timed movements by means of a half cycle of the Hopf oscillation as illustrated in Figure 3.1 (top plots). The major property of a limit cycle attractor is that if a short-time perturbation occurs, the system will resume to the limit cycle afterwards, as depicted on Figure 3.1 (bottom plots).

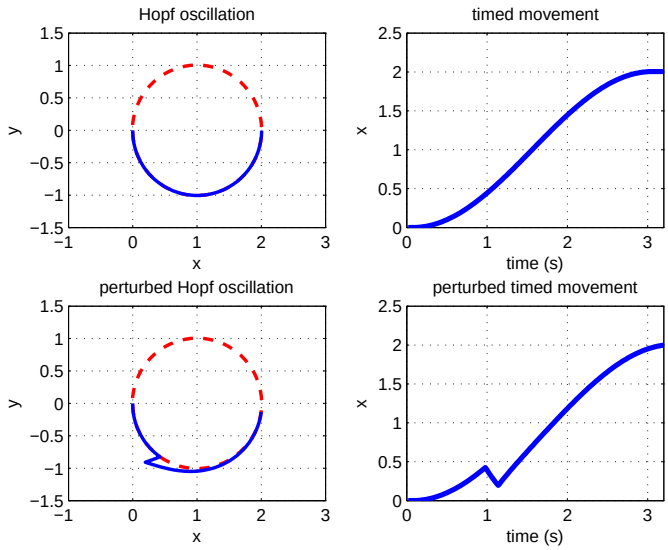


Figure 3.1: Generating a discrete timed movement by means of a half cycle of the Hopf oscillation. The plots in the top panels illustrate a discrete timed movement resulting from integrating the Hopf dynamics during a half cycle (here $3s$). When a short perturbation occurred at about one third of the movement time, the state of the system resumed quickly to the limit cycle as shown in the bottom panels.

Furthermore, the Hopf amplitude can be modulated online by directly changing the oscillator radius r in Eq. 3.2. Such feature allows us to very easily, and

smoothly, modulate the system's behavior according to the desired trajectory output.

In a robotic application, this feature is necessary given that the target location x_{target} of a timed movement is usually a varying prediction based on low level and often noisy sensory information. However, doing so will introduce a perturbation in the oscillator phase causing a permanent phase shift of the signal compared to the unperturbed system. In our case, this is translated by a non-zero velocity at the target point implying an incorrect timing of the movement as shown in Figure 3.2. Here, the timed movement defined by x expresses the position of an effector moving toward an initial target $x_{\text{target}} = 2$ in a total time of $3s$. At about two thirds of the movement cycle, the target is moved away to $x_{\text{target}} = 3$. The oscillator accelerated the movement velocity $\dot{x}$ but failed to compensate for the perturbation and reach the new target in time.

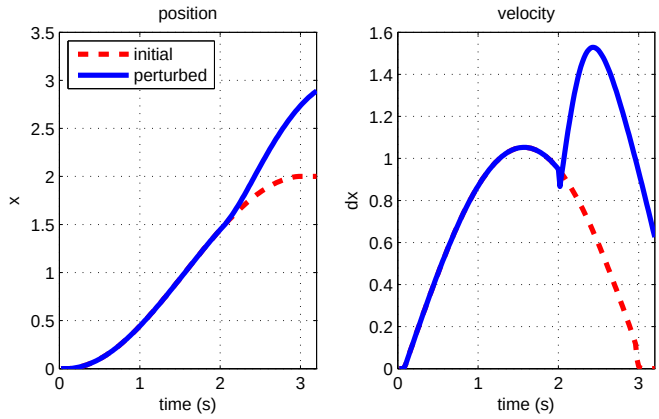


Figure 3.2: A perturbation during a discrete timed movement. The timed movement is perturbed at about two thirds of the cycle by moving away the target. As a result, the perturbed system accelerates the movement as compared to the initial one but fails to compensate for the perturbation and reach the new target leading to an incorrect timing.

To overcome this and to ensure the correct timing, the movement must be updating while it is in execution. This can be done in two different ways : (a) modulating the frequency of the oscillator, (b) modulating the movement dynamics, so as to compensate for perturbations by accelerating or decelerating the movement accordingly while keeping the total movement time stable.

3.2.1 Modulating the oscillator frequency

Modulating the oscillator frequency implies updating the cycle time of the movement. In this work, I developed an update rule for the oscillator half cycle time $T_{\text{hcycle}} = \frac{T}{2}$ in Eq. 3.2 during the timed movement, i.e., during half time of the limit cycle. The update rule couples the movement dynamics of the end-effector with perceptual information about the moving object. For instance, in a ball interception task, the perceptual information represent a predicted interception point of the ball that defines the target of the timed movement x_{target}, and a prediction of the time needed by the ball to reach this point specified by the

time-to-impact b_{tim}. Based on these perceptual information, T_{hcycle} is set to satisfy

$$T_{\text{hcycle}} = \frac{2\,r\,b_{\text{tim}}}{d(t)}, \tag{3.3}$$

where $d(t) = |x_{\text{target}} - x|$ is the currently measured remaining distance to the target.

The update rule assumes a linear relationship between the time remaining and the distance to be covered in order to reach the movement goal. This formulation stabilizes the total movement time and ensures a synchronization between the end-effector and the moving ball.

Figures 3.3 and 3.4 show different simulations of the update mechanism for a fixed movement time $MT = 3s$. In both figures, three different realizations by integrating the dynamics in Eq. 3.1 are displayed each as a cycle time T_{hcycle}, position x, and velocity $\dot{x}$. The switching between the dynamic regimes are performed using a competitive dynamics of the form described in Section 2.2.2. In both figures, the first realization generates a timed movement from an initial state $x_{\text{init}} = 0$ toward a target location $x_{\text{target}} = 2$, the second realization depicts a timed movement from $x_{\text{init}} = 0$ to $x_{\text{target}} = 3$, and the third realization starts from the same initial state $x_{\text{init}} = 0$ toward the first target $x_{\text{target}} = 2$.

In Figure 3.3, the target of the third realization is moved instantaneously to the second target $x_{\text{target}} = 3$ at about two thirds of the cycle. In Figure 3.4, the target of the third realization is moved gradually away to the second target $x_{\text{target}} = 3$ at about one third of the cycle. In both cases, the update mechanism produces a velocity curve that accelerates the movement to cover the larger distance in the same time $MT = 3s$ and reach the target in time.

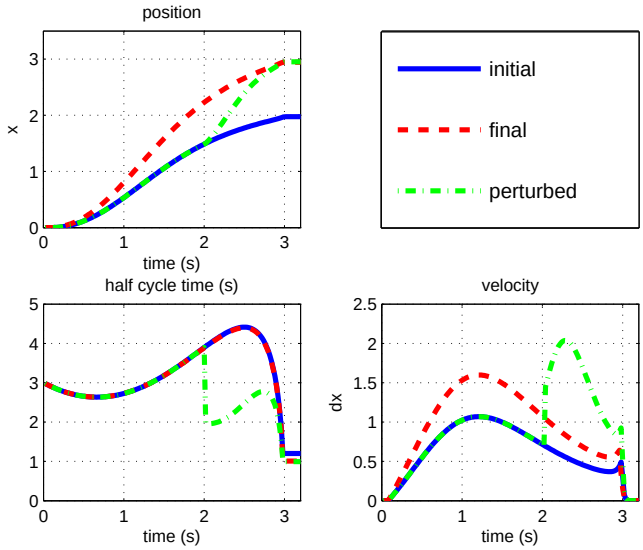


Figure 3.3: Modulating the oscillator frequency to accelerate the movement and compensate for instantaneous perturbation. Three other realizations of the timed movement dynamics in Eq. 3.1 using the proposed update rule are displayed. The initial and final systems share the same starting state but reach for different targets. The perturbed system starts with the same configuration as the initial system. At about two thirds of the cycle, the target of the movement is instantaneously increased to match the final system. The perturbation induces an acceleration and the perturbed system reaches the target in time.

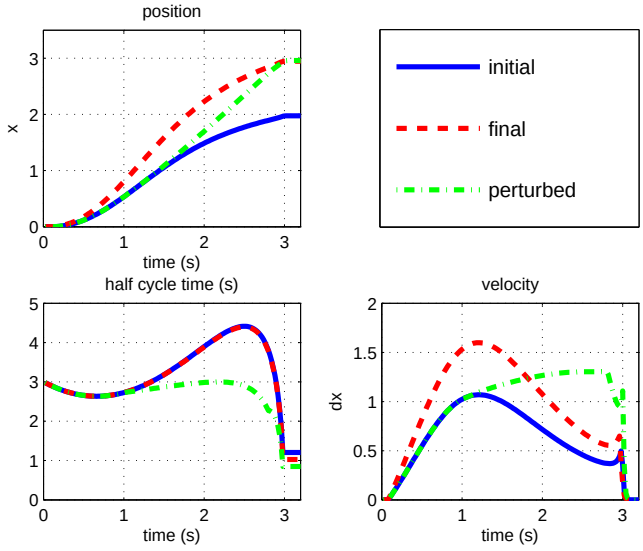


Figure 3.4: Modulating the oscillator frequency to accelerate the movement and compensate for gradual perturbation. Three different realizations of the timed movement dynamics in Eq. 3.1 using the proposed update rule are displayed. The target of the perturbed system is moved gradually at about one third of the cycle to match the final system. Here also, the updating dynamics generates the required velocity profile to reach the new target in time.

The update mechanism is also able to handle situations where the target is displaced toward the end-effector during movement. In Figure 3.5, the first realization generates a timed movement from an initial state $x_{\text{init}} = 0$ toward a target location $x_{\text{target}} = 2.5$, the second realization shows a timed movement from $x_{\text{init}} = 0$ to $x_{\text{target}} = 3$, and the third realization starts from the same initial state $x_{\text{init}} = 0$ toward the second target $x_{\text{target}} = 3$. At about two thirds of the cycle, the target of the third realization is moved down to match the first target $x_{\text{target}} = 2.5$. Consequently, the mechanism decelerates the movement to cover the smaller distance in the same time $MT = 3s$ and reach the target in time.

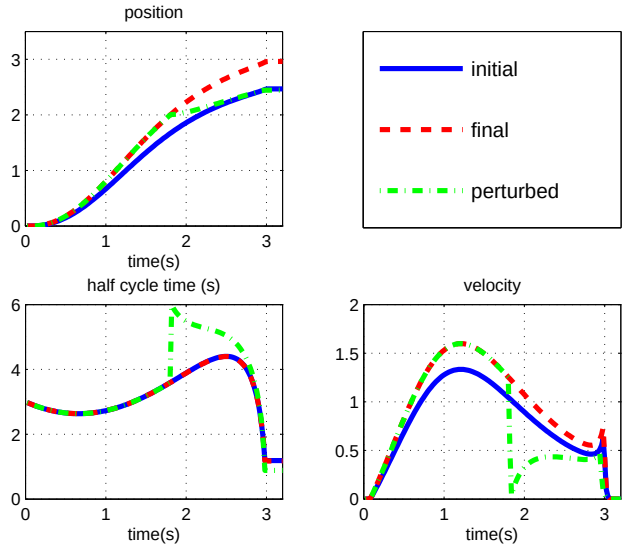


Figure 3.5: Modulating the oscillator frequency to decelerate the movement and compensate for instantaneous perturbation. Three different realizations of the timed movement dynamics in Eq. 3.1 using the proposed update rule are displayed. The target of the perturbed system is moved instantaneously down at about two thirds of the cycle to match the initial system. Here, the updating dynamics generates the required deceleration to reach the new target in time.

The update rule allows a fast and precise adaptation of the movements in presence of perturbations. Nevertheless, the update mechanism is not perfect. As can be seen in the figures, peaks of speed at the end of the movements can be observed. These undesired accelerations are due to the use of the time-to-impact variable in the rule formula. Furthermore, a division by zero can occur if the target is exactly reached before stopping the oscillation. These effects can be minimized by a behavioral organization that switches off the oscillatory regime when the end-effector reaches a predefined and small threshold distance to the target.

3.2.2 Modulating the movement dynamics

A discrete analogue of frequency locking between two coupled timing dynamics is proposed by Schöner and Santos (2001). The system permits a temporal coordination of two robot arms and synchronized movements even if the movement

onsets are not perfectly simultaneous. The coupling terms modify the Hopf contributions in both timing dynamics. A complete analysis of these properties can be found in (Schöner, 1990).

Following the same principle, an update mechanism can be realized by coupling the Hopf contribution in both dimensions (x , y) to their respective analytical solutions (x' , y') given by

$$x' = r(1 - \cos(\omega t)), \tag{3.4}$$

$$y' = -r \sin(\omega t). \tag{3.5}$$

In this case, the movement cycle time specified by T_{hcycle} remains unchanged. The coupling affects the Hopf contribution in Eq. 3.1 as follows

$$\tau \begin{pmatrix} \dot{x} \\ \dot{y} \end{pmatrix} = ... + c_{\text{hopf}} H(x, y) + c_{\text{hopf}}\, c_{\text{w}} \begin{pmatrix} x' - x \\ y' - y \end{pmatrix} + ..., \tag{3.6}$$

where c_{w} is the coupling strength. The coupling term is multiplied by the neuron c_{hopf} to ensure that the coupling is effective only when the Hopf contribution is activated. The proposed update mechanism permits to synchronize the Hopf contribution in x and y dimensions with their analytical solutions. The perturbations in the target point will modulate the movement dynamics and generate the velocity curves necessary to compensate and stabilize the movement time.

Figures 3.6, 3.7 and 3.8 illustrate the performance of the update rule in similar experiments as those exposed in Section 3.2.1.

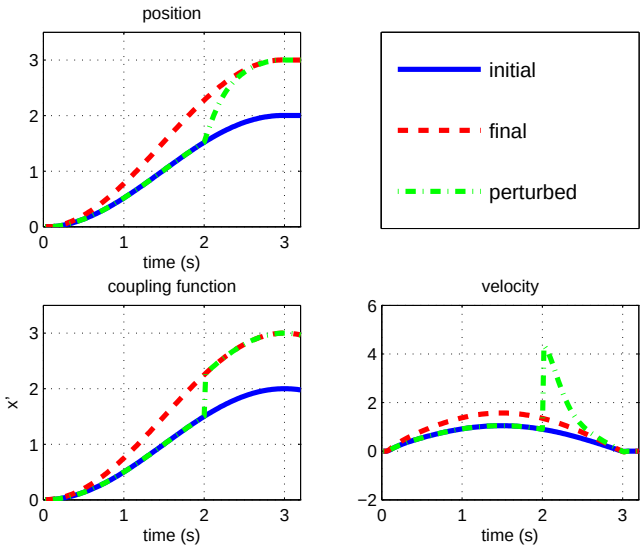


Figure 3.6: Modulating the movement dynamics to accelerate the movement and compensate for instantaneous perturbation. Performance of the update mechanism when the target of the perturbed system is moved instantanously to match the final system. The perturbed system accelerates the movement and manages to reach the new target in time.

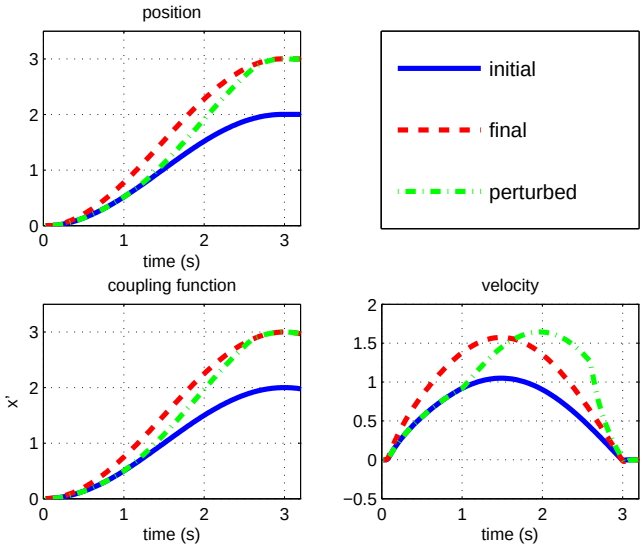


Figure 3.7: Modulating the movement dynamics to accelerate the movement and compensate for gradual perturbation. Performance of the update mechanism when the target of the perturbed system is moved gradually to match the final system. Here also, the perturbed system accelerates the movement and manages to reach the new target in time.

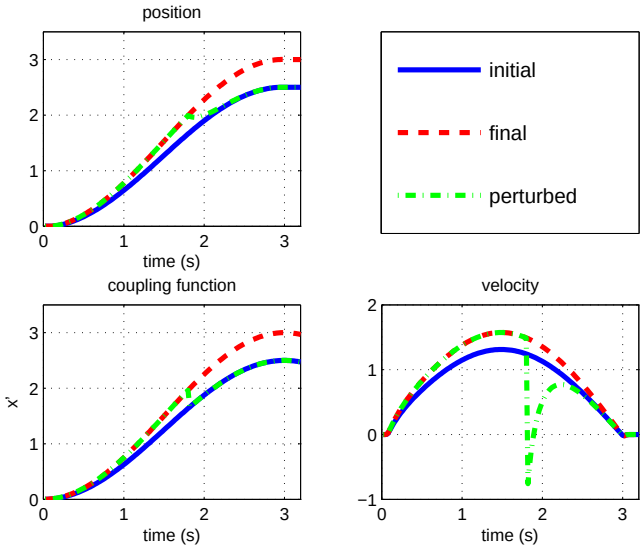


Figure 3.8: Modulating the movement dynamics to decelerate the movement and compensate for instantaneous perturbation. Performance of the update mechanism when the target of the perturbed system is moved instantanously to match the initial system. Here, the perturbed system decelerates the movement and manages to reach the new target while keeping the movement time invariant.

The update rule adapts the movements in presence of perturbations and permits a stabilization of the movement time. However, integrating the update mechanism inside the timing dynamics requires a timer for the analytical functions that should be be started each time the oscillatory regime is activated. A

necessary disposition to obtain the required solution profiles in coordination with the oscillator output.

In the present work, I opted for modulating the oscillator frequency as an update mechanism for the movements. This choice is mainly motivated by the need of a fast and precise mechanism to compensate for perturbations when high speed movements have to be generated as is the case for the robotic tasks considered in this project.

3.3 Organization of timed behaviors

In order to initiate and terminate the different timed movements at appropriate instants in time, a form of behavioral organization is required. For that purpose, a neural dynamics architecture is built upon the DFT based framework for behavioral organization previously introduced in Section 2.2.2.

The neural architecture is organized hierarchically into two levels of abstraction. The upper level models the different timed behaviors that compose a robotic task in the form of *higher level* EBs and encodes their sequencing rules. To initiate or terminate a timed movement, a higher level EB controls a lower level *movement module* that interacts directly with the movement dynamics described in Section 3.1. A movement module is organized horizontally into three EBs and, when activated, permits to (1) update the initial state before starting a movement. (2) initiate a timed movement for an end-effector toward a target location by switching the dynamics to the oscillatory regime. (3) stabilize the end-effector at a postural state after executing the movement.

3.3.1 Movement module

To generate a timed behavior, the movement module (see Figure 3.9) must be activated by a higher level EB. The two dynamic regimes described in Section 3.1 express two behaviors (a) 'move' EB where the end-effector executes the timed trajectory toward a target location x_{target} during the oscillatory regime (b) 'fix' EB stabilizes the end-effector at a postural state after the movement execution. Furthermore, the initial state x_{init} needs to be updated before starting the movement. The update process is performed by an 'update' EB that allows to start the behavior from the current 'real' end-effector state x_{cur} (read from the hardware sensory) and so, initiate the timed movement from any position along the movement dimension.

To ensure that an initial state update happens before a movement starts and that the end-effector stabilizes at a postural state after a movement, behavioral constraints are set between the movement module EBs.

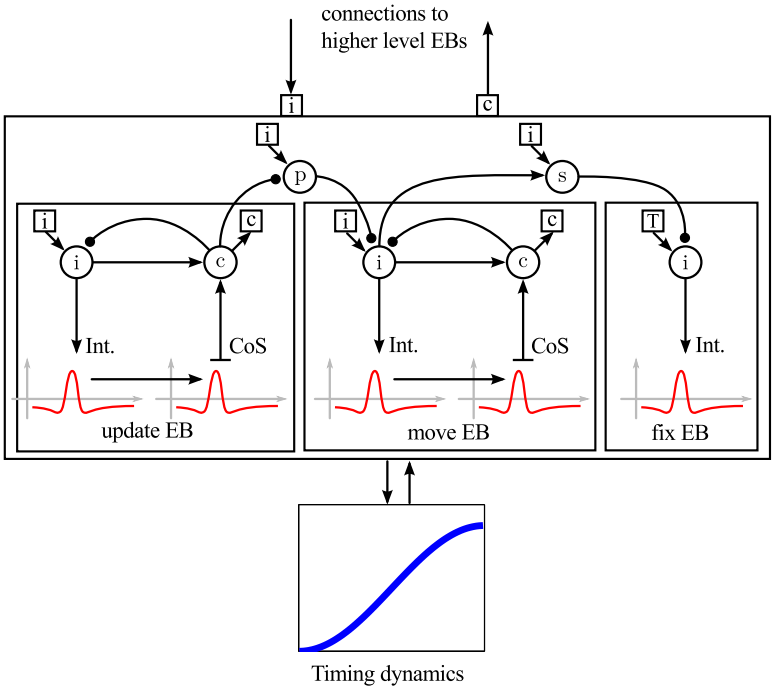


Figure 3.9: The movement module used to generate the timed behaviors. The 'update' and 'move' EBs can be activated or deactivated through the colored intention 'i' and CoS 'c' nodes by a higher level EB while the 'fix' EB is always activated by the task input 'T'. Behavioral constraints are set between the EBs through a suppression 's' and precondition 'p' nodes.

'update' EB

The 'update' EB permits to update the initial movement state x_{init} before starting the timed behavior. When the intention node of the 'update' EB gets activated, a sub-threshold boost is provided to the EB intention field and the CoS node. Combined with a Gaussian input centered at the current robot state x_{cur}, the intention field forms a localized peak that is propagated as a sub-threshold activation to the EB CoS field. The perceptual input to the intention field represents the target of the update action and is given by the Gaussian

$$S^{\text{up}}_{\text{perc,int}}(x,t) = c^{\text{up}}_{\text{perc,int}} \exp\left(-\frac{(x_{\text{cur}} - x')^2}{2\sigma^2}\right), \tag{3.7}$$

where the weight $c^{\text{up}}_{\text{perc,int}}$ sets the strength of the input and σ defines the range of the excitation. Simultaneously, the intention node output $f(u_{\text{i}}^{\text{up}}(t)) = c_{\text{up}} \approx 1$ (i.e., the sigmoided activation output) turns on a dynamical system with a fixed point attractor set at the state x_{cur} to update the initial state x_{init}

$$\dot{x}_{\text{init}} = -c_{\text{up}}\, a(x_{\text{init}} - x_{\text{cur}}), \tag{3.8}$$

where $a > 0$ sets the attraction strength. In addition to the input coming from the intention field, the CoS field receives the variable x_{init} as a sub-threshold Gaussian input defined by

$$S^{\text{up}}_{\text{perc,CoS}}(x,t) = c^{\text{up}}_{\text{perc,CoS}} \exp\left(-\frac{(x_{\text{init}} - x')^2}{2\sigma^2}\right), \tag{3.9}$$

with a strength set by $c^{\text{up}}_{\text{perc,CoS}}$ and a range specified by σ. When the two inputs overlap, the update process is completed and a peak activation forms. The CoS node of the EB acts as a peak detector, gets activated, and inhibits the EB intention node. Since the peak in the CoS field will not persist when the timed movement starts and changes the current state x_{cur}, the CoS node is self-sustained and stays active until the next update process for a new movement. The mathematical formulation of the EB follows the general mathematical description of an EB given by the DFT based framework for behavioral organization in Section 2.2.2.

'move' EB

After updating x_{init}, the 'move' EB intention node gets activated and a homogeneous boost is provided to the EB intention field and to the CoS node. With a sub-threshold activation centered at the target location x_{target}, the intention field forms a localized peak activation that is propagated to EB CoS field. The EB intention field $u^{\text{mv}}_{\text{i}}(x,t)$ receives the target spatial position either directly from the perceptual field (introduced further) or from a separate update module where dynamical fields encode the movement targets as in the case where a movement module is shared by more than one higher level EB.

The target location is supplied to the timed movement dynamics in Eq. 3.1 (see Section 3.1) by extracting the peak position on the intention field using

$$\dot{x}_{\text{target}} = -\left(\int f(u^{\text{mv}}_{\text{i}}(x,t))dx\right) x_{\text{target}} + \int x\, f(u^{\text{mv}}_{\text{i}}(x,t))dx, \tag{3.10}$$

where $f(.)$ is a sigmoid function. In Eq. 3.10, the field activation is considered as a probability distribution where the mean is mapped into an attractor, a more detailed explanation can be found in (Zibner et al., 2011).

At the same time, the intention node output $f(u^{\text{mv}}_{\text{i}}(t)) = c_{\text{hopf}} \approx 1$ (while $c_{\text{post}} \approx 0$) switches the movements dynamics from the postural to the oscillatory regime and lets the timing variable x evolve along a harmonic trajectory from x_{init} toward x_{target}.

Combined with the activation coming from the intention field, the CoS field receives the timing variable x as a Gaussian shaped input and forms a peak activation when the two inputs overlap after reaching the target. The EB CoS node turns on and inhibits the intention node. The mathematical formulation of the EB follows the general mathematical description of an EB given by the DFT based framework for behavioral organization in Section 2.2.2.

'fix' EB

After completing the timed movement, the 'fix' EB intention node gets activated and switches the dynamics in Eq. 3.1 back to the postural regime $f(u^{\text{fix}}_{\text{i}}(t)) = c_{\text{post}} \approx 1$ (while $c_{\text{hopf}} \approx 0$) which stabilizes the end-effector at the postural state x_{post}. Simultaneously, a homogeneous boost is provided to the EB intention field that forms a localized peak activation centered at the final (reached) position. The EB does not have a CoS node or field since it characterizes a postural state.

The mathematical formulation of the EB follows the general mathematical description of an EB given by the DFT based framework for behavioral organization in Section 2.2.2, but without the CoS part.

Behavioral constraints

Behavioral constraints between the movement module EBs are necessary to ensure the correct execution order (see Figure 3.9). A precondition constraint, modeled as a precondition node, between the 'update' and 'move' EBs imposes that an update of the movement initial state x_{init} is always performed before the movement starts. Similarly, a suppression constraint, in the form of a suppression node, prevents the 'fix' EB from becoming active while the 'move' EB is being executed.

While the precondition node is initially activated and gets inhibited after the update process by the 'update' EB CoS node, the suppression node is turned on by the 'move' EB intention node until the timed movement is terminated. The mathematical formulation of the dynamical nodes follows the general mathematical description given by the DFT based framework for behavioral organization in Section 2.2.2 of the nodes that compose an EB.

3.3.2 Sequential organization of timed behaviors

Within the neural dynamics architecture, the timed behaviors that compose a robotic task are modeled by higher level EBs. To initiate or terminate a timed movement, the higher level EB needs to activate or deactivate a lower level movement module (respectively).

To activate a movement module, the higher level EB intention node turns on and activates, via supra-threshold inputs, the intention nodes of the 'update' and 'move' EBs as well as the precondition constraint while only sub-threshold boosts are given to their CoS nodes, the suppression constraint, and to its own CoS node. As soon as the precondition node is active, it inhibits the 'move' EB intention node and prevents it from being active while the 'update' EB is being executed. Once the initial movement state is updated, the CoS node of the 'update' EB gets activated and inhibits the precondition node. The inhibited precondition node releases its inhibition from the 'move' EB intention node. The latter turns on and starts the timed movement. Simultaneously, the 'move' EB intention node activates the suppression constraint which, in turn, inhibits the 'fix' EB intention node to release the fixation dynamics. The fact that the suppression node is already boosted by the higher level EB intention node reduces the possibility to have a simultaneous active state of the two dynamic regimes and ensures a fast switch. In the architecture, the 'fix' EB intention node is active by default through excitatory input coming from the task node, introduced further. This configuration ensures that the movement dynamics in Eq. 3.1 is either in the postural or oscillatory regime during movement generation.

On the other hand, the CoS node of the higher level EB receives sub-threshold inputs from the CoS nodes of the 'update' and 'move' EBs so that the former gets activated only if both the update and movement are completed. As soon as the CoS node turns on, the intention node of the higher level EB gets inhibited

and consequently ceases its inputs to the movement module which gets deactivated as well. This cascade of instabilities indicates that the timed movement is completed. Additionally, the same neural connections permit to abort a timed movement by inhibiting the higher level EB intention node through, for example, a perceptual constraint about the robotic task. When the movement module is turned off, the suppression constraint is released and the 'fix' EB gets activated again, switching the dynamics back to the postural regime. The higher level EB does not have a direct linkage to the sensory-motor system like a traditional EB but is meant to control implicitly the generation of movements. Therefore, the intention and CoS fields are omitted in the EB definition but the rest of the mathematical formulation follows the general mathematical description of an EB given by the DFT based framework for behavioral organization in Section 2.2.2.

For the robotic catching task considered in Section 3.4, two timed behaviors are defined. A 'move to catch' EB starts a timed movement toward the predicted catching point to intercept a free falling ball and a 'move to base' EB brings the end-effector back to a base station after executing the task (see Figure 3.11).

Offset suppression nodes

The fact that the CoS node of the higher level EB is not self sustained and turns off as soon as the timed behavior is executed and the movement module is deactivated, will lead to reactivate the EB intention node. This may result in an undesirable oscillation between the active and inactive state of both the intention and CoS nodes of the higher level EB. To prevent such a situation, an offset suppression node is used to keep the intention node inhibited when the target location is reached. The offset suppression node plays the role of a memory of the accomplished movement and gets activated as soon as the target location is approached with a predefined small offset. The mathematical formulation of the offset suppression nodes follows the general mathematical description given by the DFT based framework for behavioral organization in Section 2.2.2 of the nodes that compose an EB.

Sequential constraints

The generated timed behaviors are coordinated and organized in time by setting sequential constraints between the higher level EBs. A suppression constraint in the form of a suppression dynamical node is used to prevent the simultaneous activation of two higher level EBs. The suppression node is activated by the intention node of the first EB and provides an inhibitory input to the intention node of the second EB. The use of such a constraint provides a form of flexibility to the architecture and permits to organize sequentially the different timed behaviors that compose a robotic task. Moreover, if two timed behaviors are to be executed sequentially along the same movement dimension, a movement module can be shared by two higher level EBs.

Since the robotic catching task is executed along a single axis, the 'move to catch' and 'move to base' EBs share a single movement module to control the end-effector timed movements as shown in Figure 3.11.

3.3.3 Task input

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 task and provides activation inputs to all intention nodes of the higher level EBs as well as to the nodes implementing the sequential constraints. Furthermore, the task node activates by default the 'fix' EB in the movement module to make sure that the dynamics is in the postural regime unless a movement is initiated. A mathematical description of a dynamical task node is given by the DFT based framework for behavioral organization in Section 2.2.2.

3.3.4 Perceptual and Motor systems

The neural dynamics architecture is completely autonomous in the sense that all timed behaviors are initiated or terminated through online coupling to sensory information about the task.

In the case of the robotic catching task, perceptual information about the free falling ball permits to initiate the catching and backward movements. The coupling is achieved by introducing a perceptual constraint in the form of a precondition dynamical node that is active by default and inhibits the intention node of the 'move to catch' EB. At the same time, the precondition node receives inhibitory input from a dynamical neural field that is used to encode a perceptual representation of the environment as well as conditions related to the task and is referred here as the perceptual field. The perceptual field is defined over the movement metric dimension and forms a peak activation localized at the predicted catching point whenever the ball is detected to be falling inside the robot's reachable region. The peak activation inhibits the perceptual precondition which activates the 'move to catch' EB intention node to start the catching movement. If the ball is successfully caught or missed, the ball is no longer detected by the visual system and the perceptual field looses its peak activation. Consequently, the perceptual precondition reactivates and inhibits the 'move to catch' EB intention node which releases the 'move to base' EB intention node to start the backward movement toward the base.

In this context, the visual system is used to algorithmically per-process perceptual information about the task. Building a dynamical or neurally plausible visual system is not considered here and is beyond the scope of the present work.

On the motor side, the trajectories of Eq. 3.1 are generated in task space before being converted into joint angles using an inverse kinematics transformation and fed to the robotic arm for execution.

3.4 Robotic catching simulation

In a Matlab simulation, the simplified catching task is performed by a two degrees of freedom (DoFs) manipulator on a free falling ball as depicted in Figure 3.10. The robot's end-effector emulates a catching container or a basket. To detect and predict the ball motion, I assume a visual system that updates perceptual variables consisting of a prediction of the catching point and an estimation of the ball reachability criterion $b_{\text{catchable}} \in [-1, 1]$.

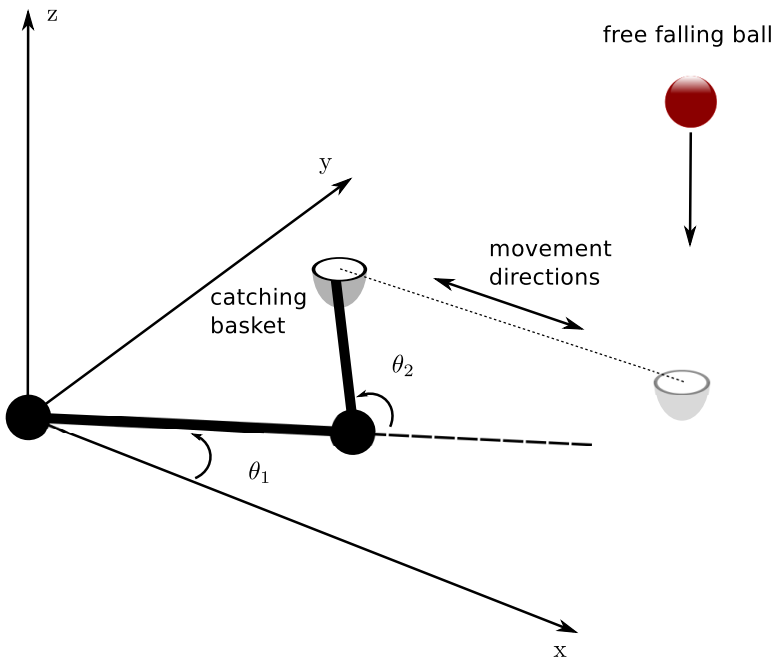


Figure 3.10: The simulated catching task performed by a two degrees of freedom manipulator on a free falling ball.

The task consists of initiating a timed movement from a base station toward the predicted catching point as soon as the ball is detected to be catchable ($b_{\text{catchable}} = 1$). When the ball is successfully caught or missed ($b_{\text{catchable}} = -1$), it is no more detected by the visual system and the end-effector returns back to the base station ready to execute a new catching movement whenever indicated. The neural dynamics architecture shown in Figure 3.11 generates the task movements. Moreover, these timed movements are autonomously sequenced and can be, at any time, aborted or re-initiated in response to any sudden change in the sensory information of the ball trajectory.

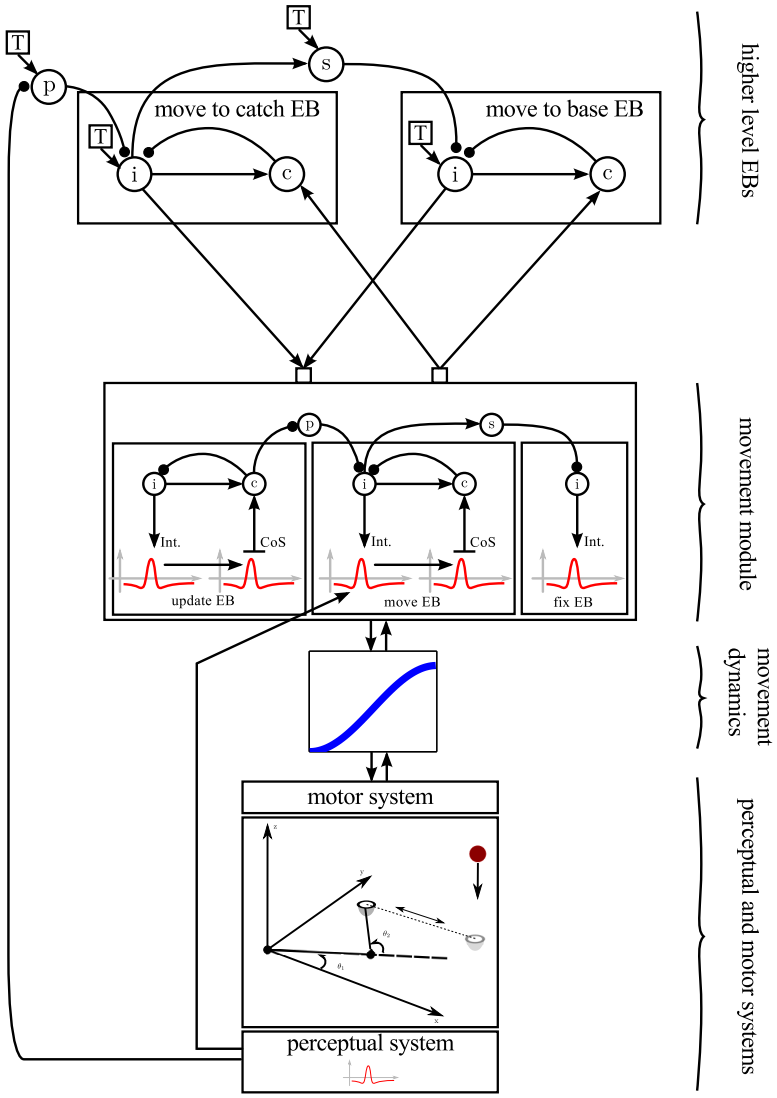


Figure 3.11: The neural dynamics architecture used to generate the sequences of timed movements for the robotic catching task. The higher level part encloses the the 'move to catch' and 'move to base' EBs which share the control of a lower level movement module. The movement dynamics gets activated by the movement module and permits to feed the robot manipulator with the timed trajectories. The perceptual system encodes a representation of the task environment and ensures the system autonomy. To prevent the higher level EBs to reactivate, offset suppression nodes are used to inhibit the EBs intention nodes when their respective target locations are reached. To preserve the clarity of the architecture figure, the offset suppression nodes are omitted.

The two timed behaviors are modeled by two higher level EBs, 'move to catch' and 'move to base' EBs, that share a single lower level movement module to control the end-effector movements along the x-axis of a world reference frame. The prediction of the catching point and the catching condition are encoded into a one-dimensional perceptual neural field defined over the movement dimension that spans on the range between [0.2 m,0.6 m]. The perceptual field receives a sub-threshold Gaussian input centered at the predicted caTo preserve the clarity of the architecture figures, the offset suppression nodes are omitted.tching point,

in addition to a sub-threshold input expressing the catching condition delivered by the parameter $b_{\text{catchable}}$. These perceptual information are propagated to the movement module (more precisely to the intention field of the 'move' EB) via an external update module that encodes both the catching and the base locations (not shown in Figure 3.11). The ball is launched from a height of 2 m and accelerates in a free fall toward the catching point. The end-effector dynamics is governed by Eq. 3.1 (see Section 3.1) along the movement dimension. The oscillator cycle time is updated using Eq. 3.3 given a prediction of the ball *time-to-impact* b_{tim} by the visual system. Since the term *time-to-contact* is often used to designate a time estimate based on the assumption of constant velocity, I use the term *time-to-impact* to express more generally the time up to contact independent of its prediction method.

3.5 Results

The following experiments illustrate the core features of the model. The results show how the system was able to generate and flexibly organize the timed movement sequences that compose the robotic task. Moreover, external perturbations in the ball trajectory permit to evaluate the performance of the dynamical model in adapting and updating the movement sequences.

To produce the present results, the solutions of the continuous time dynamics are approximated numerically using the Euler method.

3.5.1 Successful execution of a catching task

The first experiment consists of performing a successful catch (Figure 3.12). This experiment demonstrates how the dynamical model allowed the emergence of the necessary sequence of behaviors to autonomously execute the timed interception. In the following, I will present in details the most relevant results obtained from this experiment.

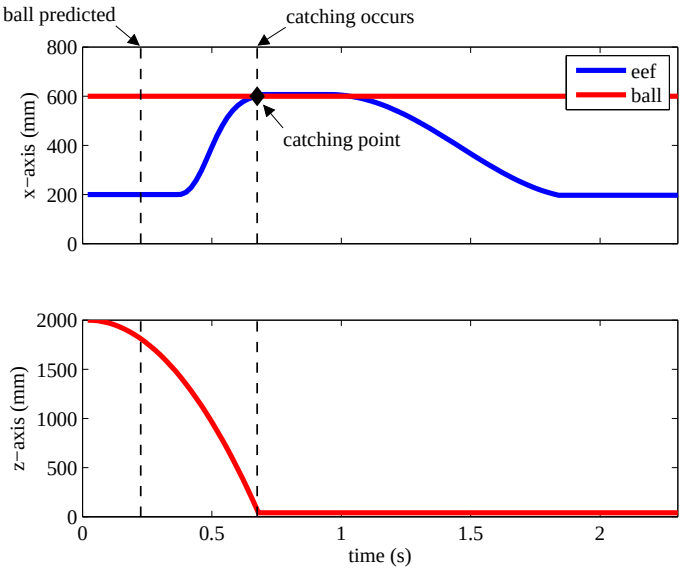


Figure 3.12: The ball and end-effector trajectories during a successful catch. The vertical dotted lines indicate ball trajectory events.

The ball is launched at time $t = 0\,\text{s}$. When the ball is detected and predicted to fall inside the robot's catching region ($b_{\text{catchable}} = 1$) at $t \approx 0.225\,\text{s}$, a localized peak activation centered at the predicted catching point forms in the perceptual field (Figure 3.13). All events related to the ball trajectory are indicated in the result figures by vertical dotted lines as they occur during the experiments.

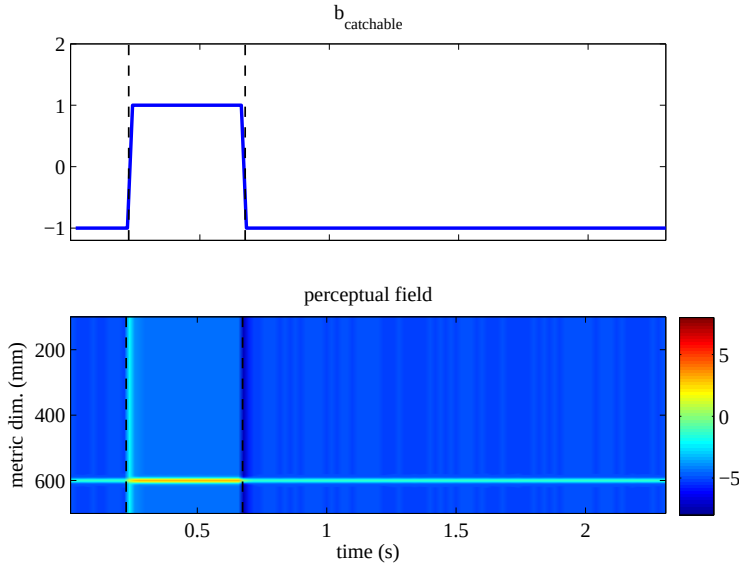


Figure 3.13: The trajectory of the perceptual parameter $b_{\text{catchable}}$ and the time course of activation of the perceptual field during a successful catch. The color-bar in the bottom right corner shows the color map used for the plot of the dynamical neural field activation.

To produce the plot over time shown in Figure 3.13 (bottom), a color map is used to show the field activation along the metric dimension as it unfolds in time. The color-bar expresses the field activation level.

As shown in Figure 3.14, the peak activation inhibits the perceptual precondition ('mtc perc') which in turn releases its inhibition from the 'move to catch' (abbreviated by 'mtc') EB intention node ('mtc int').

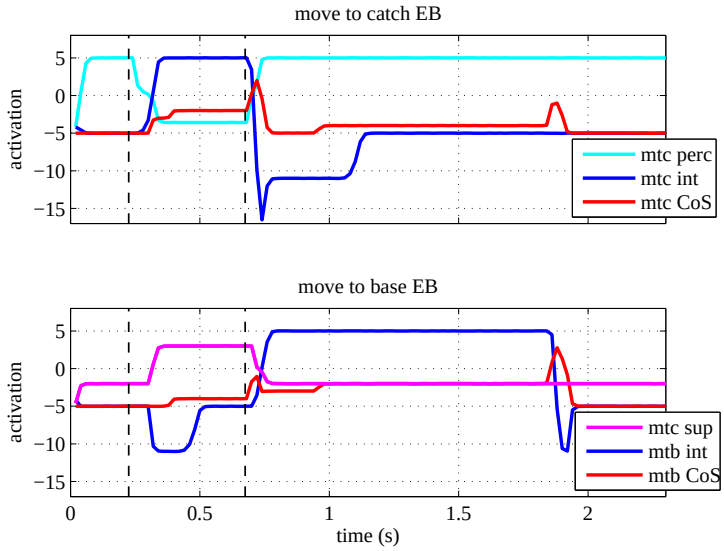


Figure 3.14: The time course of activation of the higher level EBs intention and CoS nodes and the dynamical nodes implementing the perceptual and sequential constraints during a successful catch.

At $t \approx 0.34\,\mathrm{s}$, 'mtc int' turns on and activates the movement module to generate the timed behavior of the end-effector toward the catching point. Simultaneously, 'mtc int' activates the suppression constraint 'mtc supp' that inhibits the 'move to base (mtb)' EB intention node ('mtb int') and prevents it from being active. After reaching the target, the 'move to catch' EB CoS node ('mtc CoS') gets activated and inhibits the intention node 'mtc int'. Additionally, as soon as the task is executed and the ball is caught ($b_{\mathrm{catchable}} = -1$) at $t \approx 0.675\,\mathrm{s}$, the perceptual field looses its activation which brings back the perceptual precondition 'mtc perc' to the active state. Consequently, the activated 'mtc perc' inhibits the intention node 'mtc int' which deactivates the movement module. The deactivated 'mtc int' turns off the suppression node 'mtc sup' and leads to activate the 'mtb int' intention node. At $t \approx 0.78\,\mathrm{s}$, the intention node 'mtb int' turns on and reactivates the movement module to bring the end-effector back to the base in a per-specified and fixed time. When the base location is reached, the 'move to base' EB CoS node ('mtb CoS') turns on and stops the movement by inhibiting the intention node 'mtb int' which deactivates the movement module at $t \approx 1.87\,\mathrm{s}$.

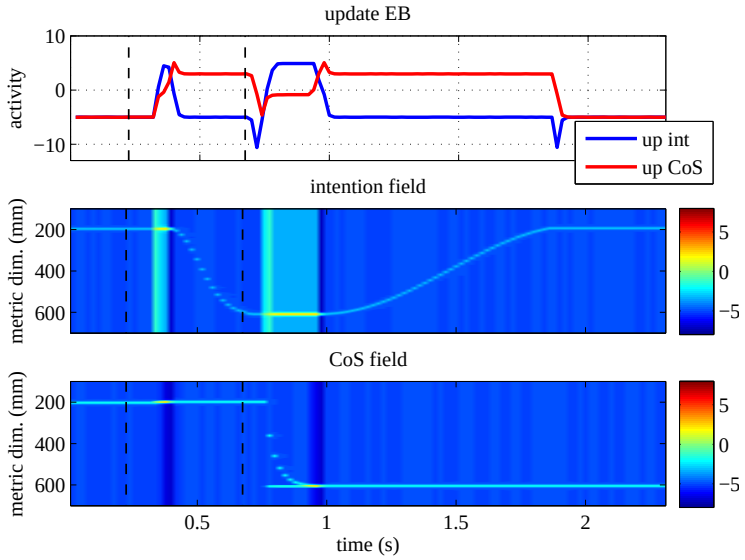


Figure 3.15: The time course of activation of the 'update' EB when executed before each movement initiation during a successful catch.

Before starting each timed behavior (here, from the base to the catching point and back to the base), it is necessary to update the initial state of the movement as shown in Figure 3.15. Every time when the movement module gets activated (at $t \approx 0.36\,\mathrm{s}$ and $0.8\,\mathrm{s}$), the 'update' EB intention node ('up int') turns on ($c_{\mathrm{up}} = 1$) and activates Eq. 3.8 which starts updating the initial movement state. The update process is completed when a peak activation forms in the EB CoS field which switches on the EB CoS node ('up CoS'). 'up CoS' is self-sustained and inhibits 'up int' leading to terminate the update process. Note that the update duration can be tuned and vary according to the metric distance between the current end-effector state and the movement starting position.

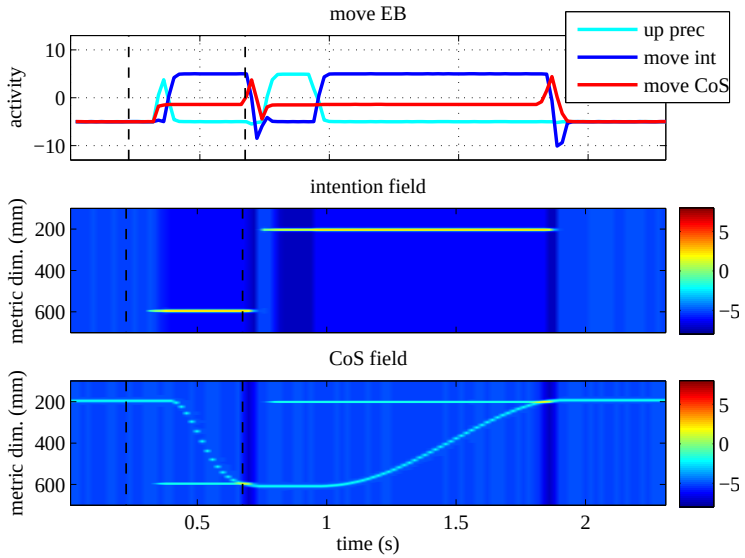


Figure 3.16: The time course of activation of the 'move' EB and the update precondition node 'up prec' during the execution of a successful catch.

After updating the initial movement state, the two timed movements can be initiated at $t \approx 0.42\,\mathrm{s}$ and $0.98\,\mathrm{s}$. The active 'up CoS' node inhibits the update precondition node ('up prec') and permits to activate the 'move' EB intention node ('move int'). The 'move int' output ($c_{\mathrm{hopf}} = 1$) switches the movements dynamics in Eq. 3.1 from the postural to the oscillatory regime and starts the timed movement (Figure 3.16). Simultaneously, the active 'move int' node provides excitatory input to the suppression node ('move sup') (Figure 3.17) which gets activated and suppresses the 'fix' EB intention node ('fix int')($c_{\mathrm{post}} = 0$).

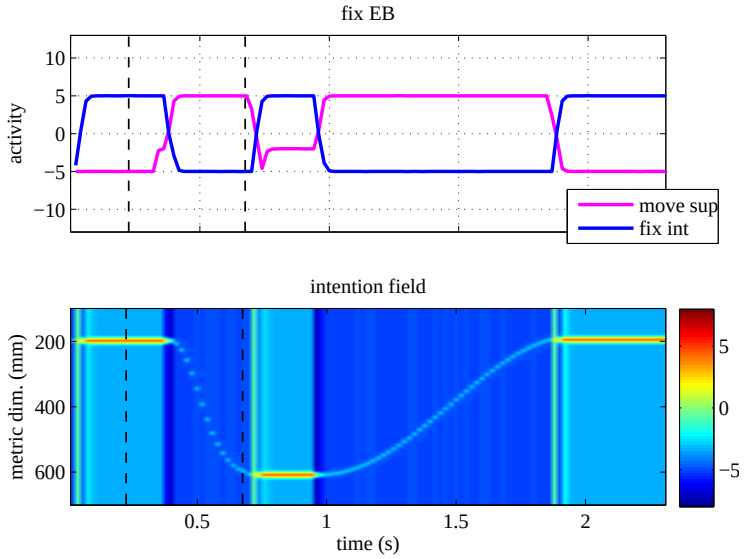


Figure 3.17: The time course of activation of the 'fix' EB when activated/deactivated through the suppression node 'move sup' each time the movement is terminated during a successful catch.

After reaching the target location of each movement (catching point or base), the 'move' EB CoS field forms a peak activation that switches on the EB CoS node ('move CoS'). In return, the 'move CoS' node inhibits the intention node 'move int'. The inhibited 'move int' node switches off the oscillatory regime ($c_{\mathrm{hopf}} = 0$) and deactivates the suppression node 'move sup'. Consequently, the 'fix' EB intention node 'fix int' turns on and switches the dynamics back to the postural regime ($c_{\mathrm{post}} = 1$) to fixate the end-effector. By default, the postural regime permits to stabilize the end-effector state at its current position before starting and after terminating every timed movement. As you can observe, a systematic delay is introduced before actual movement is initiated mainly due to the update phase and the switching dynamics. This time delay is undesirable and can be reduced by tuning the time scale of the dynamics.

3.5.2 Catching movement abortion after ball reflection

In this experiment, the ball is reflected straight to the top during the end-effector movement toward the catching point (Figure 3.18). The ball reflection led the end-effector to abort the timed catching before reaching the target and to start a backward movement to the base.

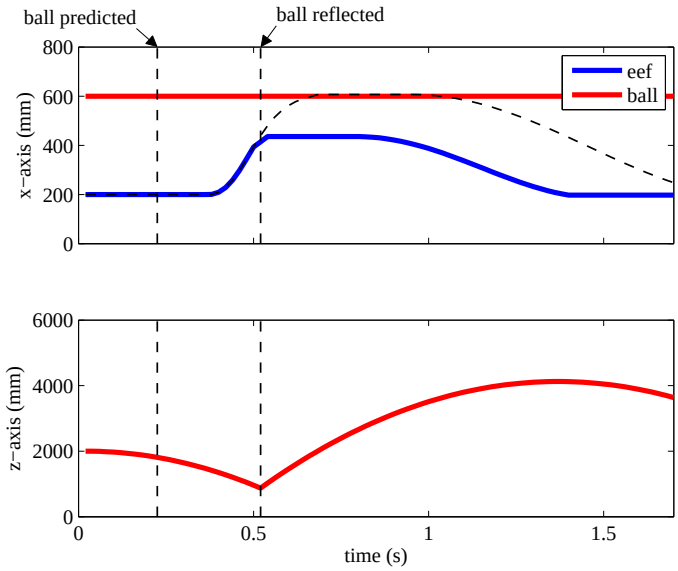


Figure 3.18: The end-effector trajectory when the ball is reflected during the catching movement. The thin and dotted black line shows the unperturbed trajectory of the end-effector.

The ball reflection ($b_{\text{catchable}} = -1$) occurs at $t \approx 0.52\,\text{s}$ and the perceptual field looses it activation (Figure 3.19).

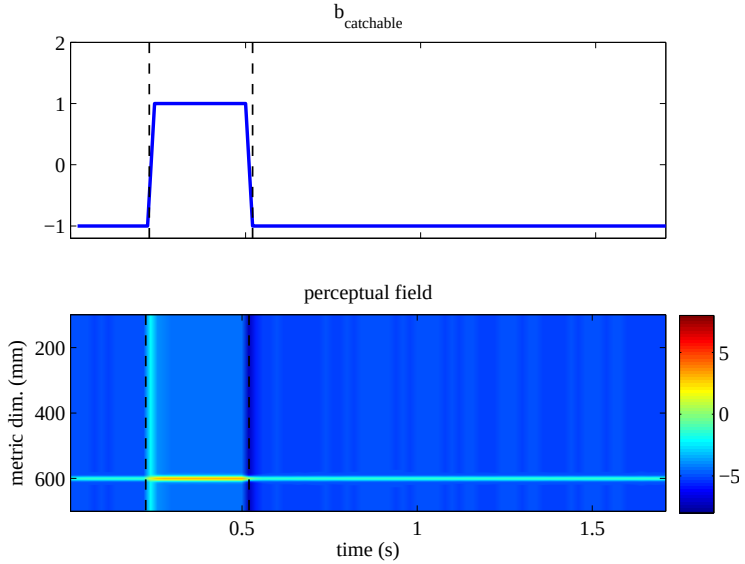


Figure 3.19: The trajectory of the perceptual parameter $b_{\text{catchable}}$ and the time course of activation of the perceptual field as the ball is reflected during the catching movement.

The perceptual precondition ‘mtc perc’ returns to its active state and inhibits the ‘move to catch’ EB intention node ‘mtc int’ which deactivates the movement module and stops the timed behavior (Figure 3.20).

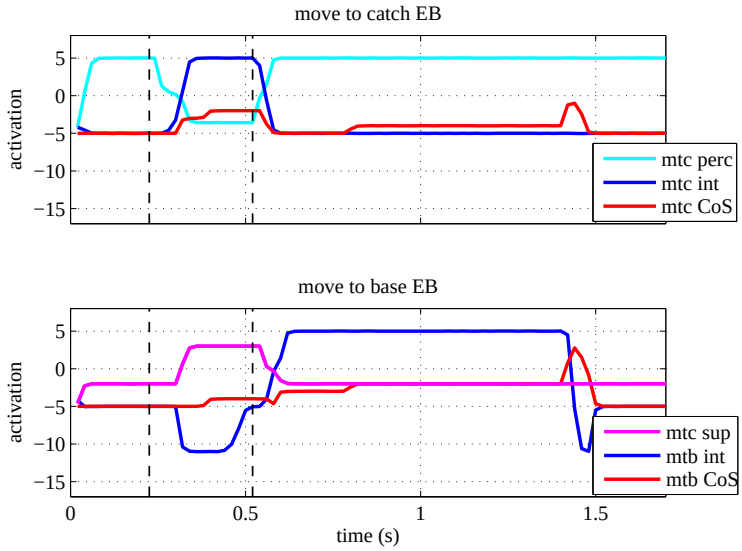


Figure 3.20: The time course of activation of the higher level EBs and the dynamical nodes implementing the perceptual and sequential constraints as the ball is reflected during the catching movement.

The 'move to base' EB intention node 'mtb int' gets activated and reactivates the movement module (Figure 3.21). The initial state is updated to the current position of the end-effector before starting the backward movement to the base.

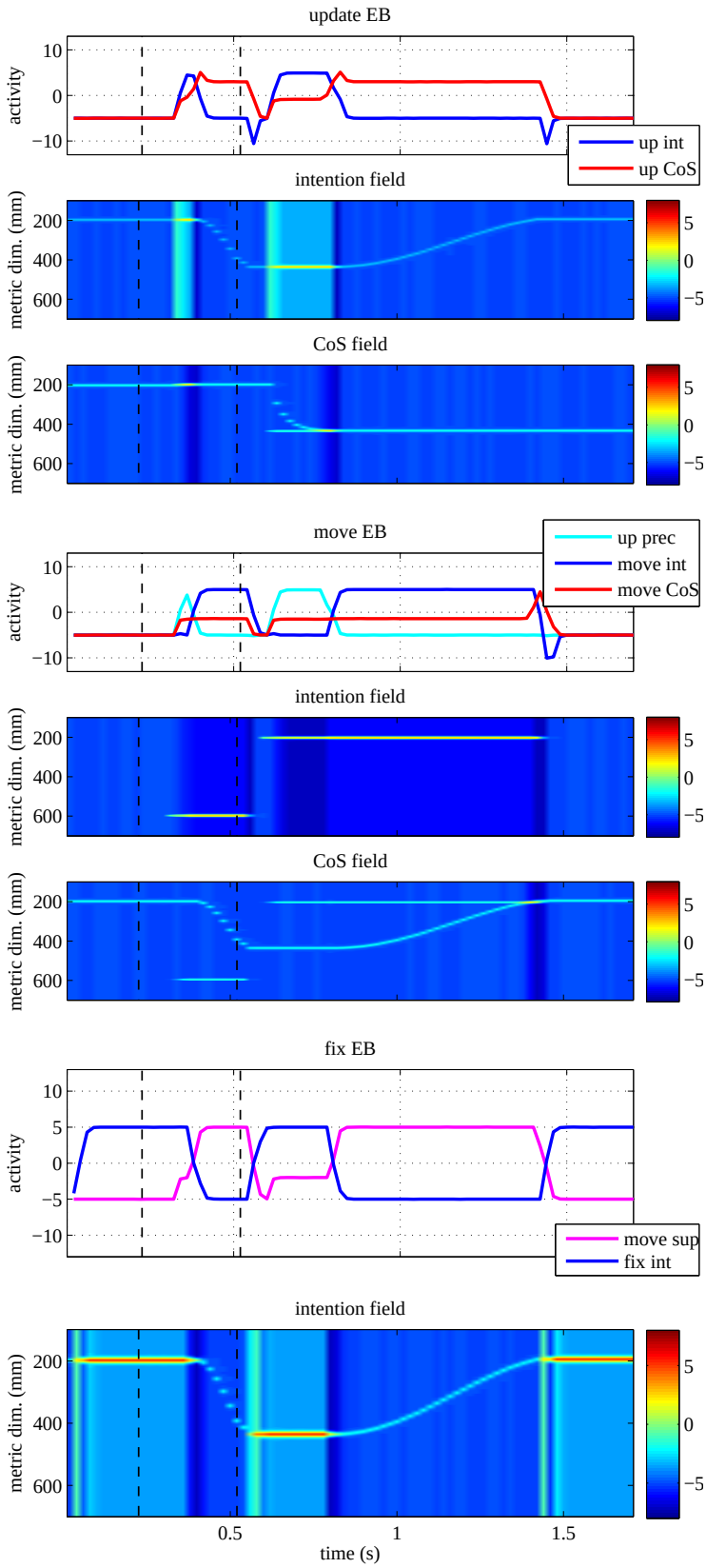


Figure 3.21: The time course of activation of the movement module EBs as the ball is reflected during the catching movement.

3.5.3 Reactivating a catching sequence

During this experiment, I show how the robot is able to reactivate a supplementary timed movement sequence to execute the catch after ball reflection (Figure 3.22).

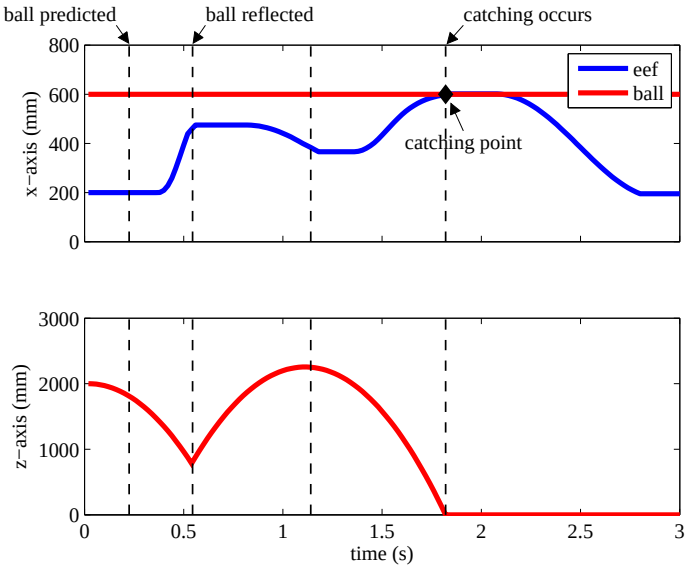


Figure 3.22: A supplementary catching sequence is reactivated while moving backward toward the base.

The ball reflection at $t \approx 0.52\,\mathrm{s}$ led to stop the execution of the catching behavior and initiate the backward movement toward the base. Simultaneously, the reflected ball starts falling down again at $t \approx 1.1\,\mathrm{s}$ and the robot re-initiated a new catching sequence from the current end-effector position to successfully catch the ball at $t \approx 1.82\,\mathrm{s}$ before moving back to the base.

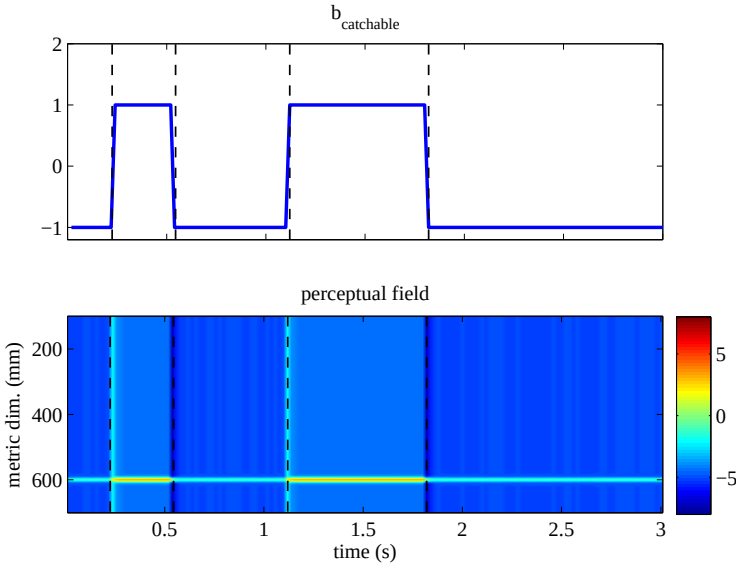


Figure 3.23: The trajectory of the perceptual parameter $b_{\text{catchable}}$ and the time course of activation of the perceptual field as a catching sequence is reactivated during the backward movement.

These events are translated to the perceptual parameters as shown in Figure 3.23 and the four timed movements are controlled via the higher level EBs as depicted in Figure 3.24.

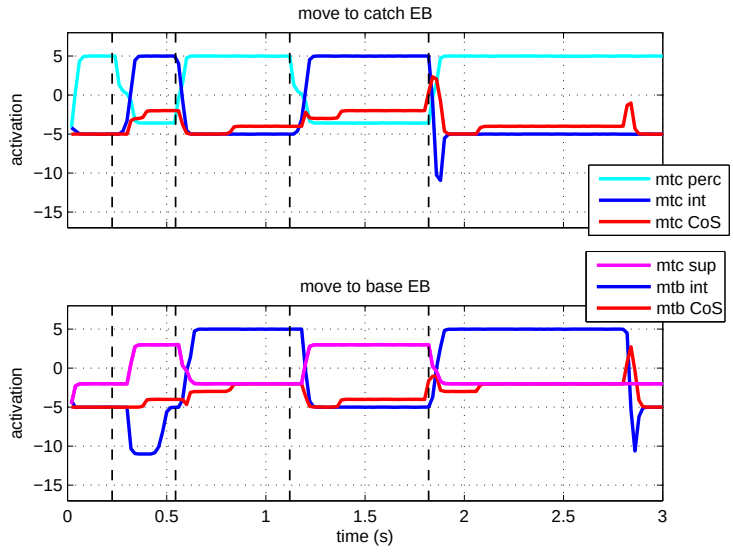


Figure 3.24: The time course of activation of the higher level EBs and the dynamical nodes implementing the perceptual and sequential constraints as a catching sequence is reactivated during the backward movement.

The movement module is activated four times at different positions along the metric dimension where, at each time, the initial state is updated before initiating every timed movement (Figure 3.25).

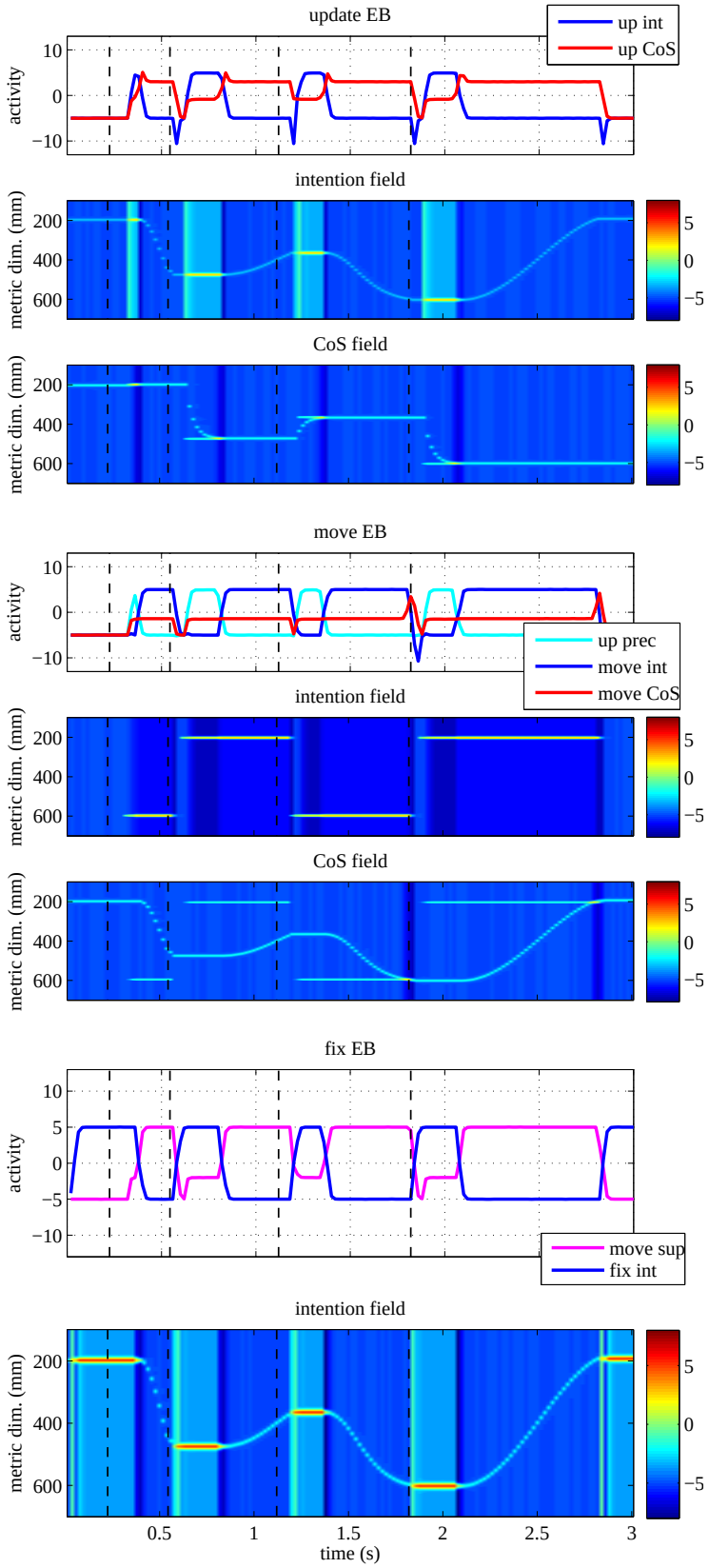


Figure 3.25: The time course of activation of the movement module EBs as a catching sequence is reactivated during the backward movement.

3.5.4 Updating the catching movement after ball deviation

In the last experiment, the ball is deviated during the end-effector catching movement as shown in Figure 3.26. In order to successfully catch the ball, the robot had to accelerate toward the new predicted catching point. This acceleration ensures the correct timing of the movement and is due to the update rule of the dynamics cycle time in Eq. 3.3 (see Section 3.2.1).

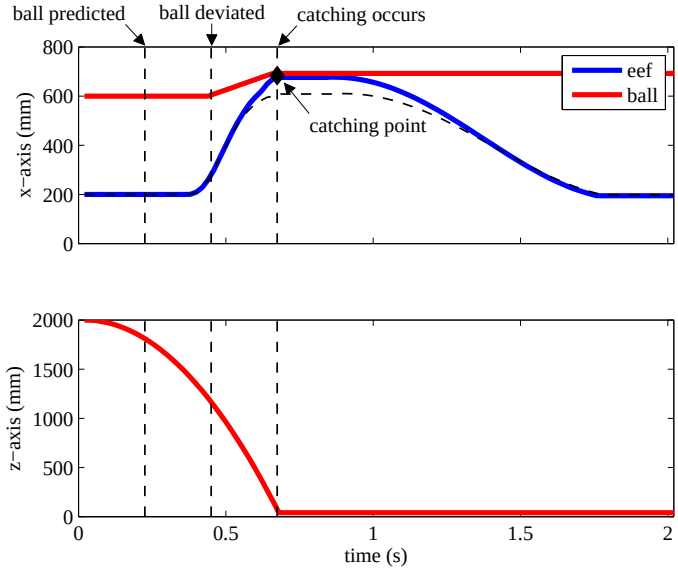


Figure 3.26: The end-effector trajectory when the ball is deviated during the catching movement.

Figure 3.26 depicts the adapting end-effector trajectory to the deviated ball as compared to the unperturbed case. The ball deviation occurs approximately at one third cycle of the movement time ($t \approx 0.42\,\mathrm{s}$). The end-effector starts accelerating toward the new catching point and catches the ball at $t \approx 0.68\,\mathrm{s}$ before initiating a backward movement to the base. The perceptual field reflected online the varying catching point prediction (Figure 3.27) and allowed the target location of the movement to be updated in the 'move' EB intention field as shown in Figure 3.29.

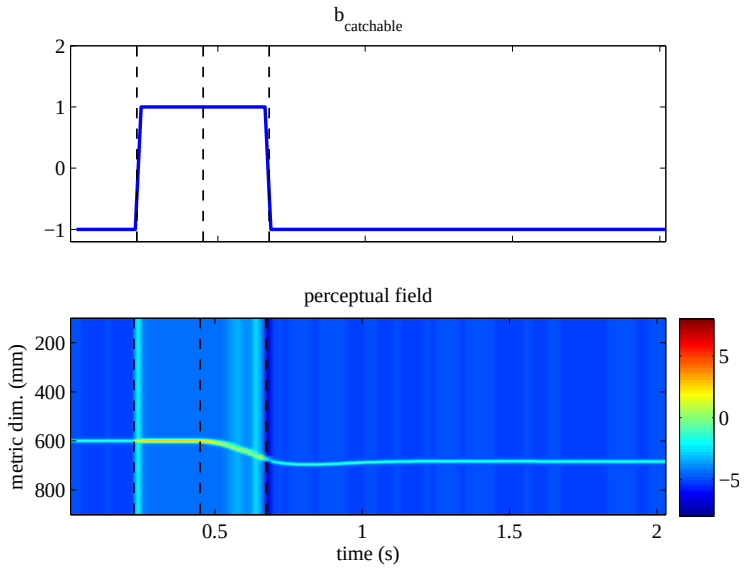


Figure 3.27: The trajectory of the perceptual parameter $b_{\text{catchable}}$ and the time course of activation of the perceptual field as the ball is deviated during the catching movement.

The higher level EBs (Figure 3.28) executed the updating movement by activating the movement module as depicted in Figure 3.29.

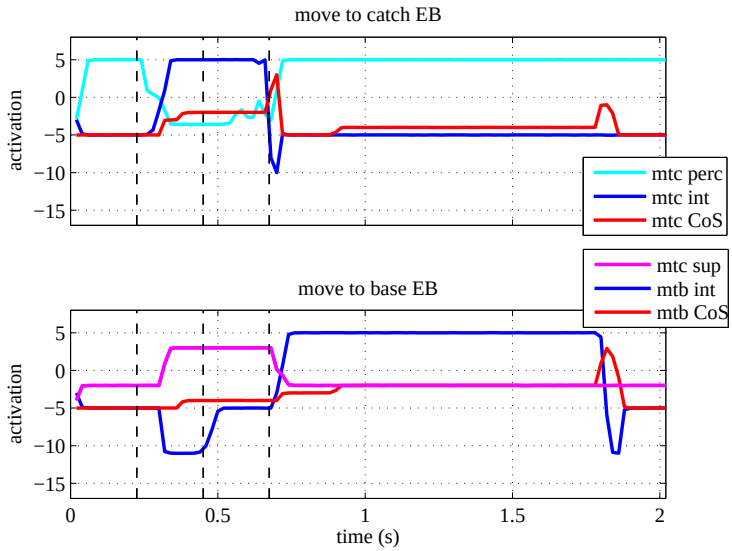


Figure 3.28: The time course of activation of the higher level EBs and the dynamical nodes implementing the perceptual and sequential constraints as the ball is deviated during the catching movement.

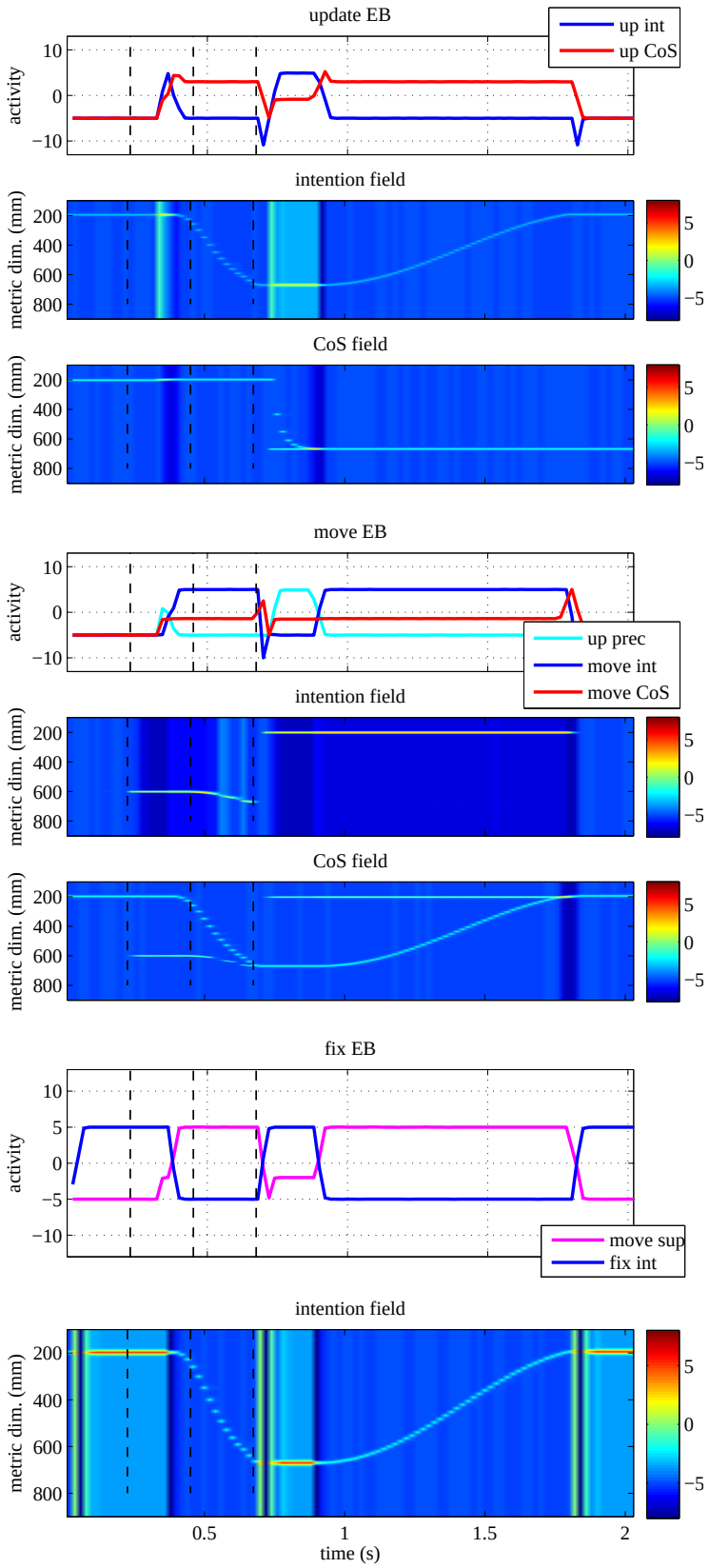


Figure 3.29: The time course of activation of the movement module EBs as the ball is deviated during the catching movement.

3.6 Discussion

In this chapter, I have introduced a dynamical model for timed movements sequence generation. The model combines oscillator and fixed point dynamics to generate discrete timed movements and a hierarchical neural dynamics architecture for behavioral organization. The timed actions are organized into sequences based on behavioral constraints between the robot's EBs and continuous linkage to perceptual information about the environment. The core properties of the system are demonstrated in a simulated catching task on a free falling ball.

The results shows that the system is able to successfully execute a robotic task that involves sequences of timed movements. Moreover, the direct coupling to a perceptual system allows the hierarchical architecture to autonomously and flexibly react to time-varying sensory information about the task by aborting a running behavior as well as initiating or re-initiating inactive behavior depending on the current situation. Furthermore, the movement parameters can be updated to adapt the trajectory on the fly to changing perceptual predictions.

While the fairly simple catching example illustrates the general characteristics of the system on a single metric dimension, the model can be easily extended to more complex robotic tasks. Making use of the hierarchical structure and the standard design of a movement module, the properties of the dynamical model will be demonstrated in a robotic hitting task (Chapter 4) where several movement components of a robot manipulator are involved and defined along multiple feature dimensions.

Chapter 4

Application of the dynamical model to a robotic hitting task

The dynamical model introduced in Chapter 3 permits to generate and flexibly organize sequences of timed movements that are tiedly coupled to sensory information about the task environment. Such characteristics play a major role in a variety of robotic applications where the manipulator needs to coordinate multiple actions with external objects or events and adapt online the movements to a quickly changing environment as for juggling or hitting a ball. Humans exhibit natural skills in performing such tasks where they are able to blend and sequence several movement primitives and coordinate their execution with a perceptual feedback.

In this chapter, I demonstrate how the proposed dynamical model can be extended, with low efforts, from a simple catching task to a relatively more complex robotic application. The adopted scenario is a blend of playing pinball machine and airhockey: a redundant manipulator arm that keeps hitting a ball up an inclined plane. The goal is to keep the ball in play on the inclined plane at all time. The robotic task requires several timed movements that need to be continuously sequenced in coordination with visual sensory information about the ball motion. In addition, obstacles on the inclined plane may introduce unpredictable perturbations in the ball trajectory. This implies a continual capacity of the system to quickly adapt the sequences of movement to new sensorial contexts, and update on the fly the movement dynamics during task execution.

Similar scenarios have been repeatedly used to probe real-time robotic control, the interception of a moving ball (Rapp, 2011; Senoo et al., 2006; Hailing et al., 2012). In my case, this choice was made to demonstrate my model in a task that involves several movement components for the end-effector. However, restrictions due to the hardware and speed limitation of the manipulator, imposed a simplified experimental setup. Indeed, the movement generation is performed in 2D and the ball velocity is tuned by the plane inclination.

The rest of the chapter is organized as follows. A detailed description of the experimental setup is given in Section 4.1 followed by a presentation of the extended model for the robotic hitting task in Section 4.2. The performance of the model as well as the reactions to different perturbation experiments are shown in Section 4.3 and these results are discussed in Section 4.4.

4.1 Task setting

An overview of the experimental setup for the robotic hitting task is shown in Figure 4.1. 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) used to hit the ball. The robot is equipped with a vision system that tracks and predicts the ball trajectory by means of a color based tracking process and a linear Kalman filter (respectively). The hitting occurs at a virtual *hitting line* 'just in time' to drive the ball back up the inclined plane before returning back to a virtual *base line*.

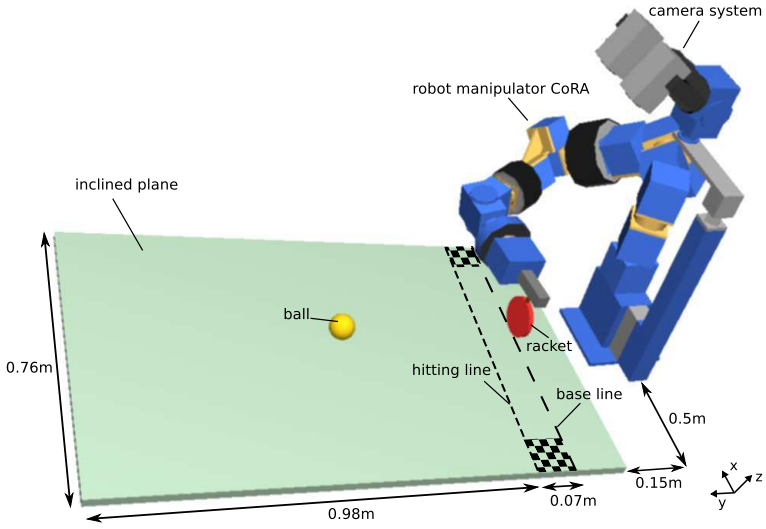


Figure 4.1: Graphical overview of the experimental setup.

Ideally, the ball is hit continuously and kept in play on the inclined plane at all times. The hitting region is constrained by safety margins set on both sides of the hitting line (marked with a checkerboard pattern in Figure 4.1) to prevent the real robot from colliding the inclined plane's 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 its landing position along the hitting line, which specify the hitting point $\boldsymbol{x}_{\text{hp}}$. In addition, the hitting movement is initiated only if the $\boldsymbol{x}_{\text{hp}}$ 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 referred here as the time-to-impact b_{tim}. These parameters control the initiation of the hitting movement sequences when the $\boldsymbol{x}_{\text{hp}}$ time is within a b_{tim} criterion and the ball is inside the robot's hitting region.

4.1.1 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 for hitting ($b_{\text{reachable}} = 1$), a timed movement

is initiated and brings the racket positions x_{eef} and y_{eef} to the predicted $\boldsymbol{x}_{\text{hp}}$. When the ball b_{tim} reaches a time threshold, a second timed movement moves the racket azimuth orientation ϕ_{eef} to hit the ball.

Once the hit occurs ($b_{\text{reachable}} = -1$), the racket ϕ_{eef} is brought to the initial orientation while its 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 as possible to the predicted $\boldsymbol{x}_{\text{hp}}$ for the next hitting sequence (a disposition imposed by the speed limitation the robot). During in the tracking movement, the robot is ready to start another timed movement sequence to hit the ball whenever the vision system signals a predicted $\boldsymbol{x}_{\text{hp}}$ and b_{tim} satisfying the task conditions.

The timed movement 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 it is reflected before a hit occurs. Furthermore, these timed movements are sequenced and adapted autonomously allowing a flexible re-organization able to accommodate any disturbance in the ball trajectory.

4.1.2 Visual system

The robot manipulator is equipped with a stereo cameras system (SONY DFW VL500) mounted on a support above the arm on an additional 2 DoF pan/tilt Amtec module. The pan range is about 180°, and the tilt movement is in the range of 90°. The cameras module is oriented toward the inclined plane (pan = 0°, tilt = 60°). It is operated with the maximum frame rate of 30 frames/s providing a new measurement estimate every ≈ 33 ms. To acquire sensory information about the ball motion, only the left camera (with respect to the robot) is used.

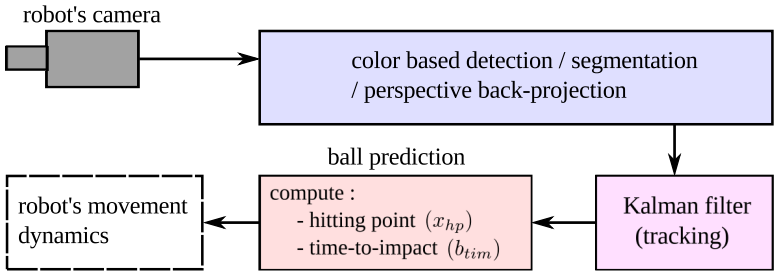


Figure 4.2: General diagram of the visual system.

The robot's cameras module tracks the ball on the inclined plane by means of a color-based tracking process. Based on the position measurements and a discrete time motion model, a linear Kalman filter is used to estimate the velocity of the ball and predict its trajectory. The hitting point $\boldsymbol{x}_{\text{hp}}$ is computed as the intersection between the ball heading vector and the hitting line while the time-to-impact b_{tim} is approximated using the motion model (see Figure 4.2). These perceptual information are extracted and algorithmically pre-processed to be included as parameters in the dynamical model for timed movements sequence generation. Building a neurally plausible tracking and prediction mechanism is beyond the present work.

Color based detection process

To detect the colored ball and measure its position on a uniformly white inclined plane, a color-based detection process is used (see Figure 4.3). Many object detection mechanisms exist and could have been used in this context, see (Yilmaz et al., 2006) for a detailed review. In my case, the choice of a color-based detection process was mainly motivated by the simple experimental setup adopted for the robotic hitting task.

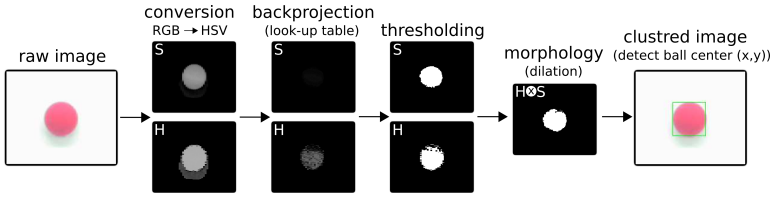


Figure 4.3: Overview of the color-based segmentation process.

From a still raw RGB (Red, Green, Blue) camera image, the detection process starts by generating empirically and off-line color histograms of the ball in HSV (Hue, Saturation and Value) space. These histograms represent color distributions in each of the image channels and are used as reference models during the detection process.

The color-based detection process is composed of five precessing steps implemented in C++ using the OpenCV library (Bradski and Kaehler, 2008) and run sequentially:

1. the input camera image is converted from the RGB to HSV color space and split into the three image channels: hue, saturation and value.

2. back projection of the hue and saturation channels is performed using the generated histogram models. For every image channel, the bin location corresponding to each pixel is searched for. These bin locations are then looked up in the histogram model and their values are read and stored in a back-projected image.

3. thresholding operation is performed in the back-projected hue and saturation channels resulting into two binary images.

4. dilation operation with a rectangular kernel is applied to the binary images. The objective is to merge neighboring but disparate pixels in each image channel that probably belongs to the detected ball. The resulting images are multiplied together element by element and combined in a unique refined image.

5. a clustering algorithm is applied on the refined image to obtain a measure of the ball position in the camera's image plane. This algorithm performs a search in the image and groups the pixels marked to be part of the object into a cluster that its center represents the ball position $\boldsymbol{q} = [u\,,\,v]^T$ in image plane.

In order to obtain the ball position $\boldsymbol{p}_{\mathrm{b}} = [x\,,\,y]^T$ in the inclined plane coordinate system, a perspective back-projection from the image to the inclined plane reference frames is required. More details about the camera projection operations can be found in Section A.1.

Kalman tracking

The position measurements provided by the detection process are corrupted by noise that may originate from various sources such that camera's sensors inaccuracies or lighting changes during image grabbing. The role of the tracking module is to keep track where the ball is at any time and provide estimates of its velocity on the inclined plane and all that based on these noisy measurements.

To perform this job, I opted for a Kalman filter. In short, a Kalman Filter is a recursive and model-based data processing algorithm that estimates the state of a noisy linear dynamic system (Schutter et al., 1999; Grewal and Andrews, 1993). It operates recursively on streams of noisy input measurements to produce a statistically optimal estimate of the underlying system state. A brief review of the linear Kalman estimator can be found in Section A.2. In order to track the ball and produce estimates of its real position and velocity by a Kalman filter, I had to make three assumptions:

1. the ball is rolling on the inclined plane without slip. The choice of small inclination should enforce this assumption.

2. ball motion can be modeled by a linear discrete-time dynamic system. This is a naive assumption because it leads to increase the model uncertainties by neglecting many non-linear parameters like air drag, spin and rolling friction that actually affect the ball motion.

3. camera measurement noise is Gaussian, white and normally distributed.

However, these assumptions are found to be reasonable and with the help of the Kalman estimator, I could produce acceptable predictions of the ball motion.

The ball dynamics is governed by the weight $m_{\mathrm{b}}\cdot g$ and friction force f, where m_{b} is the ball mass and g is the gravity constant. It is clear from Figure 4.4 that the gravity force affects the motion only along the inclined plane's y-axis while the friction effect can be decomposed into its orthogonal components for the x- and y-axes motions. Note that when the ball is rolling upward, for example after a hit, friction and weight forces along the y-axis act in the same downward direction trying to decelerate the motion. However, when the ball starts rolling down, the weight will accelerate the ball while friction is still trying to decelerate the motion. Note also that the friction f is necessary for rolling and if applied to the instantaneous rotation axis, it will not produce work and therefore does not dissipate mechanical energy (Domenech et al., 1987).

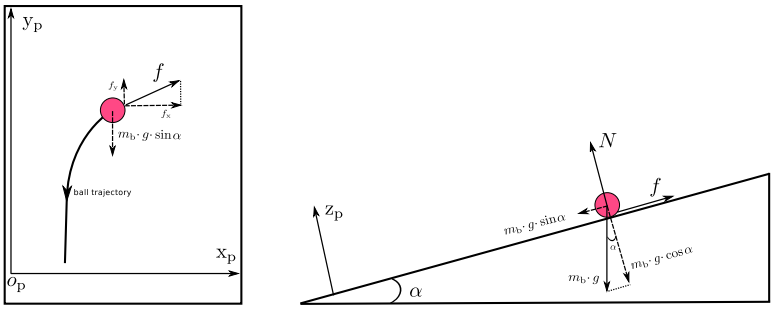


Figure 4.4: The dynamics of the ball. The side view drawing (right) shows the external forces acting on a ball rolling without slip on an inclined plane. The weight ($m_\mathrm{b}\, g$) is acting vertically on the ball center of mass. The normal reaction $\boldsymbol{N}$ of the inclined plane is equal to the weight projection ($m_\mathrm{b}\, g\, \cos\alpha$). $\boldsymbol{f}$ is the friction force, necessary for rolling. In the top view drawing (left), we can see that the friction force can be split into its orthogonal components (f_x , f_y). While f_x acts along x-axis, the component f_y affects the y-axis motion. We can also see that the weight projection ($m_\mathrm{b}\, g\, \sin\alpha$) acts only along y-axis.

Physically, forces that induce acceleration and velocities do not influence each other, if they are perpendicular (orthogonal) and independent (Kohler, 1997). According to that, the ball motion can be modeled as two separate movements along x- and y-axes. Hence, to track the ball, two separate Kalman recursions are evaluated and a linear discrete-time model is derived for each movement axis of the inclined plane

For motion along the x-axis:

$$\boldsymbol{x}_{\mathrm{k+1,x}} = \boldsymbol{A}_\mathrm{x}\boldsymbol{x}_{\mathrm{k,x}} + \boldsymbol{w}_{\mathrm{k,x}}. \tag{4.1}$$

For motion along the y-axis:

$$\boldsymbol{x}_{\mathrm{k+1,y}} = \boldsymbol{A}_\mathrm{y}\boldsymbol{x}_{\mathrm{k,y}} + \boldsymbol{B}_\mathrm{y}u_{\mathrm{k,y}} + \boldsymbol{w}_{\mathrm{k,y}}. \tag{4.2}$$

In Eq. 4.1 and 4.2, the state vectors $\boldsymbol{x}_{\mathrm{k,x}} = [x_\mathrm{k}\,,\, v_{\mathrm{k,x}}]^T$ and $\boldsymbol{x}_{\mathrm{k,y}} = [y_\mathrm{k}\,,\, v_{\mathrm{k,y}}]^T$ consist of the ball position and speed at time t_k along the inclined plane x- and y-axes (respectively). The transition matrices $\boldsymbol{A}_\mathrm{x}$ and $\boldsymbol{A}_\mathrm{y}$ describe the motion by relating the present state vector at $t_{\mathrm{k+1}}$ to the previous state at t_k with $\Delta t = t_{\mathrm{k+1}} - t_\mathrm{k}$, and are given by

$$\boldsymbol{A}_\mathrm{x} = \boldsymbol{A}_\mathrm{y} = \begin{bmatrix} 1 & \Delta t \\ 0 & 1 \end{bmatrix}. \tag{4.3}$$

The random variables vectors $\boldsymbol{w}_{\mathrm{k,x}} = [w_{\mathrm{k,x}}\,,\, w_{\mathrm{k,v_x}}]^T$ and $\boldsymbol{w}_{\mathrm{k,y}} = [w_{\mathrm{k,y}}\,,\, w_{\mathrm{k,v_y}}]^T$ represent the Gaussian and white process noise affecting position and speed. The process (additive) noise is used to account for uncertainties in the derived motion model. For the motion along y-axis in Eq. 4.2, the control matrix $\boldsymbol{B}_\mathrm{y}$ is defined by

$$\boldsymbol{B}_\mathrm{y} = \begin{bmatrix} \Delta t^2 \\ \Delta t \end{bmatrix}. \tag{4.4}$$

The matrix $\boldsymbol{B}_\mathrm{y}$ relates the control input $u_{\mathrm{k,y}}$ to the state vector $\boldsymbol{x}_{\mathrm{k,y}}$. The input $u_{\mathrm{k,y}}$ is the commanded acceleration a_y induced by the gravity and affecting

the ball motion along the inclined plane y-axis as shown in Figure 4.4. Following the derivation process in Domenech et al. (1987), the acceleration a_{y} can be derived as

$$u_{\mathrm{k,y}} = a_{\mathrm{y}} = \frac{g \, \sin\alpha}{1 + \frac{I_{\mathrm{b}}}{m_{\mathrm{b}} \, r_{\mathrm{b}}^2}}, \tag{4.5}$$

where m_{b} is the ball mass, g is the gravitational constant, α is the plane inclination, r_{b} is the ball radius and I_{b} is the ball inertia defined by

$$I_{\mathrm{b}} = \frac{2 \, m_{\mathrm{b}} \, r_{\mathrm{b}}^2}{5}. \tag{4.6}$$

For the motion along x-axis in Eq. 4.1, I assume that there is no control input since there is no gravity force acting. However, the ball motion is in reality not uniform because of perturbations due, for example, to rolling friction and air drag. These perturbations are modeled by the process noise $\boldsymbol{w}_{\mathrm{k,x}}$.

The linear measurement systems corresponding to the motion models in Eq. 4.1 and 4.2 are (respectively)

$$z_{\mathrm{k,x}} = \boldsymbol{H}_{\mathrm{x}} \begin{bmatrix} x \\ 0 \end{bmatrix} + s_{\mathrm{k,x}}, \tag{4.7}$$

$$z_{\mathrm{k,y}} = \boldsymbol{H}_{\mathrm{y}} \begin{bmatrix} y \\ 0 \end{bmatrix} + s_{\mathrm{k,y}}, \tag{4.8}$$

where $z_{\mathrm{k,x}}$ and $z_{\mathrm{k,y}}$ are the measurement variables of the ball position along the x- and y-axes at time t_{k} (respectively). The measurement matrices $\boldsymbol{H}_{\mathrm{x}}$ and $\boldsymbol{H}_{\mathrm{y}}$ are given by

$$\boldsymbol{H}_{\mathrm{x}} = \boldsymbol{H}_{\mathrm{y}} = \begin{bmatrix} 1 & 0 \end{bmatrix}. \tag{4.9}$$

The matrices $\boldsymbol{H}_{\mathrm{x}}$ and $\boldsymbol{H}_{\mathrm{y}}$ relates, respectively, the variables $z_{\mathrm{k,x}}$ and $z_{\mathrm{k,y}}$ to the position vector $\boldsymbol{p}_{\mathrm{b}} = [x \, , \, y]^T$ provided by the detection process. The scalars $s_{\mathrm{k,x}}$ and $s_{\mathrm{k,y}}$ represent the measurement noise and are used to model, for example, camera sensors inaccuracies or lighting changes. In practical, the measurement noise is already present in the position vector $\boldsymbol{p}_{\mathrm{b}}$ and is shown here for completeness and taken to be Gaussian and white.

From the results obtained from experiments conducted in the real setup, we can show the estimator's performance in tracking the ball trajectory when the latter is injected on the inclined plane. Figure 4.5 depicts the predicted, measured and estimated trajectories of the motion in a sample throw trial.

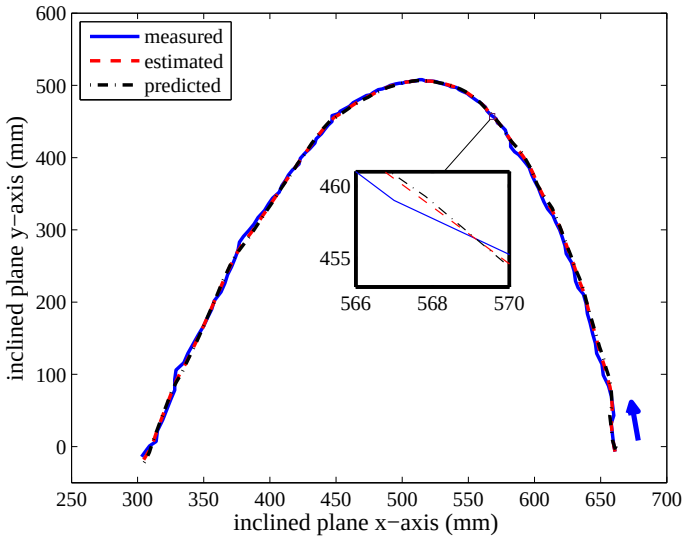


Figure 4.5: The predicted, measured and estimated position trajectories as the ball rolls on the inclined plane.

During the tracking process, the speed estimates are produced through the cross-correlation between position and speed in the ball motion model. One way to verify the correctness of these estimates, at least to a certain level, is to superpose the position trajectory with a plot of the estimated velocity vectors at every time t_k. As shown in Figure 4.6, we can clearly see that the estimated velocity is always tangent to the trajectory and following the movement direction.

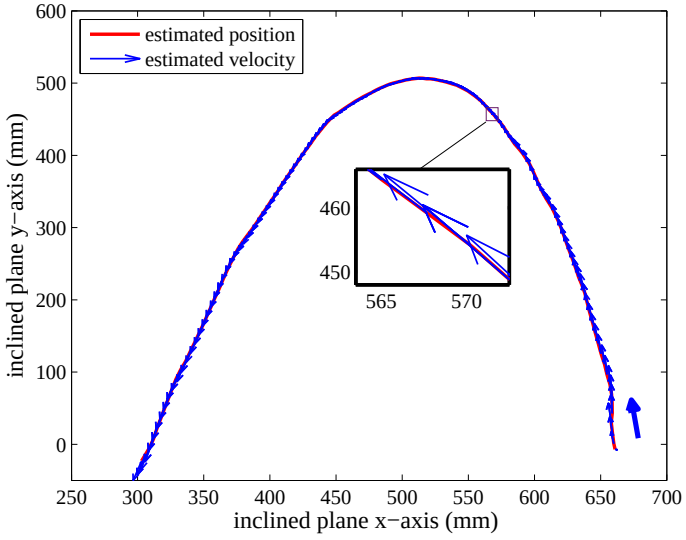


Figure 4.6: The velocity of the ball is plotted over the estimated position trajectory.

Additionally, the Kalman filters were able to handle situations where the ball bounced on the right or the left borders of the inclined plane even though no special dispositions were taken to deal with this non-linearity which illustrates the robustness of the implementation.

Prediction of the ball trajectory

For the robotic hitting task, a precise prediction of the ball motion is crucial and could have been done using different methods:

- **Model based methods**: consist of integrating over time a derived motion model to predict the hitting point as in (Muelling et al., 2010) for a robot table tennis application or in a paddle juggling scenario (Nakashima et al., 2006). In practical, model based prediction mechanisms are as precise as the motion models are. In addition, they are able to provide precise predictions only if sufficient time is given to the model integrator. For the task considered in this work, the derived motion model contains too many uncertainties to be used to predict the ball trajectory (see Subsection 4.1.2). Furthermore, the small surface of the experimental scene, consisting of the inclined plane, imposes fast and precise predictions.

- **Curve fitting based methods**: where curve fitting techniques are applied on pre-recorded set of measurements data during the ball motion as in (Senoo et al., 2006) where successive least-squares estimations are used to predict the ball trajectory in a high speed robot batting task. Hailing et al. (2012) used a similar technique to determine the hitting position in a Ping-Pong robotic application. In general, such methods need a large amount of measurements data and so, rely on high speed camera systems to be effective. In my application, a camera frame rate of 30 frames/s limits considerably the use of such techniques.

- **Learning based methods**: estimate the ball dynamics based on several, off-line, throw trials and use the estimated parameters to predict its motion during task execution. For example, Kim et al. (2010) used an autonomous dynamical system to estimate the dynamics and predict the catching point of a moving ball for an iCub humanoid robot. In practical, the throw trials are performed in an invariant experimental setting. This means that any change in the setting implies performing again the throws and re-estimating the dynamics. In order to have the possibility to change the application setup, like the plane inclination or the type of the ball, without the need to redo throw trials, I did not consider learning approaches in this work.

- **Kalman filter based methods**: modify the Kalman recursion algorithm by introducing factors to enable long-term predictions as in (Hujic et al., 1998). However, such methods assume a well-defined motion model of the ball and suffers from the lack of reactivity when sudden changes happen during motion. Such methods have not been tested in this work.

To extract the perceptual parameters and variables required for the task, a geometrical approach based on the ball position and speed estimates is developed.

Hitting point $\boldsymbol{x}_{\mathbf{hp}}$. The hitting point $\boldsymbol{x}_{\mathrm{hp}}$ is a prediction of the ball landing position as it starts rolling down on the inclined plane. A *curvilinear* Kinematics study of the ball trajectory permits to compute the motion direction at position $(x_{\mathrm{k}}, y_{\mathrm{k}})$ and time t_{k} using the tangent velocity vector at that position.

The extracted direction is used to approximate the trajectory as a *rectilinear* motion along a straight line segment of length L. While this approximation doesn't permit to predict the overall trajectory at any moment in time, it provides an estimation of the landing position that increases in precision as the ball approaches the hitting line. The prediction process is continuously performed during task execution and is described in the following:

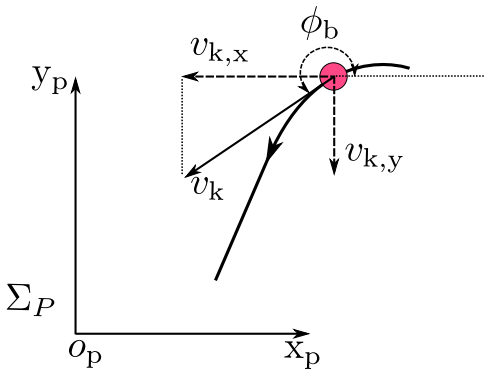


Figure 4.7: The ball heading direction ϕ_b on the inclined plane coordinate system Σ_P.

The heading direction of the moving ball (Figure 4.7) is continuously computed using the velocity vector estimate $\boldsymbol{v}_\mathrm{k} = [v_{\mathrm{k,x}}, v_{\mathrm{k,y}}]^T$ by the formula

$$\phi_\mathrm{b}(t_\mathrm{k}) = \phi_\mathrm{b} = \arctan \frac{v_{\mathrm{k,y}}}{v_{\mathrm{k,x}}}. \tag{4.10}$$

In a coordinate system Σ_B attached to the ball but always aligned with the external inclined plane coordinate system Σ_P, a vector $\boldsymbol{r} = [L\,,\,0\,,\,0]^T$ is defined, where L is a suitably large number. Using the heading direction $\phi_\mathrm{b}(t_\mathrm{k})$, the current ball position and after some simple transformations, a line segment $\boldsymbol{s}_\mathrm{h}$ of length L can be derived to represent the ball heading vector (Figure 4.8).

$\boldsymbol{x}_\mathrm{hp}$ is computed as the intersection point between $\boldsymbol{s}_\mathrm{h}$ and $\boldsymbol{s}_\mathrm{d}$ where $\boldsymbol{s}_\mathrm{d}$ is a line segment placed at the bottom of the inclined plane and parallel to the hitting line with a distance equal to the ball radius r_b as shown in Figure 4.8.

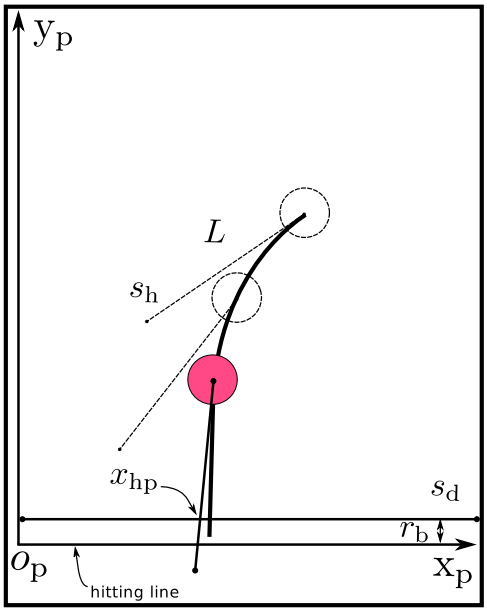


Figure 4.8: The hitting point $\boldsymbol{x}_{\mathrm{hp}}$ is computed as the intersection point of the line segments $\boldsymbol{s}_{\mathrm{h}}$ and $\boldsymbol{s}_{\mathrm{d}}$.

The predicted landing position $\boldsymbol{x}_{\mathrm{hp}}$ is continuously tested to be inside the robot's hitting region by

$$b_{\mathrm{reachable}} = \begin{cases} 1, & \text{if } \boldsymbol{x}_{\mathrm{hp}} \text{ is predicted to be inside the hitting region.} \\ -1, & \text{otherwise.} \end{cases} \tag{4.11}$$

Time-to-impact b_{tim}. The time-to-impact b_{tim} is a prediction of the time needed by the ball to reach $\boldsymbol{x}_{\mathrm{hp}}$ and can be approximated by the formula

$$b_{\mathrm{tim}} = \frac{-v_{\mathrm{k,y}} - \sqrt{v_{\mathrm{k,y}}^2 - 2\,a_{\mathrm{y}}\,(y_{\mathrm{k}} - r_{\mathrm{b}})}}{a_{\mathrm{y}}}, \tag{4.12}$$

where a_{y} is the ball acceleration along y-axis and the difference $y_{\mathrm{k}} - r_{\mathrm{b}}$ represents the remaining distance to the hitting line.

Using the same throw trial of Figure 4.5, Figure 4.9 illustrates the prediction mechanism. The ball is launched on the inclined plane at time $t = 0\,\mathrm{s}$. While it is rolling toward the top, $\boldsymbol{x}_{\mathrm{hp}}$ is given as $[0\,,\,0]^T$ and b_{tim} as well as $b_{\mathrm{reachable}}$ are set to -1. As the ball starts rolling down and an intersection is detected at $t \approx 2.46\,\mathrm{s}$, the perceptual parameters are updated.

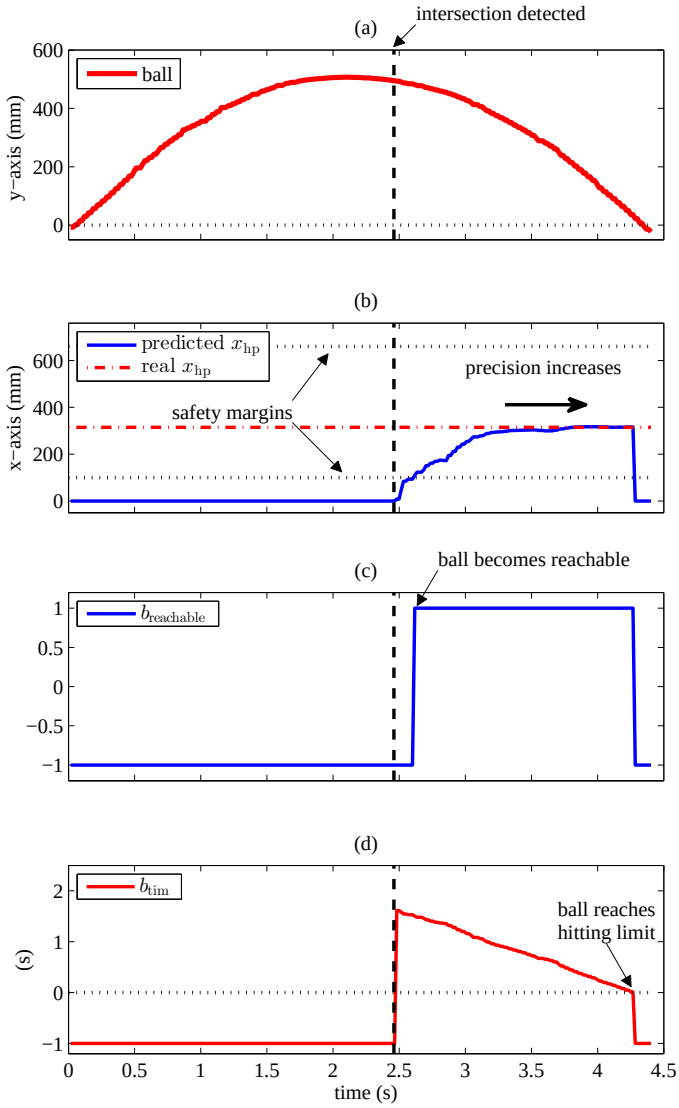


Figure 4.9: The prediction of the ball perceptual parameters : (a) depicts the ball trajectory along y-axis during the trial. When an intersection is detected (indicated by the black dotted line) the prediction of the landing position $\boldsymbol{x}_{\text{hp}}$ increases in precision as the ball approaches the hitting line (b). $b_{\text{reachable}}$ is set to 1 (c) as soon as $\boldsymbol{x}_{\text{hp}}$ is detected to be inside the robot's hitting region. (d) shows the time-to-impact b_{tim} estimation of the rolling ball.

The ball reaches the hitting line at $t \approx 4.28\,\text{s}$. If the ball falls out of the inclined plane or is hit by the robot's racket toward the top, the perceptual parameters are reset to their initial values.

4.1.3 Robotic agent

The Cooperative Robot Assistant (CoRA) in Figure 4.10 is a 7-degrees of freedom (DoFs) anthropomorphic robotic arm mounted on a 1 DoF trunk acting as a base joint. All the joints composing the manipulator are rotational. The CoRA configuration is equivalent to a broadly simplified model of a human arm with 3 DoFs shoulder, 1 DoF elbow, and 3 DoFs wrist. The last component of the

robotic arm is a two finger gripper device that represents the robot's end-effector for grasping. The maximum grasping radius is about 1 m with a maximum load of about 1.5 kg. The kinematic redundancy of the manipulator allows handling situations in which additional constraints like obstacle avoidance or joint limits has to be met while executing tasks like reaching or grasping (see Iossifidis and Schöner (2006); Reimann et al. (2011) for examples). CoRA is built as a modular system manufactured by "Amtec Cooperation". Each module has its own servo controller and communicates with the controlling PC via a CAN-bus interface integrated in an ISA card.

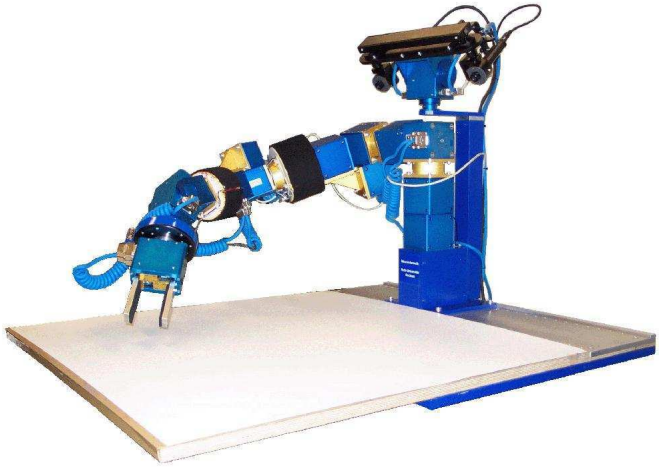


Figure 4.10: Anthropomorphic Robotic Assistant (CoRA).

CoRA is composed of two series of *roll,pitch,roll* joints for the shoulder and the wrist (Figure 4.11). This special structure allows the derivation of a closed form solution for the inverse kinematics Pieper (1968). Such a solution is always preferable for real time control of robots.

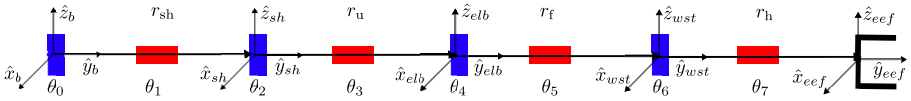


Figure 4.11: Initial CoRA arm configuration with the relevant coordinate systems.

The limb vectors $\boldsymbol{r}_{\mathrm{sh}}$, $\boldsymbol{r}_{\mathrm{u}}$, $\boldsymbol{r}_{\mathrm{f}}$, and $\boldsymbol{r}_{\mathrm{h}}$ define the spatial positions of the shoulder, upperarm, forearm, and hand segments (respectively). Given the *initial* azimuth ϕ_{eef} and elevation ϑ_{eef} orientations of the end-effector, the *initial* hand vector $\boldsymbol{r}_{\mathrm{h}}$ can be computed as

$$\boldsymbol{r}_{\mathrm{h}} = \boldsymbol{R}_{\mathrm{z}}^{\phi_{\mathrm{eef}}} \boldsymbol{R}_{\mathrm{y}}^{\vartheta_{\mathrm{eef}}} \hat{\boldsymbol{e}}_{\mathrm{x}} l_{\mathrm{h}}, \tag{4.13}$$

where l_{h} denotes the hand segment length, $\hat{\boldsymbol{e}}_{\mathrm{x}} \in \mathbb{R}^3$ represents an x-axis unit vector and $\boldsymbol{R}_{\mathrm{y}}$, $\boldsymbol{R}_{\mathrm{z}}$ are rotation matrices around the y- and z-axes (respectively) in a world reference frame Σ_b (Figure 4.12). After obtaining the gripper position $\boldsymbol{r}_{\mathrm{grip}}$ from the *desired* end-effector (or racket) position vector $\boldsymbol{r}_{\mathrm{eef}}$, we can compute the wrist vector $\boldsymbol{r}_{\mathrm{wst}}$ by

$$\boldsymbol{r}_{\mathrm{wst}} = \boldsymbol{r}_{\mathrm{grip}} - \boldsymbol{r}_{\mathrm{h}}, \tag{4.14}$$

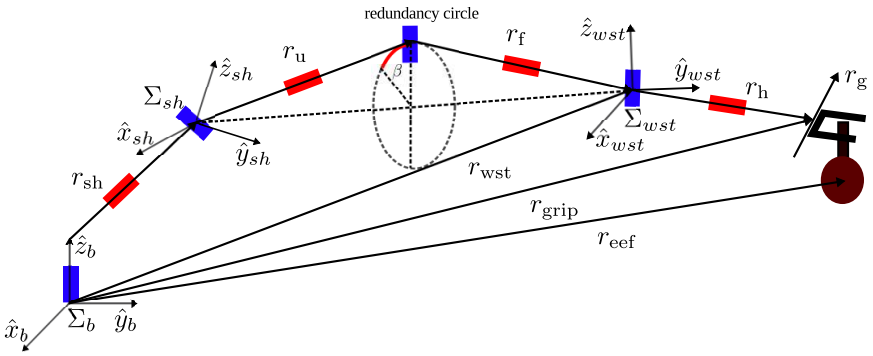


Figure 4.12: The wrist position determines the base joint θ_0.

The trunk of the robot θ_0 is controlled separately to bring the wrist to the desired position. The spatial position of the elbow with respect to the shoulder is specified by a predefined elbow angle β (see above) on a redundancy circle with a radius R

$$R = \sqrt{\|\boldsymbol{r}_{\mathrm{u}}\|^2 - \left(\frac{\|\boldsymbol{r}_{\mathrm{u}}\|^2 - \|\boldsymbol{r}_{\mathrm{f}}\|^2 + \|\boldsymbol{r}_{\mathrm{wst}}\|^2}{2\,\|\boldsymbol{r}_{\mathrm{wst}}\|^2}\right)^2}, \tag{4.15}$$

and centered around the vector $\boldsymbol{r}_{\mathrm{m}}$ laying on a ray pointing from the shoulder to the wrist

$$\boldsymbol{r}_{\mathrm{m}} = \frac{\|\boldsymbol{r}_{\mathrm{u}}\|^2 - \|\boldsymbol{r}_{\mathrm{f}}\|^2 + \|\boldsymbol{r}_{\mathrm{wst}}\|^2}{2\,\|\boldsymbol{r}_{\mathrm{wst}}\|^2}\,\boldsymbol{r}_{\mathrm{wst}}, \tag{4.16}$$

where $\boldsymbol{r}_{\mathrm{wst}}$ is expressed here in the shoulder reference frame Σ_{sh} and from which we can compute the elbow position by

$$\boldsymbol{r}_{\mathrm{u}} = \left(\boldsymbol{R}_{\mathrm{x}}^{\phi_{\mathrm{wst}}}\,\boldsymbol{R}_{\mathrm{z}}^{\vartheta_{\mathrm{wst}}}\,\boldsymbol{R}_{\mathrm{x}}^{\beta}\,\hat{\boldsymbol{e}}_{\mathrm{z}}\right)\,R + \boldsymbol{r}_{\mathrm{m}}, \tag{4.17}$$

where ϑ_{wst}, ϕ_{wst} are the wrist orientations and R_{y}, R_{z} are rotation matrices around the y- and z-axes (respectively) of the shoulder reference frame Σ_{sh}. Following a similar procedure, the vector $\boldsymbol{r}_{\mathrm{f}}$ can be derived and the joint angles $\theta_{i=1..4}$ can be computed from $\boldsymbol{r}_{\mathrm{u}}$ and $\boldsymbol{r}_{\mathrm{f}}$ using a straight forward solution.

Having the robot's maximum speed limitation and to achieve higher speed hitting movements, the manipulator hand segment $\boldsymbol{r}_{\mathrm{h}}$ is controlled separately and permits to compute the two joints θ_5 and θ_6. The *desired* azimuth ϕ_{eef} and the elevation ϑ_{eef} control the hand segment using the formula

$$\boldsymbol{r}_{\mathrm{h}} = \boldsymbol{R}_{\mathrm{z}}^{\phi_{\mathrm{eef}}}\,\boldsymbol{R}_{\mathrm{y}}^{\vartheta_{\mathrm{eef}}}\,\hat{\boldsymbol{e}}_{\mathrm{x}}\,l_{\mathrm{h}}, \tag{4.18}$$

where $\hat{\boldsymbol{e}}_{\mathrm{x}} \in \mathbb{R}^3$ represents an x-axis unit vector and $\boldsymbol{R}_{\mathrm{z}}$, $\boldsymbol{R}_{\mathrm{y}}$ are rotation matrices around the z- and y-axes (respectively) of the wrist reference frame Σ_{wst}.

To ensure that the racket is correctly oriented during the hitting movements, the normal unit vector $\hat{n}$ must be continuously kept parallel to the inclined plane. This can be achieved by computing the gripper vector $\boldsymbol{r}_{\mathrm{g}}$ using

$$\boldsymbol{r}_{\mathrm{g}} = \boldsymbol{r}_{\mathrm{h}} \times (\boldsymbol{R}_{\mathrm{x}}^{\alpha}\, \hat{\boldsymbol{e}}_{\mathrm{z}}), \tag{4.19}$$

and extracting the joint angle θ_7, where α defines the plane inclination, $\boldsymbol{R}_{\mathrm{x}}$ is a rotation matrix about the world frame x-axis, and $\hat{\boldsymbol{e}}_{\mathrm{z}} \in \mathbb{R}^3$ represents a z-axis unit vector (Figure 4.13).

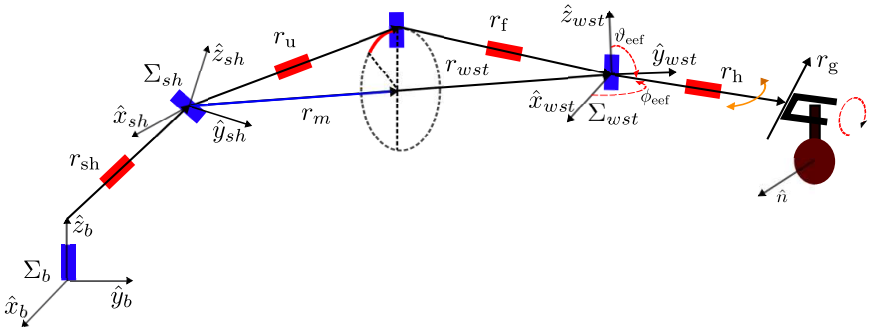


Figure 4.13: The hand segment $\boldsymbol{r}_{\mathrm{h}}$ is controlled separately to perform the hitting movements while the joint angle θ_7 ensures the correct orientation of the racket.

To respect the manipulator's task 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°.

4.1.4 Reference frames and transformations

Mainly, three relevant reference frames are considered in this application :

1. The inclined plane reference frame: in which the ball is tracked and its trajectory predicted to estimate the perceptual variables $\boldsymbol{x}_{\mathrm{hp}}$, $b_{\mathrm{reachable}}$, and b_{tim}.

2. The world coordinate system: centered at the manipulator base and used to define the task space in which the end-effector trajectories are generated.

3. The joint space: characterized by the 8 joint angles of the redundant manipulator CoRA.

The link between the frames (1) and (2) is established using a straightforward coordinate transformation. Frames (2) and (3) are linked through a forward kinematics formulation and its inverse kinematics based on the exact solution for the 8 DoFs CoRA manipulator. During arm movements, coordinates transformation between the inclined plane frame (1) and the task coordinate system (2) are continuously updated to compute the desired joint angles through the inverse kinematics transformation.

4.2 Dynamical model for the robotic hitting task

The dynamical model described in Chapter 3 is extended to generate the sequence of timed movements described in Section 4.1.1. This illustrates the inherent

flexibility of the proposed architecture to generalize to more complex robotic tasks. If compared to the simulated catching (see Section 3.4), the robotic hitting task requires the coordination of more movement components for the end-effector (x_{eef}, y_{eef} and ϕ_{eef}). Moreover, the timed behaviors are defined along different task related metric dimensions.

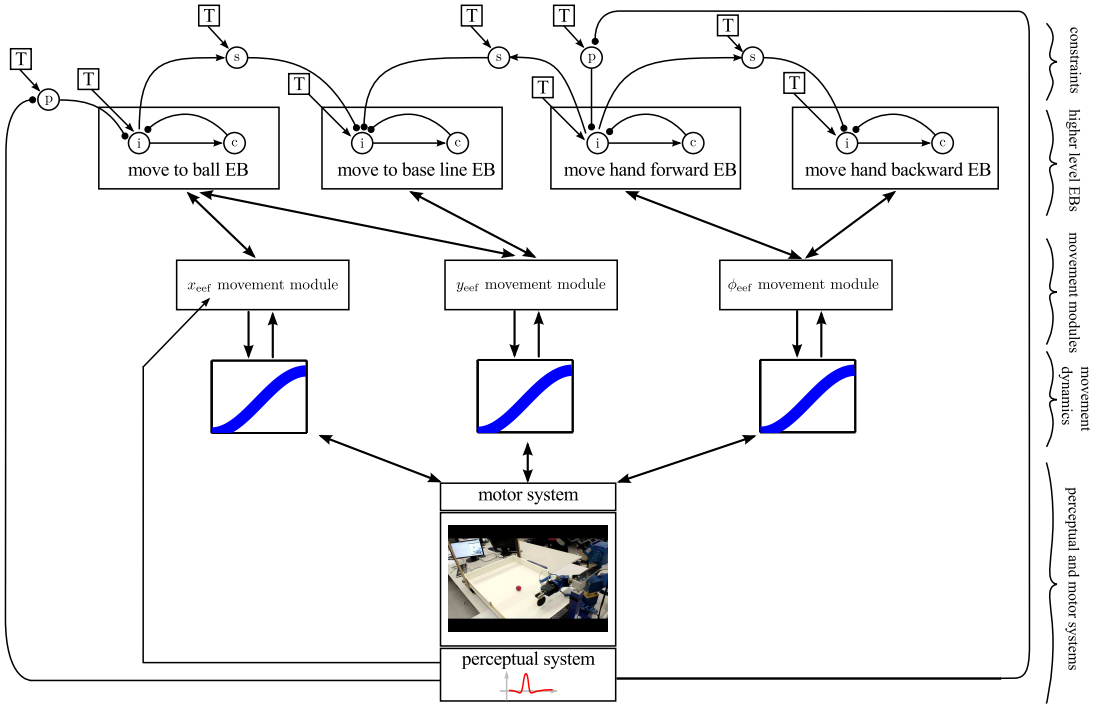


Figure 4.14: The neural dynamics architecture used to generate the sequences of timed movements for the robotic hitting task.

Figure 4.14 depicts the neural dynamics architecture for the robotic hitting task. The standard design of the timing dynamics as well as of the movement module used for the low level behavioral organization, allows their multiple use with the three movement components that define the end-effector state. The higher level EBs model the task movements by activating or deactivating the lower level movement modules. The sequencing mechanism between the timed behaviors is tiedly coupled to the ball perceptual information extracted by the robot's visual system (Section 4.1.2). Furthermore, behavioral constraints set between the EBs permit the autonomous organization in time and space of the task movements.

4.2.1 Higher level elementary behaviors

Four higher level EBs are required to model the end-effector movements as shown in Figure 4.15.

- 'Move to ball' EB: brings the end-effector x_{eef} and y_{eef} positions toward the predicted hitting point $\boldsymbol{x}_{\text{hp}}$.
- '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 successful hit or a failure.
- '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 with the 'Move hand forward' and 'Move hand backward' EBs for the ϕ_{eef} movement module (Figure 4.14). This shows the flexible use of the movement modules. Note also the suppression constraint set between the 'Move hand backward' and "Move to base line" EBs to ensure that the 'Move to base line' EB can be initiated only if the hitting action has been accomplished. 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.

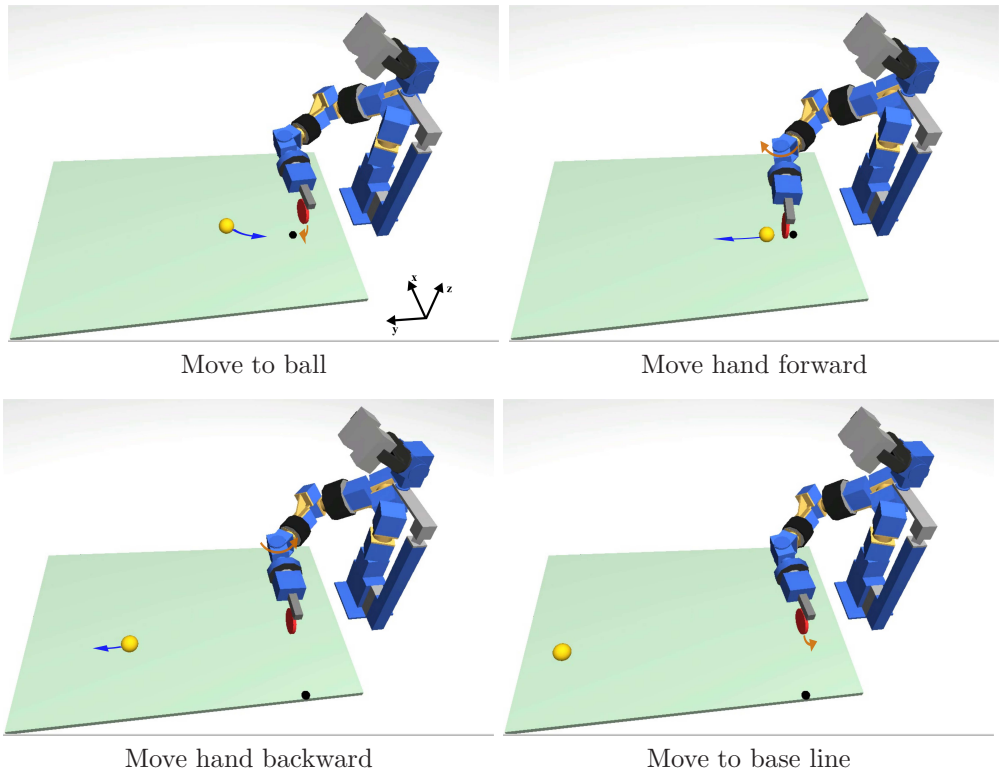


Figure 4.15: Movements of the racket during the robotic hitting task.

4.2.2 Movement modules

A movement module is defined for every movement component of the end-effector (x_{eef}, y_{eef} and ϕ_{eef}) and permit to generate the required timed trajectory (see Section 3.3.1 for more details). The target location of the x_{eef} movement is the predicted $\boldsymbol{x}_{\text{hp}}$ that is provided directly by the perceptual field of the architecture, introduced further.

The movement targets of the y_{eef} and ϕ_{eef} components are predefined and encoded separately into two external *update modules* (not shown in Figure 4.14). A necessary arrangement imposed by the shared use of each of these movement

modules by two higher level EBs with different targets. Every update module is composed of two dynamical fields, each of which receive a sub-threshold Gaussian input located at one of the two possible targets (hitting or base lines for y_{eef} and initial or final hand orientation for ϕ_{eef}). Additionally, every dynamical field receives a homogeneous boost from the intention node of the associated higher level EB. When a specific higher level EB is activated, a peak activation forms at the corresponding target location and is propagated to the movement module (more precisely to the intention field of the 'move' EB) to execute the required movement.

The tracking movement along the hitting line is obtained by coupling the fixed point attractor of the x_{eef} timing dynamics to the ball position along x-axis.

4.2.3 Perceptual and Motor systems

The complete autonomy of the dynamical model during task execution is ensured by an on-line coupling to the ball perceptual information. For that purpose, a perceptual field, defined over the hitting line metric dimension, is used to encode this information. The field receives a sub-threshold Gaussian activation centered at the hitting point location $\boldsymbol{x}_{\text{hp}}$ and a homogeneous boost from the reachability criterion $b_{\text{reachable}}$ through a sigmoid function.

In the architecture, the coupling is realized via a perceptual constraint in the form of a precondition dynamical node that is active by default and inhibits the 'Move to ball' EB (meaning the EB intention node). During task execution, the perceptual field forms a peak activation localized at $\boldsymbol{x}_{\text{hp}}$ as soon as the ball is predicted to be inside the robot's hitting region ($b_{\text{reachable}} = 1$). The peak activation turns off the perceptual precondition which releases the 'Move to ball' EB inhibition and permits to start the end-effector (x_{eef} and y_{eef}) movement toward $\boldsymbol{x}_{\text{hp}}$. Moreover, an additional perceptual precondition permits to inhibit the 'Move hand forward' EB until the b_{tim} reaches a predefined time threshold t_{thr} to initiate the hand segment movement and perform the hit ($b_{\text{tim}} < t_{\text{thr}}$ and $b_{\text{tim}} > 0$).

If the ball is successfully hit back up the inclined plane ($b_{\text{tim}} > t_{\text{thr}}$ and $b_{\text{reachable}} = -1$) or missed and falls down ($b_{\text{tim}} = -1$ and $b_{\text{reachable}} = -1$), the perceptual field looses its peak activation and both perceptual preconditions turn on. Consequently, the 'Move to ball' and 'Move hand forward' EBs gets inhibited which activate the 'Move hand backward' and 'Move to base line' EBs to bring the hand segment back to the resting posture and the end-effector back to the base line (respectively).

The generated task trajectories are converted into joint angles through the inverse kinematics closed form solution (see Section 4.1.3) to be executed autonomously by the robotic agent.

4.2.4 Stroke movement

After every hit, the ball should be directed toward the inclined plane's middle top with the aim to facilitate the next interception and achieve the 'keep the ball always in play' goal. Ideally, the ball landing position (or hitting point $\boldsymbol{x}_{\text{hp}}$)

should be brought after every hit closer to the hitting line center, and so, as far as possible away from the safety margins on both sides.

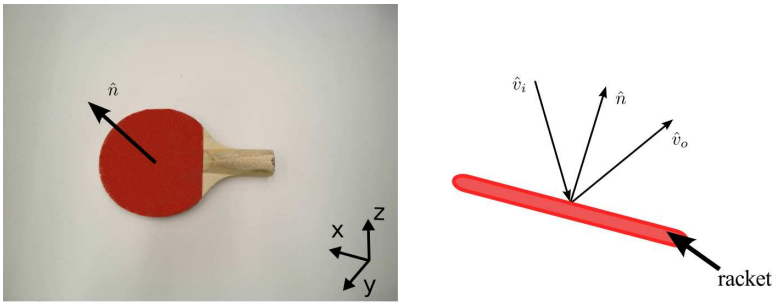


Figure 4.16: Orientation of the racket during hitting movements.

To derive the required racket normal vector $\hat{\boldsymbol{n}}$ so that the outgoing ball direction after a hit $\hat{\boldsymbol{v}}_{\mathrm{o}}$ is correctly oriented, we can use the reflection law

$$\hat{\boldsymbol{n}} = \frac{\hat{\boldsymbol{v}}_{\mathrm{o}} - \hat{\boldsymbol{v}}_{\mathrm{i}}}{\sqrt{2\,(1 - \hat{\boldsymbol{v}}_{\mathrm{i}}^{T}\,\hat{\boldsymbol{v}}_{\mathrm{o}})}}, \tag{4.20}$$

where $\hat{\boldsymbol{v}}_{\mathrm{i}}$ is the ball incoming velocity predicted at $\boldsymbol{x}_{\mathrm{hp}}$. From $\hat{\boldsymbol{n}} = [n_x\,,\,n_y]^T$ we can compute the desired racket orientation $\phi_{\mathrm{des}} = \arctan(n_y/n_x)$ at the hitting point. To obtain the required orientation ϕ_{des}, the cycle time T_{hit} of the 'Move hand forward' EB is updated during the first half cycle of the movement using

$$T_{\mathrm{hit}} = \frac{2\,r_{\mathrm{hit}}\,b_{\mathrm{tim}}}{d(t)}, \tag{4.21}$$

where $d(t) = |\phi_{\mathrm{des}} - \phi_{\mathrm{eef}}|$ is the current angular distance to the desired orientation ϕ_{des}. The update rule of Eq. 4.21 adapts the movement's profile (by accelerating or decelerating the dynamics) to realize the desired orientation at the hitting moment. Consequently and after few consecutive hits, the ball is brought to the middle of the hitting line along x-axis as shown in Figure 4.17.

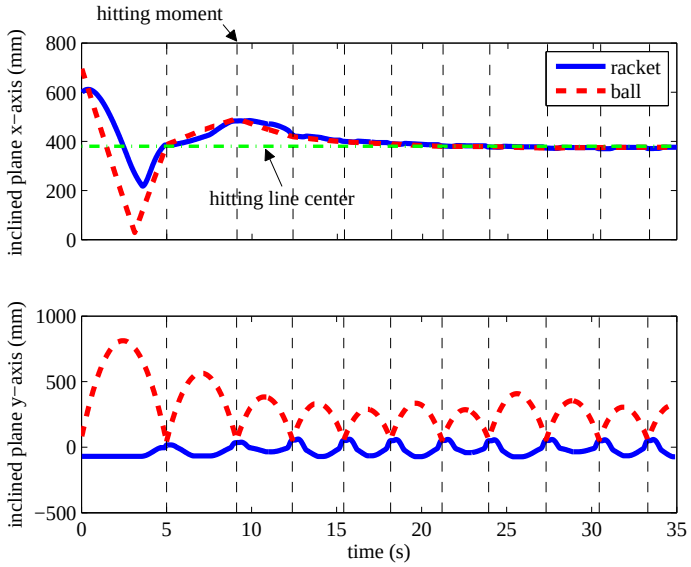
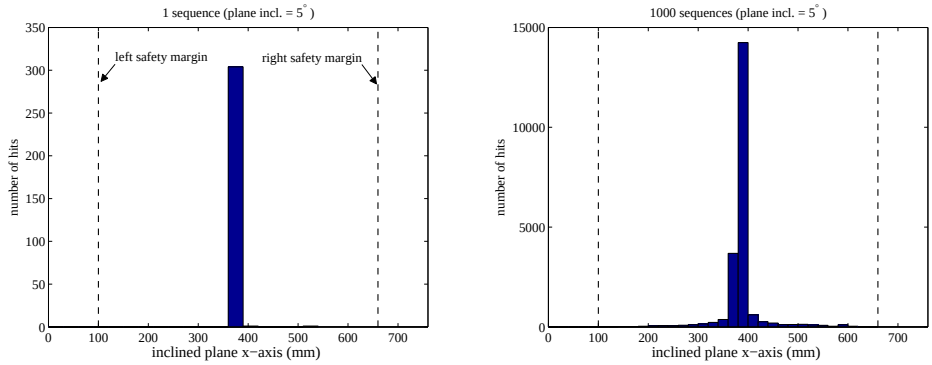


Figure 4.17: Trajectories of the ball and the racket during several consecutive hits in simulation. The ball is brought to the middle of the hitting line to facilitate future interceptions and keep the ball always in play.

The performance of the stroke movement can also be well demonstrated using a multiple trials scenario in a simulated setup. The trials were organized in the form of several (1000) hitting sequences. During every sequence, the robot keeps hitting the ball continuously until a failure or a miss occurs which leads to start the next sequence. Before starting any sequence, the ball was re-launched on the inclined plane with a random speed and launching angle.

As shown in the histograms of Figure 4.18, the derived reflection rule and the timed hitting movements permitted efficiently to accomplish the goal.



(a) (b)

Figure 4.18: Histogram plots to demonstrate the performance of the reflection rule in simulation during multiple trials scenario. It can be clearly seen that most of the ball hits occurred approximately at the middle of the inclined plane x-axes.

While the desired racket orientation can be obtained by timing the hitting movement, the mechanism doesn't ensure a maximum speed at the hitting instant in order to provide the ball with maximum momentum. To overcome that, the

initiation time of the movement can be tuned so that the desired orientation can be obtained *approximately* at a peak speed while respecting the hardware limitations of the robot. The angular distance $\Delta\phi$ traveled during a hitting time Δt and inside the movement range $\zeta = 10°$ of the azimuth orientation ϕ_{eef} can be approximated by

$$\Delta\phi = \frac{\zeta}{2}\left(\cos(\omega\Delta t) - 1\right), \tag{4.22}$$

where $\omega = \frac{2\pi}{T'_{\text{hit}}}$ defines a pre-selected cycle time T'_{hit} for the hitting component ϕ_{eef} and $\Delta t = t - t_{\text{init}}$ is the duration between the current time t and the initiation instant t_{init}. From Eq. 4.22, the desired racket orientation ϕ_{des} can be written as

$$\phi_{\text{des}} = \phi_{\text{init}} + \frac{\zeta}{2}\left(1 - \cos(\omega\Delta t)\right), \tag{4.23}$$

where $\phi_{\text{init}} = 95°$ is the initial orientation. Using Eq. 4.23 and given T'_{hit}, we can extract the time Δt needed to reach the desired orientation by

$$\Delta t = \frac{T'_{\text{hit}}}{2\pi}\arccos\left(\frac{2\left(\phi_{\text{des}} - \phi_{\text{init}}\right)}{\zeta} + 1\right). \tag{4.24}$$

Δt represents the shortest time required to reach ϕ_{des} given a cycle time T'_{hit}. T'_{hit} can be tuned to respect the hardware speed limitation of the robot manipulator. To obtain the hardest hit on the ball while respecting the robot's constraints, the time threshold t_{thr} (see Section 4.2.3) is set to Δt and the hitting movement is initiated when $b_{\text{tim}} < t_{\text{thr}}$. This disposition ensures that small angular distances $(\phi_{\text{des}} - \phi_{\text{init}})$ are executed in shorter periods compared to larger distances. Additionally, the system permits to have a variable initiation time depending on the prediction of the desired racket orientation ϕ_{des} at the hitting point. However, this approximation had a limited effect during task execution due to the small movements range ($\zeta = 10°$) and the updating cycle time T_{hit}.

Figure 4.19 depicts the first three hitting movements of Figure 4.17 as racket speed and orientation. It can be clearly seen that the hit occurs almost every time at peak speed.

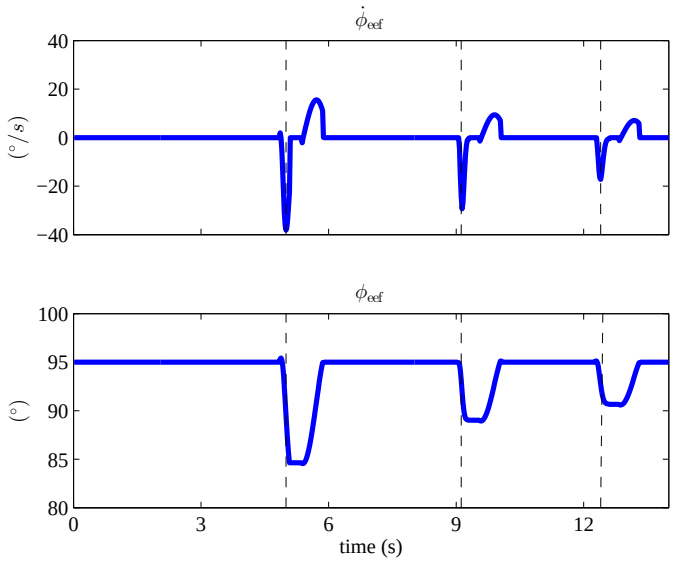


Figure 4.19: Trajectory of the racket as speed and orientation during three consecutive hits in simulation. The hit occurs at peak speed almost every time.

4.3 Results

In this section, the dynamical model is evaluated both in simulation and in a hardware implementation of the robotic hitting task. The obtained results illustrate that the robot performed successfully the required task. Moreover, the robot was able to react autonomously and flexibly to different external perturbations on the ball trajectory due to obstacles placed on the inclined plane.

4.3.1 Simulation results

The following experimental results are obtained from a physically realistic Matlab simulation of the whole setup. The experiments include several perturbation scenarios in addition to statistics about the robotic task. The solutions of the continuous time dynamics that compose the model are approximated numerically using the Euer method.

Successfully hitting the ball

Figure 4.20 shows that the robot was able to hit the ball successfully.

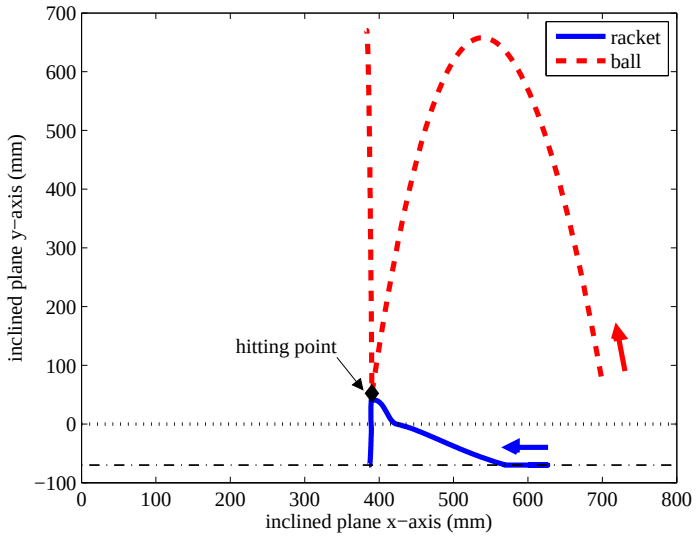


Figure 4.20: Trajectories of the ball and the racket for a successful hit.

The detailed time course of the relevant variables and parameters is shown in Figure 4.21. At $t = 0\,\mathrm{s}$, the ball is launched upwards from the bottom of the inclined plane. At $t \approx 2.56\,\mathrm{s}$, the ball starts rolling down and the visual system provides a prediction of the hitting point $\boldsymbol{x}_{\mathrm{hp}}$ and time-to-impact b_{tim}, forming a peak in the perceptual field. The intention node of the 'move to ball' EB (abbreviated by 'mtb') turns on and drives the end-effector x_{eef} and y_{eef} positions toward the predicted $\boldsymbol{x}_{\mathrm{hp}}$. At this phase, you can clearly see the stability and robustness of the system in presence of fluctuating sensory information due to noise so that the movement is not interrupted or restarted. As the ball approaches the hitting point and the current time-to-impact falls bellow the variable threshold t_{thr} (depicted by the blue line) at $t \approx 3.95\,\mathrm{s}$, the 'move hand forward (mhf)' EB gets activated and starts a timed movement for the racket orientation ϕ_{eef} to hit the ball at $t \approx 4.42\,\mathrm{s}$. The hit drives the ball back up the inclined plane, removes the ball prediction, and leads the perceptual field to loose its peak activation. The intention nodes of the EBs 'move hand backward (mhb)' and 'move to base line (mtbl)' switch on and initiate the movements that drive ϕ_{eef} back to the initial orientation and x_{eef} and y_{eef} back to the initial posture. At the same time, the end-effector starts tracking the ball horizontal motion.

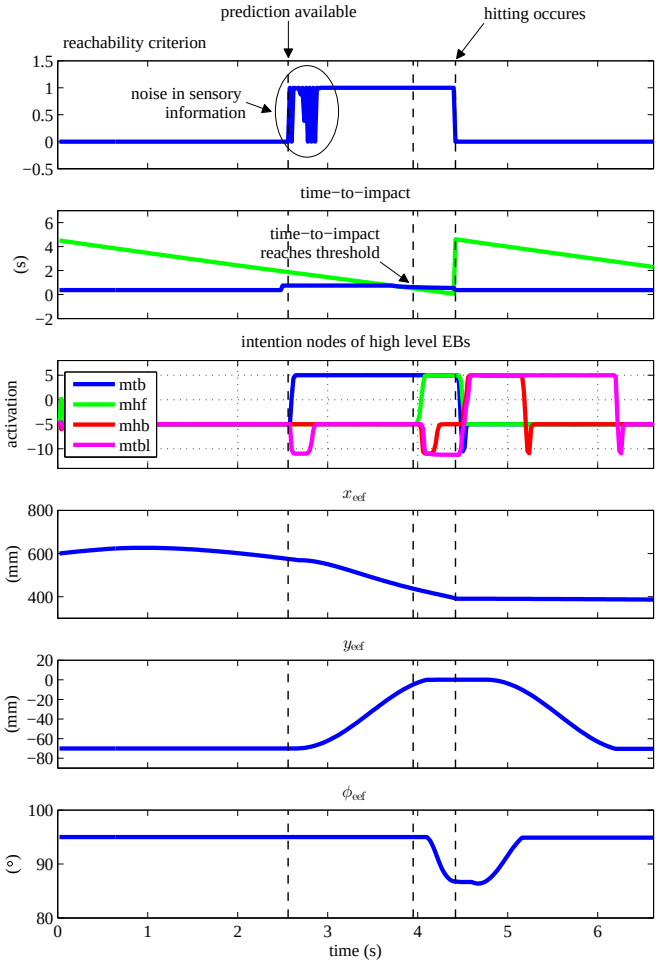


Figure 4.21: Trajectories of the significant parameters and variables during a successful hit and return to the base line. The top three plots represent the ball reachability criterion as extracted from the perceptual field, time-to-impact, and time courses of the higher level EBs intention nodes. The bottom three plots illustrate the timed end-effector trajectories.

Hitting abortion after ball reflection

In the first perturbation scenario, shown in Figure 4.22, the ball is reflected by an obstacle while the racket is moving toward the predicted $\boldsymbol{x}_{\text{hp}}$. The robot autonomously reacted to this unexpected change by interrupting the movements sequence and initiating a fixed time movement back to the initial posture ready to initiate the next hitting actions.

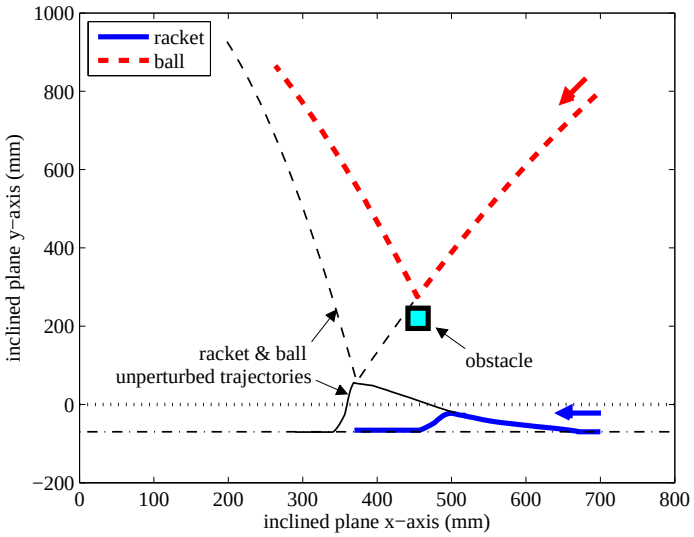


Figure 4.22: Trajectory of the racket when the ball is reflected by an obstacle during the racket movement. The thin black lines show the unperturbed trajectories of the ball and the racket.

Figure 4.23 depicts the time courses of the most relevant variables during the experiment. The direct coupling to sensorial information about the ball motion allowed the robot to handle autonomously the perturbation.

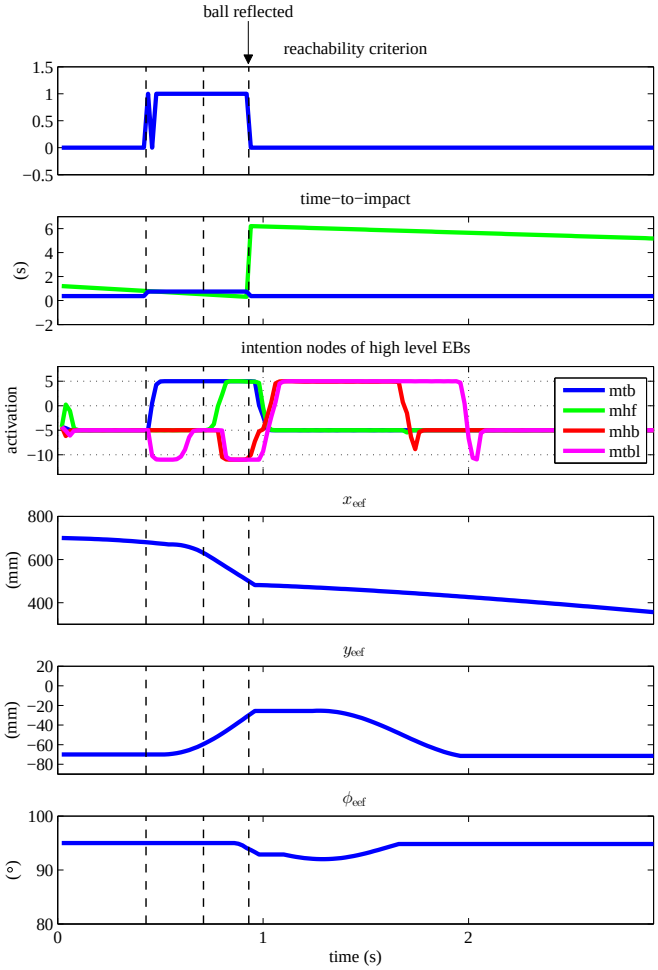


Figure 4.23: Trajectories of the relevant parameters and variables when the ball gets reflected by an obstacle during racket movement.

Reactivating a hitting sequence

The experiment result shown in Figure 4.24 illustrates the flexible reaction of the robot in reactivating, while in the backward movement, a supplementary hitting sequence if the initial one was not sufficient to provide the ball enough momentum and bring it sufficiently far up the inclined plane.

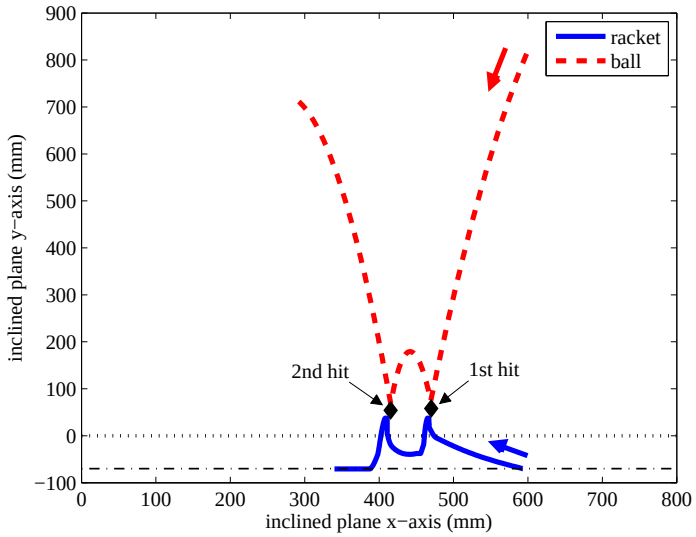


Figure 4.24: A new hitting movement sequence is re-initiated while the racket is moving back to the initial posture.

As illustrated in Figure 4.25, the new hitting sequence is started from the current end-effector state before the robot reaches the resting posture. This demonstrates the system feature in flexibly reorganizing on the fly the timed behaviors in response to new sensorial contexts.

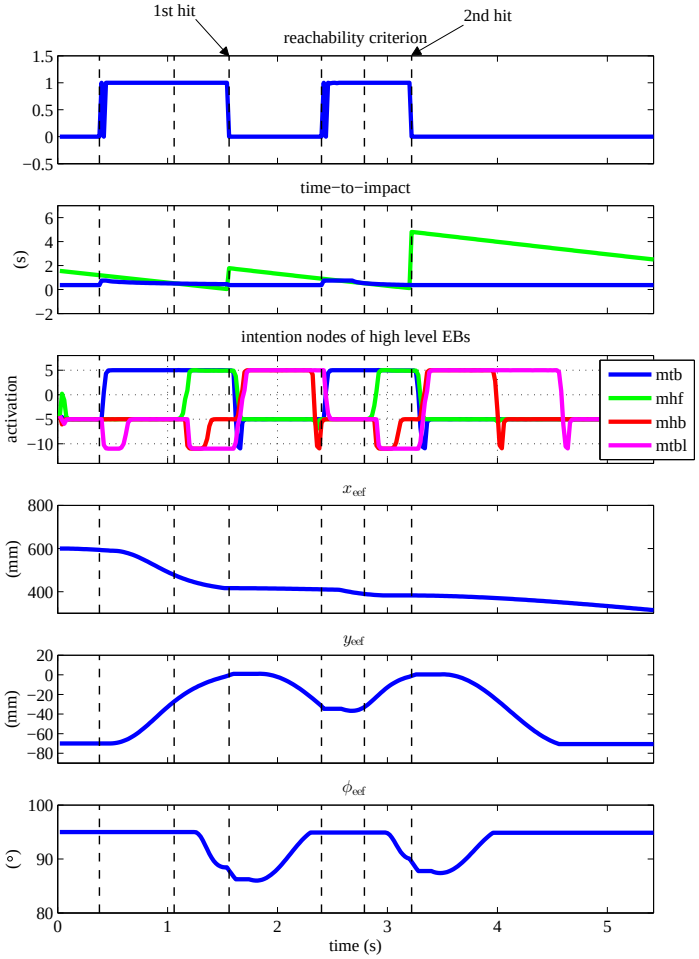


Figure 4.25: Time courses of the system's relevant variables and parameters during the re-initiation of a new hitting sequence.

Updating hitting movement after ball deviation

During this experiment, the ball trajectory is deviated by an obstacle during racket movement as illustrated in Figure 4.26. Consequently, the model updated the dynamics and the robot accelerated toward the new predicted $\boldsymbol{x}_{\text{hp}}$ with the aim to preserve the correct movement timing and successfully execute the hit. This reaction demonstrates the system capacity to update on-line the movement parameters when subjected to external disturbances that shift the spacial or temporal constraints of the timed behaviors.

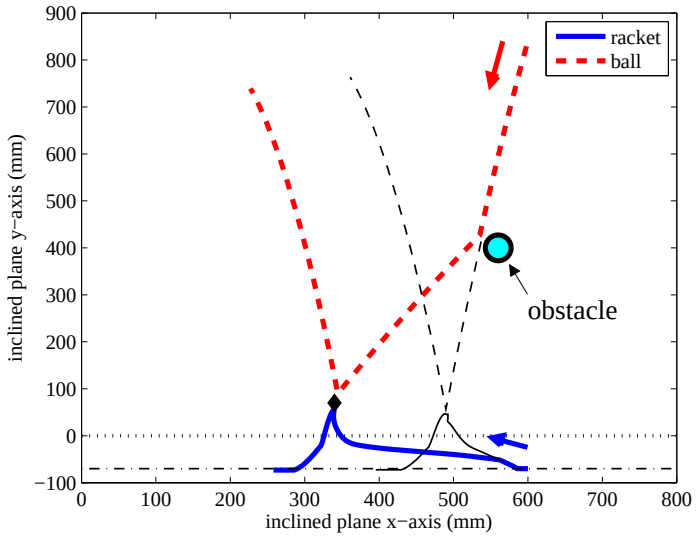


Figure 4.26: Trajectory of the racket when the ball is deviated by an obstacle during the racket movement. The thin black lines show the unperturbed trajectories of the ball and the racket.

Figure 4.27 depicts the system's relevant parameters and variables during the experiment.

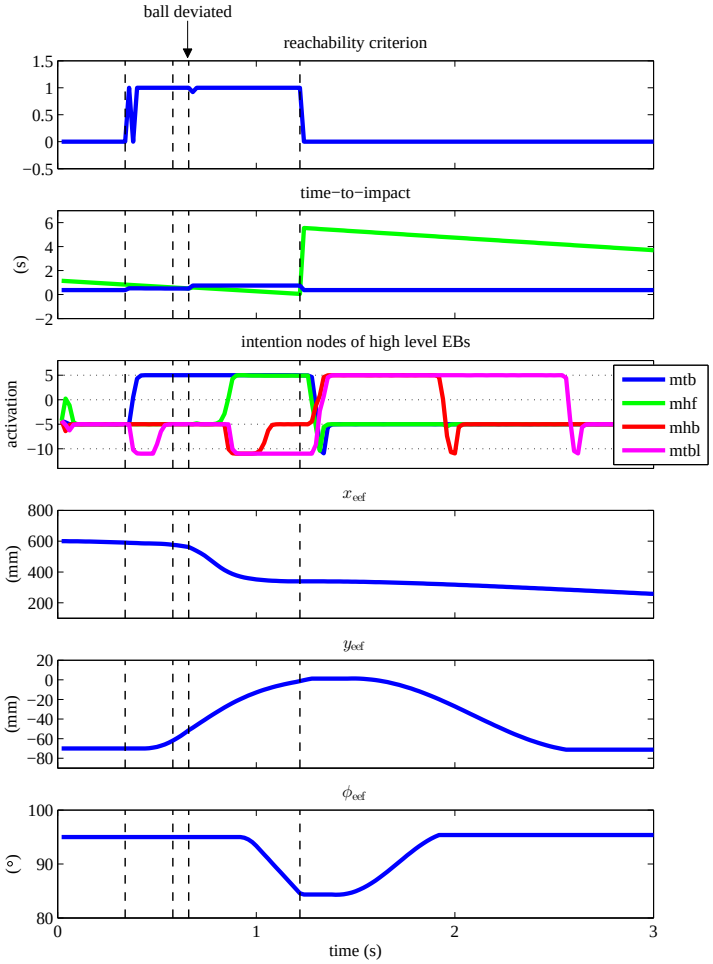


Figure 4.27: Trajectories of the system's relevant variables and parameters when the ball is deviated during the racket movement.

The perturbation occurs right after initiating the 'move to ball' EB at $t \approx 0.66\,\text{s}$. This leads to accelerate the movement along x-axis and decelerate along y-axis by updating the dynamics cycle times according to Eq. 3.3 (see Section 3.2.1) as can be seen in Figure 4.28.

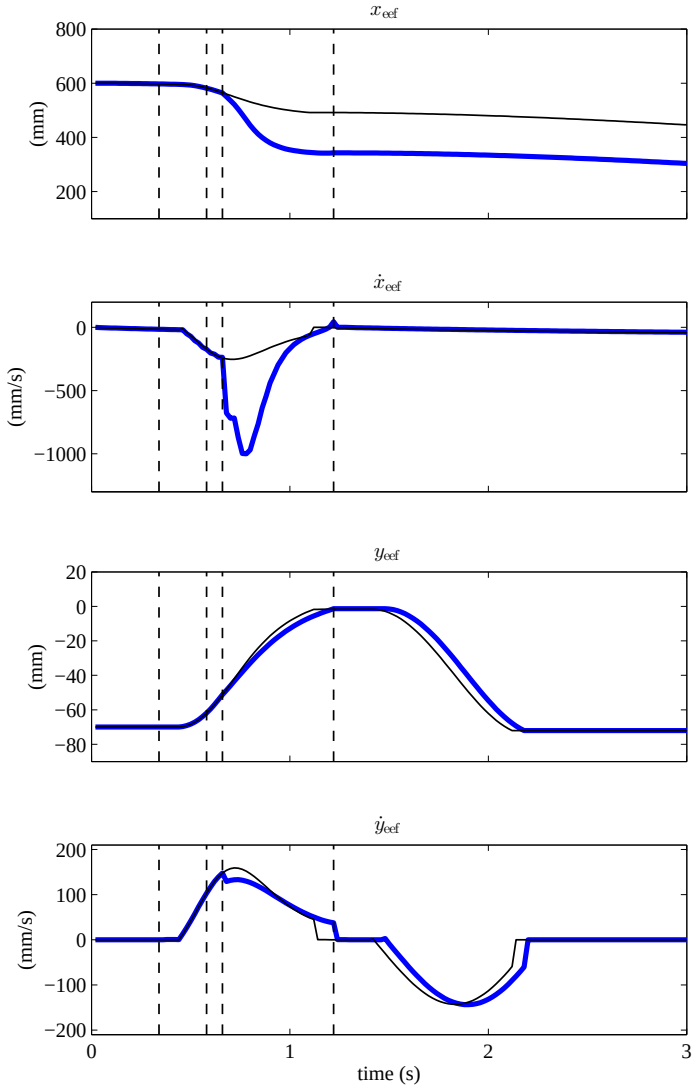


Figure 4.28: Trajectories of the racket along x- and y-axis as the ball is deviated during racket movement. The thin black lines show the unperturbed trajectories of the racket.

Flexibly reacting to multiple perturbations

The last scenario (Figure 4.29) combines multiple perturbations on the ball trajectory.

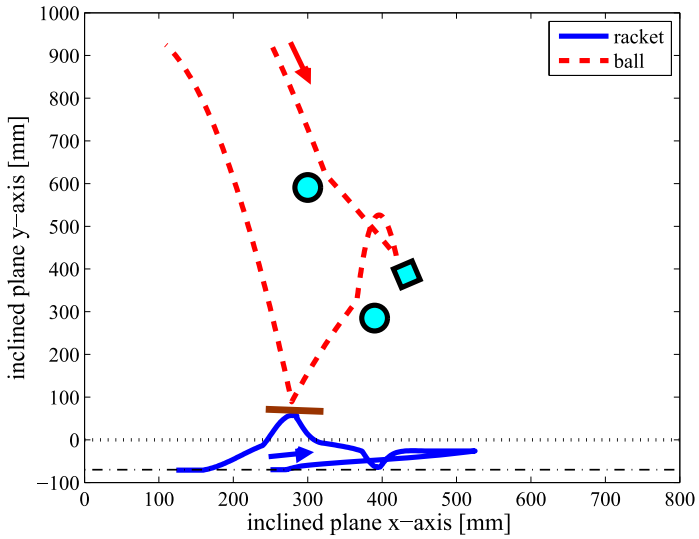


Figure 4.29: Trajectory of the racket when the ball is perturbed by many obstacles during the racket movement.

The robot was able to handle this complex situation by flexibly (re-)organizing the sequences of timed movements (Figure 4.30) and manages to adapt and successfully hit the ball at the hitting line.

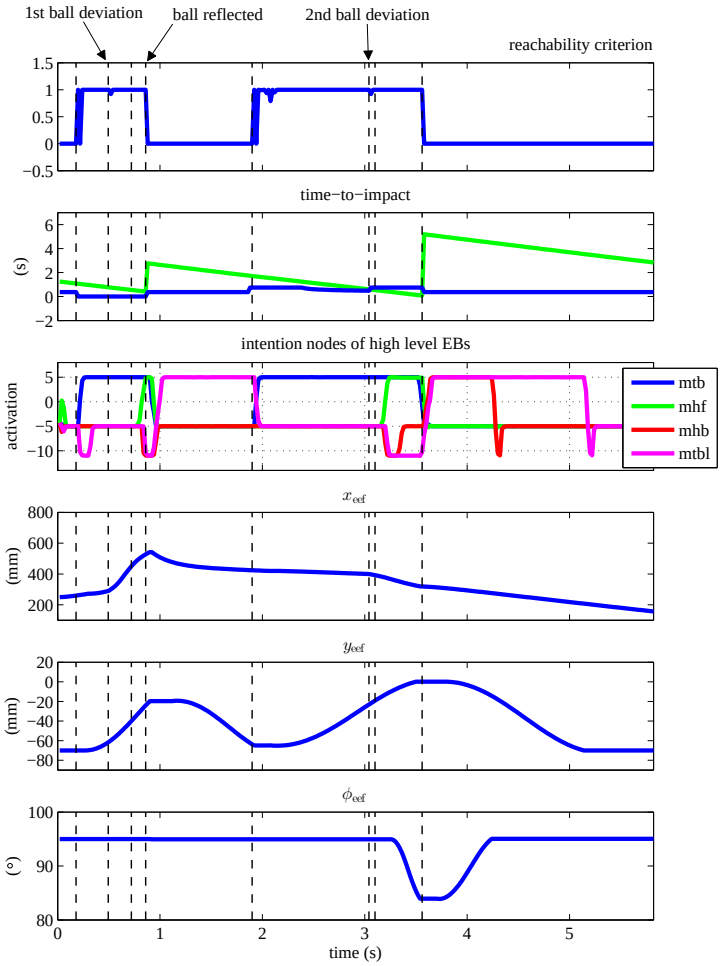


Figure 4.30: Time courses of the system's variables and parameters when the ball is perturbed by many obstacles during the racket movement.

Results of statistical tests

During a multiple trials scenario in simulation, statistics data has been gathered in order to evaluate the performance of the model. The test consisted of (re-)initiating a new sequence of hits after every failure or miss by injecting the ball on the inclined plane with random speed ($\in [0.6\,\mathrm{m\,s^{-1}}, 0.8\,\mathrm{m\,s^{-1}}]$) and launching angle ($\in [95°, 120°]$). In the following, the results of 1000 hitting sequence for two different plane inclinations are shown. If the ball lands on the safety margins along the hitting line, it is re-injected without starting a new sequence.

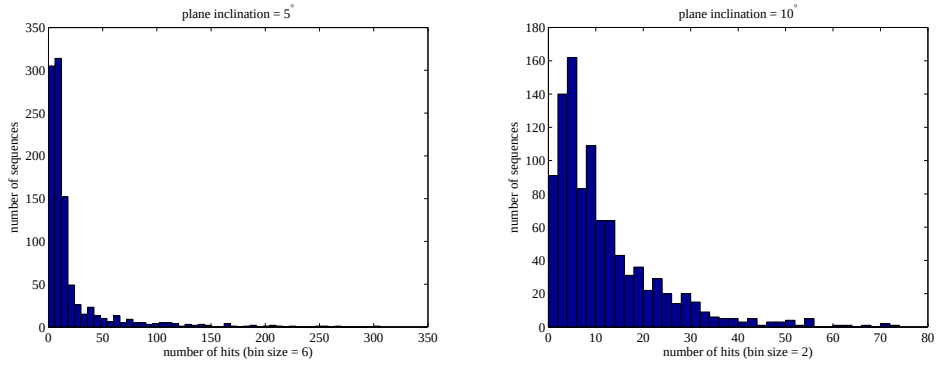


Figure 4.31: Histograms plot of the results obtained from a multiple trials scenario (1000 sequence).

The histograms of Figure 4.31 illustrate the robot performance in continuously hitting the ball back up the inclined plane and keeping it in play as long as possible.

Plane incl. (°)	Success rate (%)	Mean (hits)	Average dev. ball⇆racket (mm)
5	95.442	20.938	21.2
10	92.432	12.214	18

Table 4.1: Performance of the robot on successive ball hits in simulation.

Table 4.1 shows the most relevant statistical data of the experiment. It can be clearly observed that the success rate and the hits mean number decreases as the ball motion gets quicker after increasing the plane inclination. This decrease in performance can be explained by the diminished capacity of the model to react fast enough and accelerate the movements when the ball starts rolling faster. As discussed in Section 3.5, the states update phases and the switching dynamics induce a systematic delay (almost fixed in average) before starting any behavior. Consequently, adapting to a fast rolling ball will require generating very rapid movements that are generally harder to update on the fly. This is due to the dynamics reaction time as can be observed in the average deviation between the racket and ball positions at the hitting moments. Additionally, flexibly reorganizing the timed behaviors in response to perturbations of the type discussed earlier will take too much time leading to increase the number of failures.

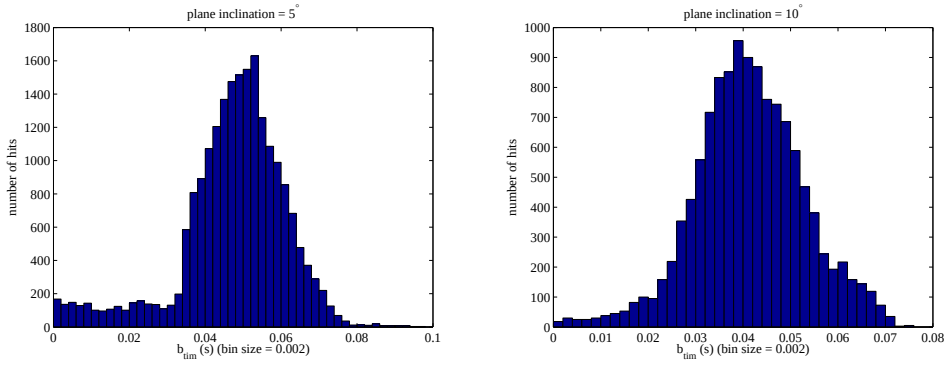


Figure 4.32: Histograms plot of the ball time-to-impact b_{tim}, recorded at every hitting moment (1000 sequence).

Figure 4.32 shows histograms of the ball time-to-impact b_{tim} recorded when the hitting occurs. Ideally, the ball should be hit at $b_{\text{tim}} = 0.0\,\text{s}$. The histograms illustrate how precise the timing of the movements was and reveal that the mean b_{tim} for the two plane inclinations (5° and 10°) is respectively ($b_{\text{tim}} \approx 0.0474\,\text{s}$ and $b_{\text{tim}} \approx 0.0413\,\text{s}$) which confirms the effect of the plane inclination on movement generation discussed earlier.

4.3.2 Hardware implementation results

The dynamical model has been successfully ported to the real manipulator CoRA. The test in the hardware platform suffered from several implementation problems like the limited speed and precision of the robotic arm movements. In addition, the low frame rate (30 frames/s) of the robot's camera affected ball tracking and prediction.

Figure 4.33 shows the ball and racket trajectories during a successful hit by the robot manipulator.

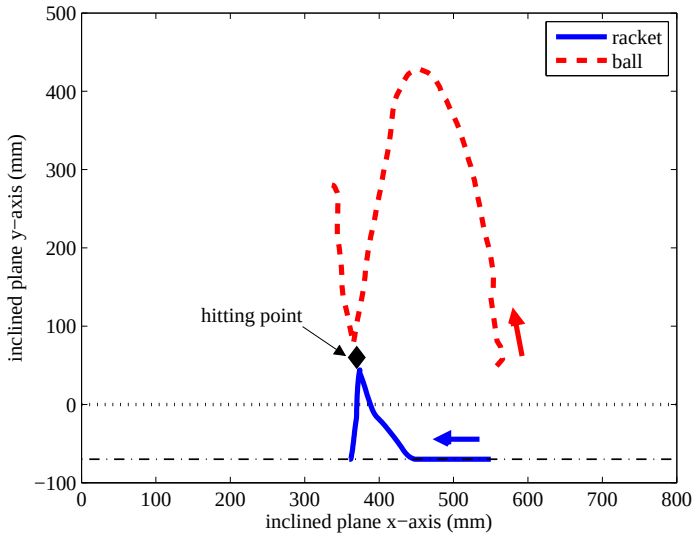


Figure 4.33: Trajectories form the hardware implementation of the ball and the racket during a successful hit.

Trajectories of the significant parameters and variables during the successful hit are depicted in Figure 4.34. It can be clearly seen how direct coupling to noisy perceptual information is used to initiate, steer, and terminate to different actions which illustrates the stability and robustness of the dynamical model.

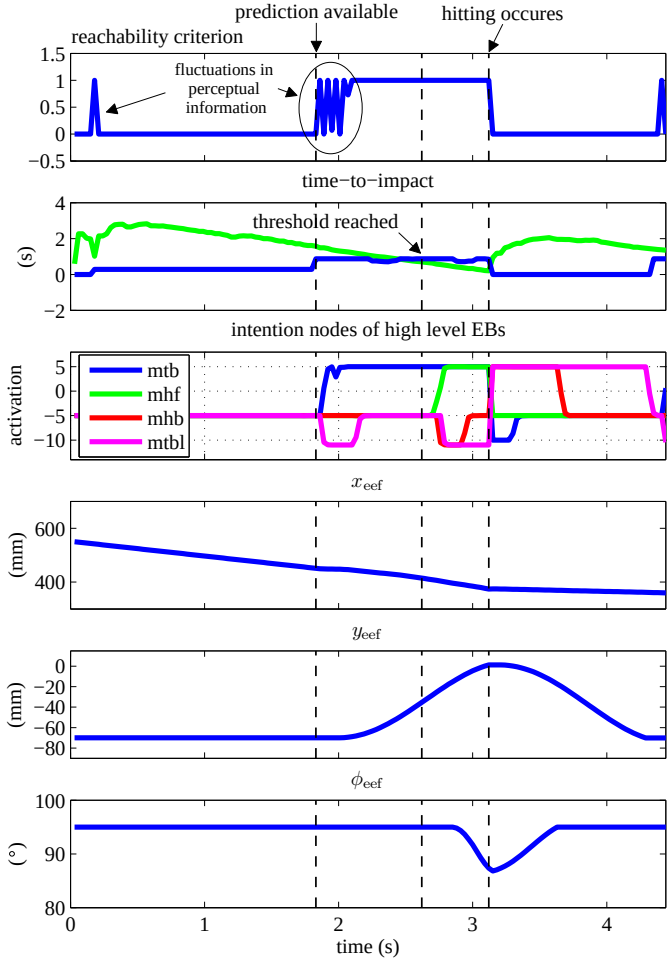


Figure 4.34: Trajectories form the hardware implementation of the significant parameters and variables during a successful hit. The top three plots represent the ball reachability criterion as extracted from the perceptual field, time-to-impact, and time courses of the higher level EBs intention nodes. The bottom three plots illustrate the timed end-effector trajectories.

Figure 4.35 depicts snapshots of the robot manipulator during a successful ball hit.

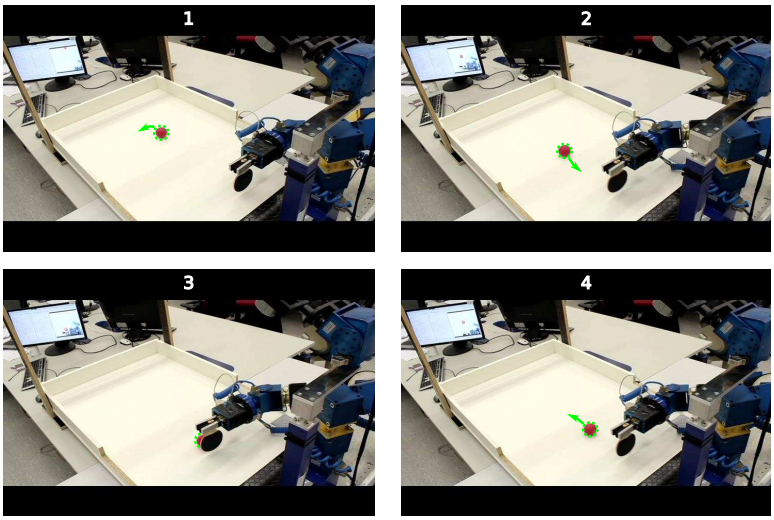


Figure 4.35: Snapshots of the robot manipulator during a successful ball hit. (1) Ball tracking before initiating the hitting sequence. (2) Initiating the hitting movements through the 'Move to ball' EB. (3) Executing the hit by the 'Move hand forward' EB. (4) Returning to the base posture by the 'Move hand backward' and 'Move to base line' EBs.

The robot was able to perform several consecutive hits and keep the ball in play for several minutes (the robot had to be stopped after $\approx 5\,\text{min}$ to prevent any damage). Figure 4.36 shows sample trajectories of the robot and the ball for seven successive hits. As can be observed, the robot tried to keep the ball on the middle of the inclined plane's x-axes by timing the stroke movements (see Section 4.2.4 for more details). Nevertheless, the robot failed to do so and the ball was continuously balanced around the hitting line center. This illustrates that the performance of the update rule was considerably reduced in the hardware implementation due to the limited speed and movements precision of the real manipulator.

Statistical data from 25 hitting sequences are gathered in Table 4.2. The performance of the real robot is equivalent to the results obtained from simulation. Even though, the hardware limitations prevented us from conducting higher number of hitting sequences with larger plane inclinations.

Plane incl. (°)	Success rate (%)	Mean (hits)	Average dev. ball⇆racket (mm)
2	94.32	18.9	28.2
4	93.56	14.52	32.11

Table 4.2: Performance of the real robot on successive ball hits.

The hardware implementation has been also tested using perturbation scenarios equivalent to those performed in simulation (see Section 4.3.1). In Figure 4.37, the ball is reflected by an obstacle during racket movement toward the predicted hitting point $\boldsymbol{x}_{\text{hp}}$. Consequently, the robot aborted the hitting sequence and started a backward movement to the initial posture.

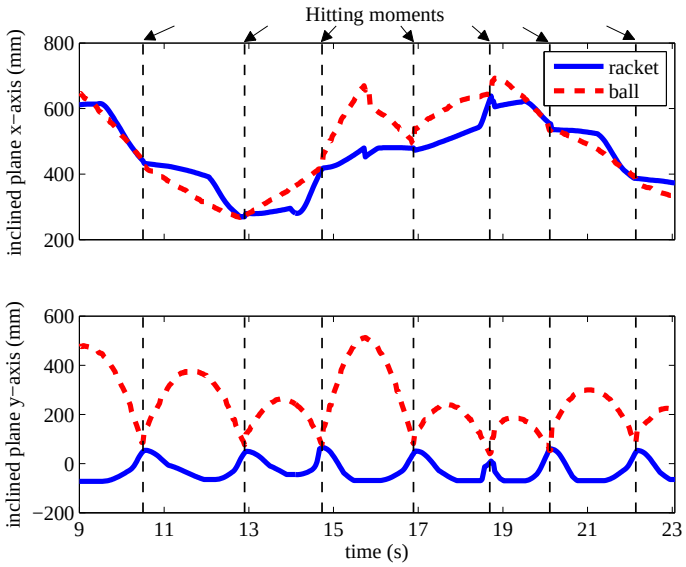


Figure 4.36: Sample trajectories of the real robot and the ball for seven successive hits.

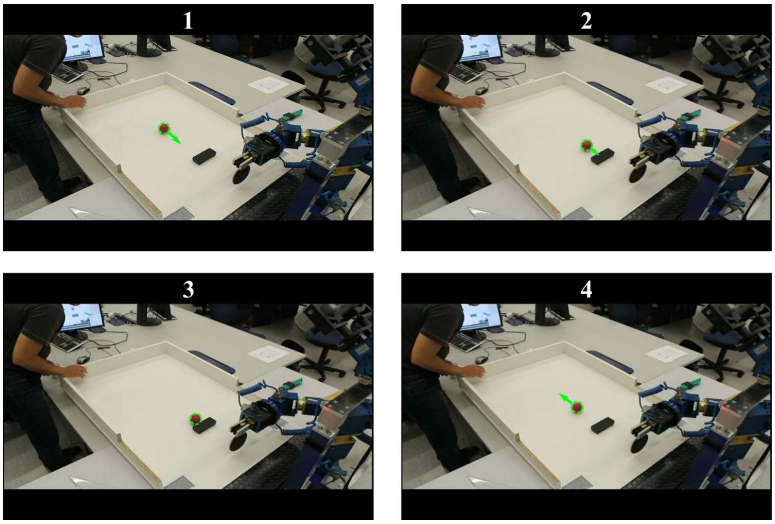


Figure 4.37: Snapshots of the robot as the ball is reflected by an obstacle during racket movement. (1) The ball is introduced in the scene and starts rolling down. (2) A prediction of the hitting point $\boldsymbol{x}_{\text{hp}}$ is provided and the hitting sequence is started. (3) The ball is reflected up on the way by an obstacle. (4) A backward movement to the initial posture is initiated.

In the next experiment (see Figure 4.38), the ball is first reflected by an obstacle during racket movement before it starts rolling down with a new $\boldsymbol{x}_{\text{hp}}$ prediction. As a result, the robot re-initiated a new hitting sequence toward the new prediction and successfully executed the hit.

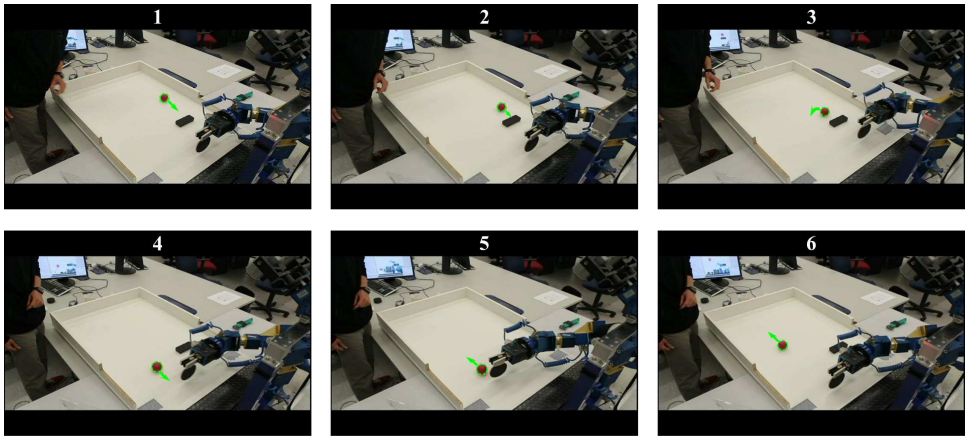


Figure 4.38: Snapshots of the robot manipulator during a hitting sequence re-initiation. (1) The ball is introduced in the scene and starts rolling down. (2) A hitting sequence toward $\boldsymbol{x}_{\text{hp}}$ prediction is started. (3) The ball is reflected by an obstacle but starts rolling down again somewhere on the left side. (4) A new hitting sequence is initiated in the direction of the new $\boldsymbol{x}_{\text{hp}}$ prediction from the current end-effector state. (5) The ball is successfully hit up the inclined plane by the robot. (6) Backward movements to the initial posture are initiated.

For the last experiment shown in Figure 4.39, the ball is deviated by an obstacle during racket movement. In response to that, the model updated the movement dynamics and the robot accelerated toward the new $\boldsymbol{x}_{\text{hp}}$ prediction and successfully hit the ball.

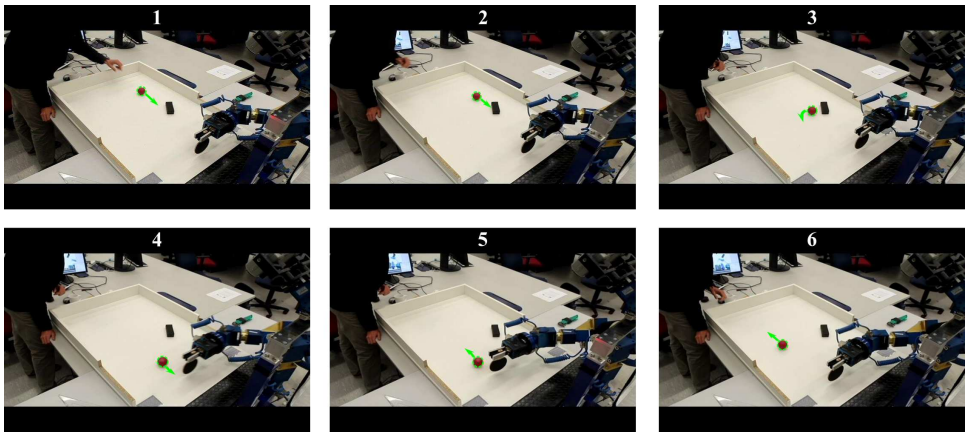


Figure 4.39: Snapshots of the robot manipulator as the ball is deviated by an obstacle during racket movement. (1) The ball is introduced in the scene and starts rolling down. (2) A hitting sequence is initiated in the direction of the $\boldsymbol{x}_{\text{hp}}$ prediction. (3) The ball is deviated to the left by an obstacle. (4) The robot updates on the fly the movement parameters and accelerates toward the new $\boldsymbol{x}_{\text{hp}}$ prediction. (5) The robot succeeded to catch up and successfully hit the ball. (6) The robot is going back to the initial posture.

4.4 Discussion

In this chapter, I have demonstrated the core properties of the dynamical model for timed movements sequence generation in a robotic hitting task. The robotic application entails coordinating multiple different actions in time and space with a ball rolling on an inclined plane. The architecture is directly coupled to the task environment through perceptual parameters and variables that encode sensorial events and are computed algorithmically by the visual system. The extension of the model was easy due to the standard design of the different elements that compose the architecture.

In simulation, the evaluation results showed that the robot succeeded to achieve the task goal: continuously hit the ball and keep it in play. Moreover, the system was able to react autonomously in response to external perturbations on the ball by initiating, aborting, or re-initiating inactive behaviors at any time. Furthermore, the system could update the movement parameters such as the amplitude and movement time on the fly when such perturbations shift the timing and spatial constraints. However, I have observed that the model performance was directly influenced by the reaction time of the dynamics. The effect of this influence significantly increased when faster movement sequences are involved, for example when the ball motion speed was incremented by elevating the plane inclination.

For the hardware test, the dynamical model has been successfully ported onto a real manipulator. The robot was able to hit the ball continuously for several minutes. In addition, equivalent performance to that obtained from simulation were demonstrated during similar perturbation scenarios. Nevertheless, the implementation suffered from numerous limitations. These include the modest maximal speed of the robot, the large inertia of the 8 DoF manipulator that affected the overall reaction time of the system, and the lack of precision during movements. Moreover, the robot's camera low frame rate affected ball tracking and prediction during task execution.

The obtained results, both from simulation and hardware, illustrated the ability of the model to generate and flexibly organize sequences of timed movements in coordination with perceived events. Both the sequencing mechanism and the movement dynamics adapted to sensorial changes detected through noisy and fluctuating sensory information. These characteristics could be obtained only because the different states of the system were stabilized against perturbations which demonstrates clearly the robustness of the dynamical model.

Chapter 5

Conclusion

5.1 Summary of contributions

In this thesis, I investigated how different movement behaviors may be timed to sensory events while being sequentially and flexibly organized. The problem is directly inspired by the skills demonstrated by humans in a variety of tasks in every day life and particularly in ball sports and racket games.

For that purpose, I proposed a model for timed movements sequence generation. The model is built within the dynamical systems approach and consists of: (a) timing dynamics that generate movements through stable oscillations and postural states by fixed point attractors, and (b) a hierarchical architecture for behavioral organization in which higher level EBs model the timed behaviors and activate lower level modules to control the movement dynamics. Both layers are coupled online to time-varying sensory information and are able to adapt to perturbations either by updating the movement parameters or by flexibly reorganizing the sequence of behaviors according to the current sensorial context. Furthermore, mechanisms to stabilize the temporal structure of movements and compensate for noisy predictions have been developed and integrated in the system.

The properties of the dynamical model were assessed in simulation by two robotic tasks, catching and hitting a ball. In addition, the model has been successfully tested in a hardware implementation of a ball hitting task.

5.2 Overview of the results

To demonstrate the core features of the dynamical model, several experiments have been conducted with two robotic applications, catching and hitting a ball. The two robotic tasks require generating sequences of timed movements in coordination with perceived events from the ball motion. Both from simulation and hardware tests, the obtained results demonstrate the system ability to fulfill these constraints and perform both tasks with success. Moreover, the robot could react quickly and flexibly to several perturbation scenarios either by updating the movement profile or adapting the sequence of actions by aborting ongoing behaviors, and initiating or re-initiating inactive behaviors. These results and particularly those obtained from the implementation on a real robot reveal the

method's feasibility.

The dynamical model shows up several appealing properties, such as reusability of movement primitives and perception-action coupling for decision selection. In addition, the standard structure of the different elements that compose the model reduces the number of parameters and synaptic weights to be tuned and allows a simple extension to different robotic tasks that are, still, conceptually very close.

The movement dynamics of the model is completely sensor-driven and generate either postural states via stable fixed points to fixate the end-effector or goal-directed timed movements by means of periodic solutions along limit cycle attractors. The design assumes that complex movements can be generated through the sequencing and reuse of simpler movement primitives modeled as discrete and rhythmic dynamical systems. Additionally, the dynamics ensure that the end-effector executes the trajectories while sitting at all times in an attractor state. Moreover, the generated movements can compensate for varying predictions by continuously updating movement parameters such as amplitude and movement time while keeping timing stable. Being able to modify on the fly the movement plan to respect temporal constraints is crucial in this kind of robotic applications.

The autonomous organization of timed behaviors is performed by a DFT based neural dynamics architecture. The elementary behaviors (EBs) of the robot are hierarchically structured and dynamically organized into sequences based on task specific behavioral constraints and online perceptual information. In the architecture, low-level modules play the role of switching dynamics between the two operational regimes of the movement dynamics. In this particular framework, the action selection mechanism is tightly coupled to the agent's low-level sensory-motor systems and thus, fully embodied and situated. The neural dynamics architecture enables the system to flexibly organize the behaviors through controlled instabilities and stabilize decisions against perturbations. In addition, the direct coupling between the neural architecture and the timing dynamics ensures a synchronous and a coordinated performance for the overall dynamical model. As a result, the sequencing mechanism permits to integrate, simultaneously, various sources of time-varying perception about the task with an internal knowledge about the required logical order of the behaviors, their logical interplay, and the historical state of the system. These properties enable the robot to react flexibly and modify strategies in order to adapt to a dynamically changing environment.

The dynamical model is formulated inside the dynamical systems approach. The formulation relies on a uniform mathematical framework using nonlinear and autonomous dynamical systems. In the model, decisions making for behavioral organization, timed movements planning and control are generated from attractor solutions of nonlinear continuous time dynamics which leads to continuous behaviors. This also guarantees an intrinsic stability and robustness for the states of the system when a direct coupling to fluctuating and noisy sensory information is used to initiate, steer, and terminate the different actions. Moreover, flexibility properties of the dynamics permit to generate autonomously structured actions according to the current sensorial context and the system state. These properties emerge from bifurcations at different levels within the dynamical model: (a) in

the neural dynamics architecture to sequence actions and stabilize selected decisions, and (b) in the movement dynamics where bifurcations from fixed point attractors to stable oscillations permit to generate timed movements. Another distinguishable feature of the proposed approach is the analytical solvability of the dynamics. Indeed, the explicit specification of movement parameters like frequency and amplitude permits an online modulation of the trajectories via a direct linkage of the dynamics to time-varying sensory information. Furthermore, the low computation cost of the approach makes it well suited for real-time applications.

The problem of timing robot's actions to external events has been addressed using a plenty of methods ranging from planning algorithms (Rapp, 2011; Baetz et al., 2009), finite-state machine (Kober et al., 2012) or learning primitives by imitation (Kober et al., 2010; Kim et al., 2010). These approaches proved to be robust and adaptive control techniques in a variety of robotic tasks. However, they all showed drawbacks when it comes to react flexibly to variations in the task environment or to adapt movements to perturbations (see Section 2.1 for a review). By a tight coupling of planning, sensing and execution, the proposed dynamical model generates smooth, yet flexible sequences of timed behaviors in coordination with sensed events. I believe that these features, among others, may enable robots to operate autonomously in changing and uncertain environments and interact safely with human agents.

5.3 Limitations and future work

The experimental results of the proposed dynamical model proves the applicability of the approach in addressing the problem of timing and coordinating robot's behaviors with perceived events. In this sense, the dynamical model represents a first step toward generating safe and coordinated movements for robots that operate with humans in unstructured environments.

The model presents several interesting features, nevertheless, limitations due to various factors can be observed. In the following, I will summarize the main points along with propositions of possible improvements:

- To track and predict the ball motion, the model relies on external preprocessed perceptual information obtained from algorithmic processes. A necessary improvement will be to develop a neurally grounded perceptual system to acquire these data. Several evidences show that the brain may rely on internal physics model (Zago et al., 2009; Hayhoe et al., 2005) or experience-based processes (Kveraga et al., 2007) to predict upcoming events and plan movements in anticipation of those events. In our context, such a system should be able to produce internal prediction estimates of the ball trajectory. This addition will represent a base to achieve a fully neural architecture.

- In this work, the movement dynamics permit to generate adaptive discrete movements with stable temporal structures. Elaborating a more sophisticated mechanism to couple between effector's and ball's motions will certainly ensure a full coordination of robot's movements and increase the

performance of the dynamical model. In addition, using neural instead of Hopf oscillators to produce timed behaviors will make the approach more consistent. Also, learning these timed movements from human demonstrations may represent a possibility to be explored.

- Taking profit from the different properties offered by the DFT framework to implement cognitive functionalities in the architecture will be an important achievement. For instance, decision making mechanisms can be built based on sensory information, learned activity patterns, or behavioral history to sort out adequate behavior sequences or even select among different tasks according to the current context of the system.

- In the neural dynamics architecture, the synaptic weights that establish couplings between the different EBs and the associated intention and condition of satisfaction fields are set and tuned by hand. A major improvement will be to incorporate autonomous learning methods that start from early guesses and fine tune the parameters according to sensed rewards or from experiences during task execution. Such a step may considerably increase the robot's autonomy and optimize its performance. Learning methods like value-based reinforcement learning and Hebbian learning are directions of ongoing work in our laboratory (Kazerounian et al., 2013; Luciw et al., 2013).

- The performance of the model can be better assessed in a more adapted hardware platform. Transferring the implementation to another robotic manipulator that allows much faster movements, like the KUKA lightweight arm in our lab, will permit to evaluate the approach in absence of external constraints and even with more challenging applications.

Appendices

Appendix A

Methods and Parameters

During this work, a simulation software of the robotic hitting task was made in Matlab and is availabe on request. The C++ software interfaces for the hardware implementation were built using libraries availble at the Autonomous Robotics Lab in the Institute for Neural Computation. Several Open Source libraries for image processing were used as well.

In the following, I will provide more details about the visual system and the algorithms used in the implementation. After that, tables will list the parameters of exemplary elements that compose the dynamical model.

A.1 Visual system: supplement

The color-based detection process (see Section 4.1.2) permits to measure the ball pixel position $\boldsymbol{q} = [u\,,\, v]^T$ in image plane. To obtain the corresponding position $\boldsymbol{p}_\mathrm{b} = [x\,,\, y]^T$ in the inclined plane coordinate system Σ_P, a perspective back-projection is required.

First, I will explain briefly the camera perspective projection that is used to obtain the image plane 2D location from a 3D position estimate $\boldsymbol{p} = [x\,,\, y,\, z]^T$ of the ball on the inclined plane. As the ball rolls on the inclined plane and does not bounce on it, the z coordinate is set to the ball radius r_b. Then, I will describe the perspective back-projection which is the reverse operation, and show how can we extract $\boldsymbol{p}_\mathrm{b}$ from its corresponding image pixel point.

A.1.1 Perspective projection

According to the pinhole camera model Forsyth and Ponce (2003), a scene view is formed by projecting 3D points into the image plane via a perspective projection in homogeneous coordinates given by

$$s \begin{pmatrix} \boldsymbol{q} \\ 1 \end{pmatrix} = [\boldsymbol{K}\ \boldsymbol{0}_3]\, \boldsymbol{T} \begin{pmatrix} \boldsymbol{p} \\ 1 \end{pmatrix}, \tag{A.1}$$

or

$$s \begin{bmatrix} u \\ v \\ 1 \end{bmatrix} = \begin{bmatrix} f \cdot i_{\mathrm{x}} & 0 & o_{\mathrm{x}} & 0 \\ 0 & f \cdot i_{\mathrm{y}} & o_{\mathrm{y}} & 0 \\ 0 & 0 & 1 & 0 \end{bmatrix} \begin{bmatrix} \boldsymbol{R} & -\boldsymbol{R}\boldsymbol{c} \\ \mathbf{0}_3^T & 1 \end{bmatrix} \begin{bmatrix} x \\ y \\ z \\ 1 \end{bmatrix},$$

where s is a homogeneous scaling factor. The 4×4 homogeneous transform $\boldsymbol{T}$ is the matrix of extrinsic parameters that describes the position and orientation of the camera in an external reference frame (here, the inclined plane reference frame Σ_P). Mathematically, the position and orientation of the camera are defined by a 3×1 vector $\boldsymbol{c}$ and by a 3×3 rotation matrix $\boldsymbol{R}$ (respectively). $\boldsymbol{c} = [c_{\mathrm{x}}\,,\, c_{\mathrm{y}}\,,\, c_{\mathrm{y}}]^T$ is called the camera center or the center of projection and permits to define the camera coordinate system Σ_C. The all zero elements column vector is denoted by $\mathbf{0}_3$.

The homogeneous matrix $\boldsymbol{T}$ encodes two transforms

$$\boldsymbol{T} = \boldsymbol{T}^{W \to C}\, \boldsymbol{T}^{P \to W}, \tag{A.2}$$

where $\boldsymbol{T}^{P \to W}$ describes the position and orientation of the inclined plane with respect to a world coordinate system Σ_W centered at the robot manipulator base and is given by

$$\boldsymbol{T}^{P \to W} = \begin{bmatrix} \boldsymbol{R}_{\mathrm{x}}^{\alpha} & \boldsymbol{\ell} \\ \mathbf{0}_3^T & 1 \end{bmatrix}, \tag{A.3}$$

where $\boldsymbol{\ell} = [\ell_{\mathrm{x}}\,,\, \ell_{\mathrm{z}}\,,\, \ell_{\mathrm{z}}]^T$ is a position vector encoding the shifts along the world x-, y- and z-axes of the inclined plane and $\boldsymbol{R}_{\mathrm{x}}^{\alpha}$ is a rotation around the world x-axis by the plane inclination α and can be written as

$$\boldsymbol{R}_{\mathrm{x}}^{\alpha} = \begin{bmatrix} 1 & 0 & 0 \\ 0 & \cos\alpha & -\sin\alpha \\ 0 & \sin\alpha & \cos\alpha \end{bmatrix}, \tag{A.4}$$

$\boldsymbol{T}^{W \to C}$ expresses the transformation from the world to the camera coordinate system Σ_C and is derived by

$$\boldsymbol{T}^{W \to C} = \begin{bmatrix} (\boldsymbol{R}_{\mathrm{x}}^{\psi_{\mathrm{t}}} \times \boldsymbol{R}_{\mathrm{z}}^{\psi_{\mathrm{p}}}) & \boldsymbol{\jmath} \\ \mathbf{0}_3^T & 1 \end{bmatrix}, \tag{A.5}$$

where $\boldsymbol{\jmath} = [\jmath_{\mathrm{x}}\,,\, \jmath_{\mathrm{z}}\,,\, \jmath_{\mathrm{z}}]^T$ is the camera position vector with respect to Σ_W and the rotation matrices $\boldsymbol{R}_{\mathrm{x}}^{\psi_{\mathrm{t}}}$ and $\boldsymbol{R}_{\mathrm{z}}^{\psi_{\mathrm{p}}}$ define the Pan and Tilt camera orientations and are given by

$$\boldsymbol{R}_{\mathrm{x}}^{\psi_{\mathrm{t}}} = \begin{bmatrix} 1 & 0 & 0 \\ 0 & \cos\psi_{\mathrm{t}} & -\sin\psi_{\mathrm{t}} \\ 0 & \sin\psi_{\mathrm{t}} & \cos\psi_{\mathrm{t}} \end{bmatrix} \text{ and } \boldsymbol{R}_{\mathrm{z}}^{\psi_{\mathrm{p}}} = \begin{bmatrix} \cos\psi_{\mathrm{p}} & -\sin\psi_{\mathrm{p}} & 0 \\ \sin\psi_{\mathrm{p}} & \cos\psi_{\mathrm{p}} & 0 \\ 0 & 0 & 1 \end{bmatrix}. \tag{A.6}$$

The matrix of extrinsic parameters $\boldsymbol{T}$ describes the entire set of transformations between the coordinate systems Σ_P and Σ_C (see Figure A.1).

$\boldsymbol{K}$ is the camera matrix or the matrix of intrinsic parameters. It expresses the geometric, digital, and optical characteristics of the camera. These includes the focal length f which is the distance in mm from the center of projection $\boldsymbol{c}$ and

the principal point $\boldsymbol{o} = [o_{\mathrm{x}}\,,\, o_{\mathrm{y}}]^T$ that is usually at the image plane center and along the optical axis. The focal length f is usually multiplied by the effective size of the individual imager element $\boldsymbol{i} = [i_{\mathrm{x}}\,,\, i_{\mathrm{y}}]^T$ and expressed in pixels. The geometric distortion introduced by the optics and the lenses is also included in the the matrix as correction factors and are computed during the camera calibration process. $\boldsymbol{K}$ is used as a transformation matrix between the camera frame Σ_C and the pixel coordinate system Σ_I whose origin is set at the top left corner of the image plane (see Figure A.1).

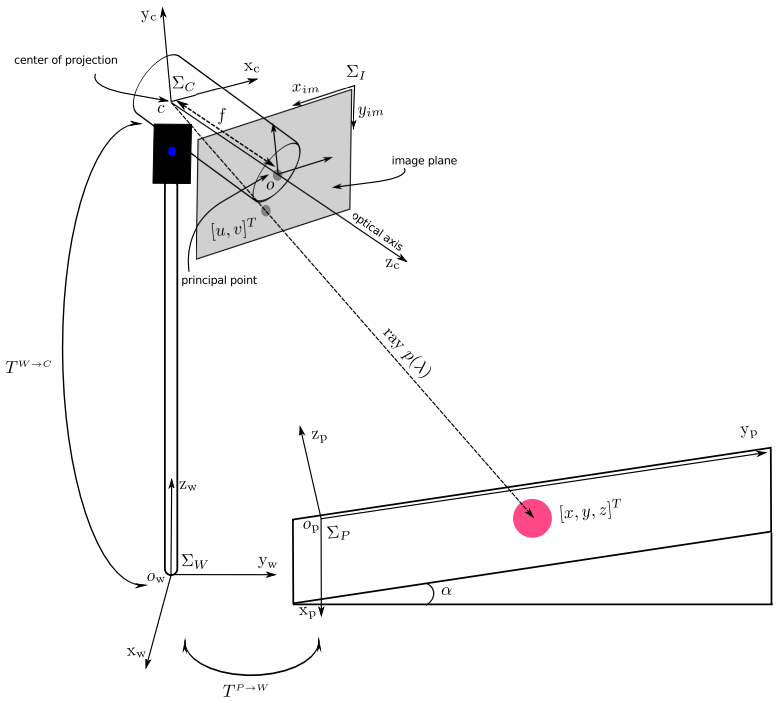


Figure A.1: The perspective projection of the robot's camera. The pinhole camera model describes the relationship between the ball position $[x\,,\, y\,,\, z]^T$ on the inclined plane and its corresponding 2D projection $[u\,,\, v]^T$ onto the image plane. The ray $\boldsymbol{p}(\lambda)$ represents the collection of 3D points that are mapped onto the same 2D projection $[u\,,\, v]^T$ (see the text about Perspective back-projection for more explanations.). The matrix of extrinsic parameters $\boldsymbol{T} = \boldsymbol{T}^{W\to C}\,\boldsymbol{T}^{P\to W}$ describes the geometric transformation between the inclined plane Σ_P and camera Σ_C coordinate systems. The matrix of intrinsic parameters $\boldsymbol{K}$ permits to transform positions expressed in the camera reference frame Σ_C to the pixel coordinate system Σ_I set at the top left corner of the image plane.

A.1.2 Perspective back-projection

The perspective back-projection is the reverse operation of Eq. A.1). Using the perspective back-projection, we can extract the ball position $\boldsymbol{p} = [x\,,\, y\,,\, z]^T$ in Σ_P from its image pixel coordinate $\boldsymbol{q} = [u\,,\, v]^T$.

Given a pixel position $\boldsymbol{q}$, there exists a collection of 3D points that are mapped and projected onto the same point $\boldsymbol{q}$. This collection of 3D points constitutes a ray connecting the camera center $\boldsymbol{c} = [c_{\mathrm{x}}\,,\, c_{\mathrm{y}}\,,\, c_{\mathrm{z}}]^T$ and $\boldsymbol{q} = [u\,,\, v]^T$ (see Figure A.1).

From Eq. A.1), the ray $\boldsymbol{p}(\lambda)$ associated to $\boldsymbol{q} = [u\,,\,v]^T$ can be defined as

$$\begin{bmatrix} x \\ y \\ z \end{bmatrix} = \underbrace{\boldsymbol{c} + \lambda\, \boldsymbol{R}^{-1}\, \boldsymbol{K}^{-1} \begin{bmatrix} \boldsymbol{q} \\ 1 \end{bmatrix}}_{\boldsymbol{p}(\lambda)}, \tag{A.7}$$

where λ is a positive scaling factor used to define the position of a 3D point on the ray. Since the z coordinate of the ball is known and equal to the radius r_{b}, it is possible to obtain the coordinates x and y by calculating λ using the relation

$$\lambda = \frac{z - c_{\mathrm{z}}}{z_3}, \tag{A.8}$$

where $(z_1\,,\,z_2,\,z_3)^T = \boldsymbol{R}^{-1}\, \boldsymbol{K}^{-1} \begin{bmatrix} \boldsymbol{q} \\ 1 \end{bmatrix}$.

A.2 Brief review of the Kalman filter

The Kalman filter addresses the general problem of estimating the state $\boldsymbol{x} \in \mathbb{R}^n$ of a discrete-time controlled process that is governed by the linear stochastic difference equation (Welch and Bishop, 2006) :

$$\boldsymbol{x}_{\mathrm{k}} = \boldsymbol{A}\boldsymbol{x}_{\mathrm{k\text{-}1}} + \boldsymbol{B}\boldsymbol{u}_{\mathrm{k\text{-}1}} + \boldsymbol{w}_{\mathrm{k}}, \tag{A.9}$$

with a measurement $\boldsymbol{z} \in \mathbb{R}^m$ given by

$$\boldsymbol{z}_{\mathrm{k}} = \boldsymbol{H}\boldsymbol{x}_{\mathrm{k}} + \boldsymbol{s}_{\mathrm{k}}. \tag{A.10}$$

The random variables $\boldsymbol{w}_{\mathrm{k}}$ and $\boldsymbol{s}_{\mathrm{k}}$ represent the process and measurement noise (respectively). They are assumed to be independent (of each other), white, and with normal probability distributions such that

$$p(w) \sim \boldsymbol{N}(0, \boldsymbol{Q}), \tag{A.11}$$

$$p(s) \sim \boldsymbol{N}(0, \boldsymbol{R}), \tag{A.12}$$

where $\boldsymbol{N}(\mu.\nu)$ is a Gaussian function with mean μ and covariance ν. It means that the noise is with a Gaussian amplitude distribution and statistically uncorrelated in time. To know more about such kind of noise and its characteristics see Maybeck (1979). $\boldsymbol{Q}$ and $\boldsymbol{R}$ are the process and measurement noise covariance matrices (respectively) and can be defined as

$$\boldsymbol{Q} = \boldsymbol{E}[\boldsymbol{w}_{\mathrm{k}}\boldsymbol{w}_{\mathrm{k}}^T], \tag{A.13}$$

$$\boldsymbol{R} = \boldsymbol{E}[\boldsymbol{s}_{\mathrm{k}}\boldsymbol{s}_{\mathrm{k}}^T], \tag{A.14}$$

where $\boldsymbol{E}[x]$ is the expectation function of a random variable x and defines the weighted average of a all the possible values of x.

The $n \times n$ matrix $\boldsymbol{A}$ in Eq. A.9 relates the state at the previous time step k-1 to the current state at k. The $n \times l$ matrix $\boldsymbol{B}$ relates the optional control

input $\boldsymbol{u} \in \mathbb{R}^l$ to the state $\boldsymbol{x}$. The $m \times n$ matrix $\boldsymbol{H}$ in Eq. A.10 relates $\boldsymbol{x}$ to the measurement $\boldsymbol{z}_\mathrm{k}$. In practice, all the previously defined matrices can be made to change with each time step or measurement, but here we assume them to be constant during the estimation process.

The Kalman filter estimates the process state $\boldsymbol{x}_\mathrm{k}$ by minimizing the estimate error $\boldsymbol{e}_\mathrm{k} = \boldsymbol{x}_\mathrm{k} - \hat{\boldsymbol{x}}_\mathrm{k}$ through minimizing the *a posteriori* estimate error covariance $\boldsymbol{P}_\mathrm{k} = \boldsymbol{E}[\boldsymbol{e}_\mathrm{k}\boldsymbol{e}_\mathrm{k}^T]$ where $\hat{\boldsymbol{x}}_\mathrm{k}$ is the current *a posteriori* estimate of the system state after a measurement update at time step k.

Note the recursive nature and the probabilistic origin of the filter that represent very appealing features of the Kalman estimator for various practical applications.

A.2.1 Operations of the Kalman filter

The operations of the Kalman filter can be thought as a form of feedback control. The filter estimates the process state (considered as a state prediction) at some time step and then uses the noisy measurements feedback to correct (or improve) this prediction. As such, the equations of the Kalman estimator fall into two groups :

1. The time update equations (also called predictor equations) are responsible for projecting forward (in time) the current state and error covariance matrix to obtain an *a priori* estimates for the next time step

$$\hat{\boldsymbol{x}}^-_\mathrm{k} = \boldsymbol{A}\hat{\boldsymbol{x}}_\mathrm{k\text{-}1} + \boldsymbol{B}u_\mathrm{k\text{-}1}, \tag{A.15}$$

   where $\hat{\boldsymbol{x}}^-_\mathrm{k}$ is an *a priori* state estimate computed from the discrete linear model (Eq. A.9) with an initial *a posteriori* estimate $\hat{\boldsymbol{x}}_\mathrm{k\text{-}1}$ that is updated later in Eq. A.18;

$$\boldsymbol{P}^-_\mathrm{k} = \boldsymbol{A}\boldsymbol{P}_\mathrm{k\text{-}1}\boldsymbol{A}^T + \boldsymbol{Q}, \tag{A.16}$$

   where $\boldsymbol{P}^-_\mathrm{k}$ is an *a priori* estimate of the error covariance matrix computed from the initial *a posteriori* error covariance estimate $\boldsymbol{P}_\mathrm{k}$ that is updated later in Eq. A.19; $\boldsymbol{Q}$ is the process error covariance matrix.

2. The measurement update equations (also called corrector equations) are responsible for the feedback, i.e., for incorporating a new measurement into the *a priori* estimate to obtain an improved *a posteriori* estimate

$$\boldsymbol{K}_\mathrm{k} = \boldsymbol{P}^-_\mathrm{k}\boldsymbol{H}^T(\boldsymbol{H}\boldsymbol{P}^-_\mathrm{k}\boldsymbol{H}^T + \boldsymbol{R})^{-1}, \tag{A.17}$$

   where $\boldsymbol{K}_\mathrm{k}$ is the Kalman gain that acts as a blending or weighting factor to minimize the *a posteriori* error covariance matrix in Eq. A.19; $\boldsymbol{R}$ is the measurement error covariance matrix;

$$\hat{\boldsymbol{x}}_\mathrm{k} = \hat{\boldsymbol{x}}^-_\mathrm{k} + \boldsymbol{K}_\mathrm{k}(\boldsymbol{z}_\mathrm{k} - \boldsymbol{H}\hat{\boldsymbol{x}}^-_\mathrm{k}), \tag{A.18}$$

where $\hat{\boldsymbol{x}}_{\mathrm{k}}$ is an *a postriori* state estimate computed from the *a priori* error covariance $\boldsymbol{P}_{\mathrm{k}}^{-}$; the difference $(\boldsymbol{z}_{\mathrm{k}} - \boldsymbol{H}\hat{\boldsymbol{x}}_{\mathrm{k}}^{-})$ is called the innovation and it is weighted by the Kalman gain $\boldsymbol{K}_{\mathrm{k}}$ that is used to express the trust either to the current measurement $\boldsymbol{z}_{\mathrm{k}}$ or the predicted measurement $\boldsymbol{H}\hat{\boldsymbol{x}}_{\mathrm{k}}^{-}$ depending on the measurement error covariance matrix $\boldsymbol{R}$ and the *a priori* error covariance estimate $\boldsymbol{P}_{\mathrm{k}}^{-}$.

$$\boldsymbol{P}_{\mathrm{k}} = (\boldsymbol{I} - \boldsymbol{K}_{\mathrm{k}}\boldsymbol{H})\boldsymbol{P}_{\mathrm{k}}^{-}, \tag{A.19}$$

where $\boldsymbol{P}_{\mathrm{k}}$ is the *a posteriori* estimate of the error covariance matrix and $\boldsymbol{I}$ is the identity.

After each time and measurement update pair, the process is repeated by projecting the previous *a posteriori* state estimate to predict a new *a priori* estimate. For a more detailed explanation of the Kalman filter process and its various applications see (Welch and Bishop, 2006; Mohinder, 2008; Maybeck, 1979).

A.2.2 Tunning of the Kalman filters

Defining the measurement error covariance matrix $\boldsymbol{R}$ is done by making off-line sample measurements and computing the variance from the process ground truth data. In the other hand, the process error covariance matrix $\boldsymbol{Q}$ is more difficult to obtain mainly because it is, generally, impossible to observe the process we are estimating. The values choice of these matrices is crucial for a correct functioning of the Kalman filter since they are used during the estimation process but never updated and so, don't converge to any specific numbers. In practical, even obtaining ground truth data about the measurable process quantities is hard. Thus, the $\boldsymbol{Q}$ and $\boldsymbol{R}$ matrices are tuned, most of the time, empirically.

The initial *a posteriori* estimate $\hat{\boldsymbol{x}}_0$ is set to the initial process state if known or to any approximate value. If the initial process state value is known, the initial *a posteriori* estimate of the error covariance matrix $\boldsymbol{P}_0$ can be set to zero

$$\boldsymbol{P}_0 = \begin{bmatrix} 0 & \cdots & 0 \\ \vdots & \ddots & \vdots \\ 0 & \cdots & 0 \end{bmatrix}, \tag{A.20}$$

but if there are uncertainties, $\boldsymbol{P}_0$ should be initialized with a suitably large number, say L, on its diagonal

$$\boldsymbol{P}_0 = \begin{bmatrix} L & \cdots & 0 \\ \vdots & \ddots & \vdots \\ 0 & \cdots & L \end{bmatrix}. \tag{A.21}$$

For the robotic hitting task, the parameters of the two Kalman filters (Section 4.1.2) are tuned empirically after a number of ball launch trials.

- For ball motion along x-axis, If I assume an acceleration in the form of a white Gaussian noise with a standard deviation $\sigma_{\mathrm{a,x}}$ that affects the

ball position and speed estimates, then we get the following process noise covariance matrix

$$\boldsymbol{Q}_{\mathrm{x}} = \boldsymbol{E}[\boldsymbol{w}_{\mathrm{k,x}}\boldsymbol{w}_{\mathrm{k,x}}^{T}] = \begin{bmatrix} \frac{\Delta t^4}{4} & \frac{\Delta t^3}{2} \\ \frac{\Delta t^3}{2} & \Delta t^2 \end{bmatrix} \sigma_{\mathrm{a,x}}^2. \tag{A.22}$$

A Gaussian measurement noise with a standard deviation $\sigma_{\mathrm{z,x}}$ gives the measurement noise covariance matrix

$$\boldsymbol{R}_{\mathrm{x}} = \boldsymbol{E}[s_{\mathrm{k,x}}s_{\mathrm{k,x}}^{T}] = \sigma_{\mathrm{z,x}}^2. \tag{A.23}$$

The initial *a posteriori* state estimate $\hat{\boldsymbol{x}}_{0,\mathrm{x}}$ is set to zero while the initial *a posteriori* estimate of the error covariance matrix $\boldsymbol{P}_{0,\mathrm{x}}$ is initialized by

$$\boldsymbol{P}_{0,\mathrm{x}} = \begin{bmatrix} L_{\mathrm{x}} & 0 \\ 0 & L_{\mathrm{x}} \end{bmatrix}, \tag{A.24}$$

where L_{x} is a suitably large number.

- For ball motion along y-axis, the tuning is similar. I assume a white Gaussian noise with a standard deviation $\sigma_{\mathrm{a,y}}$ affecting the commanded acceleration a_{y} and a process noise covariance matrix given by

$$\boldsymbol{Q}_{\mathrm{y}} = \boldsymbol{E}[\boldsymbol{w}_{\mathrm{k,y}}\boldsymbol{w}_{\mathrm{k,y}}^{T}] = \begin{bmatrix} \frac{\Delta t^4}{4} & \frac{\Delta t^3}{2} \\ \frac{\Delta t^3}{2} & \Delta t^2 \end{bmatrix} \sigma_{\mathrm{a,y}}^2. \tag{A.25}$$

With a measurement noise of standard deviation $\sigma_{\mathrm{z,y}}$, the measurement covariance matrix is set to

$$\boldsymbol{R}_{\mathrm{y}} = \boldsymbol{E}[s_{\mathrm{k,y}}s_{\mathrm{k,y}}^{T}] = \sigma_{\mathrm{z,y}}^2. \tag{A.26}$$

The initial *a posteriori* state estimate $\hat{\boldsymbol{x}}_{0,\mathrm{y}}$ is also set to zero and the initial *a posteriori* estimate of the error covariance matrix $\boldsymbol{P}_{0,\mathrm{y}}$ is initialized by

$$\boldsymbol{P}_{0,\mathrm{y}} = \begin{bmatrix} L_{\mathrm{y}} & 0 \\ 0 & L_{\mathrm{y}} \end{bmatrix}, \tag{A.27}$$

where L_{y} is large enough number.

The parameters of the two Kalman filters are tunned according to Table A.1.

Tracking axes	σ_{a} (m/s^2)	$\sigma_{\mathrm{z,y}}$ (m)	L (m^2)
x	0.15	0.01	20
y	0.85	0.01	20

Table A.1: Numerical values for the parameters of the two Kalman filters.

A.3 Parameters of the dynamics and their values

The parameters listed below are given with their values and are duplicated with their assciated elements. Some parameters are adapted according to the movement dimensions. When tuned, the same set of values is used both in simulations and the hardware implementation. Units are given where applicable.

A.3.1 Timing dynamics

parameter	value
fixation strength a	3.0
noise standard deviation $\sqrt{r}$	0.0
time scale τ	1.0 (s)
step time Δt	0.02 (s)

Table A.2: Numerical values for the parameters of the timing dynamics.

A.3.2 Movement module

parameter	value
sigmoid steepness β	5.0
sigmoid inflection point x_0	0.0
noise standard deviation $\sqrt{r}$	0.005
resting level h	-5.0
time scale τ	0.022 (s)
step time Δt	0.02 (s)

Table A.3: Numerical values for the parameters of the dynamical neural fields and nodes.

	'update'	'move'	'fix'
parameter	value	value	value
int. field kernel c_{exc}	5.0	5.0	3.0
int. field kernel c_{inh}	0.2	0.2	0.2
int. field kernel σ	5.0	5.0	5.0
CoS field kernel c_{exc}	5.0	5.0	-.-
CoS field kernel c_{inh}	0.2	0.2	-.-
CoS field kernel σ	5.0	5.0	-.-
int. node self excitation $c_{\text{exc,i}}$	4.0	4.0	4.0
CoS node self excitation $c_{\text{exc,CoS}}$	4.0	4.0	-.-
weight, int. node to CoS node $c_{\text{i,CoS}}$	0.1	0.1	-.-
weight, int. node to int. field $c_{\text{i,int}}$	4.0	4.0	5.0
weight, int. field to CoS field $c_{\text{int,CoS}}$	3.2	3.2	-.-
weight, CoS field to CoS node $c_{\text{CoS,CoS}}$	0.25	0.4	-.-
weight, CoS node to int. node $c_{\text{CoS,i}}$	-6.0	-6.0	-.-
weight, task node to int. node $c_{\text{t,i}}$	-.-	-.-	6.0

Table A.4: Numerical values for the parameters of a movement module EBs. Superscript relative to each EB is omitted in the parameter's names for clarity.

parameter	value
prec. node self excitation $c_{\text{exc,p}}$	4.0
suppr. node self excitation $c_{\text{exc,s}}$	4.0
weight, 'update' EB CoS node to prec. node $c_{\text{CoS,p}}$	-6.0
weight, prec. node to 'move' EB int. node $c_{\text{p,i}}$	-6.0
weight, 'move' EB int. node to suppr. node $c_{\text{i,s}}$	3.0
weight, suppr. node to 'fix' EB int. node $c_{\text{s,i}}$	-6.0

Table A.5: Numerical values for the parameters of behavioral constraints. Superscripts relative to each constraint and EB are omitted in the parameters names for clarity. The precondition node (prec.) is inhibited by the 'update' EB CoS node while the suppression node (suppr.) is boosted by a sub-threshold input coming from the 'move' EB intention node and needs an additional input from the higher level EB intention node to become active, see bellow.

A.3.3 Higher level elementary behavior

parameter	value
int. node self excitation $c_{\text{exc,i}}$	4.0
CoS node self excitation $c_{\text{exc,CoS}}$	4.0
weight, int. node to CoS node $c_{\text{i,CoS}}$	2.0
weight, CoS node to int. node $c_{\text{CoS,i}}$	-6.0
weight, offset node to int. node $c_{\text{off,i}}$	-6.0
weight, int. node to 'update' EB int. node $c_{\text{i,i}}$	6.0
weight, int. node to 'move' EB int node $c_{\text{i,i}}$	6.0
weight, int. node to prec. node $c_{\text{i,p}}$	6.0
weight, int. node to suppr. node $c_{\text{i,s}}$	3.0
weight, 'update' EB CoS node to CoS node $c_{\text{CoS,CoS}}$	2.0
weight, 'move' EB CoS node to CoS node $c_{\text{CoS,CoS}}$	2.0

Table A.6: Numerical values for the parameters of a higher level EB. Superscript relative to the EB is omitted to keep generality. The precondition node (prec.) is activated by the higher level EB CoS node and the suppression node (suppr.) is activated by an additional input provided by the higher level EB intention node.

Bibliography

Aboaf, E., Drucker, S., and Atkeson, C. (1989). Task-level robot learning : juggling a tennis ball more accurately. In *International Conference on Robotics and Automation (ICRA)*, pages 1290–1295. 4

Althaus, P. (2003). *Indoor Navigation for Mobile Robots: Control and Representations*. PhD thesis, Royal Institute of Technology. 11

Amari, S. (1977). Dynamics of pattern formation in lateral-inhibition type neural fields. *Biological Cybernetics*, 27:77–87. 13, 25

Atkeson, C. G., Moore, A. W., and Schaal, S. (1997). Locally weighted learning. *Artificial Intelligence Review*, 11:11–73. 6

Baetz, G., Lee, K., Wollherr, D., and Buss, M. (2009). Robot basketball: a comparison of ball dribbling with visual and force/torque feedback. In *International Conference on Robotics and Automation (ICRA)*, pages 1472–1477. 5, 103

Bicho, E., Mallet, P., and Schöner, G. (1998). Using attractor dynamics to control autonomous vehicle motion. In *Proceedings of the 24th Annual Conference of the IEEE Industrial Electronics Society*, volume 2, pages 1176–1181. 9

Bootsma, R. and Van Wieringen, P. (1990). Timing an attacking forehand drive in table tennis. *Journal of Experimental Psychology: Human Perception and Performance*, 16:21–29. 24, 26, 27, 30

Bouabdallah, S., Noth, A., and Siegwart, R. (2004). Pid vs lq control techniques applied to an indoor micro quadrotor. In *International Conference on Intelligent Robots and Systems (IROS)*, pages 2451–2456. 5

Bradski, G. and Kaehler, A. (2008). *Learning OpenCV*. O'Reilly Media, Sebastopol, USA. 63

Buehler, M., Koditschek, D., and Kindlmann, P. (1994). Planning and control of robotic juggling and catching tasks. *International Journal of Robotics Research*, 13:101–108. 4

Degallier, S. and Ijspeert, A. (2010). Modeling discrete and rhythmic movements through motor primitives: a review. *Biological Cybernetics*, 103:319–338. 26

Degallier, S., Righetti, L., Gay, S., and Ijspeert, A. (2011). Toward simple control for complex, autonomous robotic applications: Combining discrete and rhythmic motor primitives. *Autonomous Robots*, 31:155–181. 8

Dijkstra, T. M. H., Katsumata, H., de Rugy, A., and Sternad, D. (2004). The dialogue between data and model: Passive stability and relaxation behavior in a ball bouncing task. *Nonlinear Studies*, 11:319–345. 27

Domenech, A., Domenech, T., and Cebrian, J. (1987). Introduction to the study of rolling friction. *American Journal of Physics*, 55:231–235. 64, 66

Ellekilde, L. P. and Christensen, H. I. (2009). Control of mobile manipulator using the dynamical systems approach. In *International Conference on Robotics and Automation (ICRA)*, pages 1370–1376. 11, 13

Erlhagen, W. and Bicho, E. (2009). Dynamic field theory : Applications in cognitive science and robotics. Technical report, The European Network for Advancement of Artificial Cognitive Systems. 13, 16

Forsyth, D. A. and Ponce, J. (2003). *Computer Vision, A Modern Approach*. Prentice Hall. 106

Gibson, J. J. (1998). Visually controlled locomotion and visual orientation in animals. *Ecological Psychology*, 49:161–176. 26

Goodale, M. A. and Humphrey, G. K. (1998). The objects of action and perception. *Cognition*, 67:181–207. 24

Grewal, M. and Andrews, A. (1993). *Kalman filtering: theory and practice*. Prentice-Hall, Inc., Upper Saddle River, NJ, USA. 64

Gribovskaya, E. and Billard, A. (2009). Learning nonlinear multi-variate motion dynamics for real-time position and orientation control of robotic manipulators. In *International Conference on Robotics and Automation (ICRA)*, pages 472–477. 6, 7

Hailing, L., Haiyan, W., Lei, L., Kuehnlenz, K., and Ravn, O. (2012). Ping-pong robotics with high-speed vision system. In *International Conference on Control Automation Robotics and Vision (ICCARV)*, pages 106–111. 5, 60, 68

Haken, H., Kelso, J. A. S., and Bunz, H. (1985). A theoretical model of phase transitions in human hand movements. *Biological Cybernetics*, 51:347–356. 25

Hancock, P. A. and Manser, M. P. (1997). Time-to-contact more than tau alone. *Ecological Psycology*, 9:265–297. 26

Hayhoe, M., Mennie, N., Sullivan, B., and Gorgos, K. (2005). The role of internal models and prediction in catching balls. In *American Association for Artificial Intelligence (AAAI)*. 103

Hecht, H. and Savelsbergh, G. (2004). *Time-to-Contact*. Elsevier. 26

Hujic, D., Croft, E., Zak, Fenton, G. R., Mills, J., and Benhabib, B. (1998). The robotic interception of moving objects in industrial settings: Strategy development and experiment. *IEEE-ASME Transactions on Mechatronics*, 3:225–239. 68

Ijspeert, A. (2008). Central pattern generators for locomotion control in animals and robots: A review. *Neural Networks*, 21:642–653. 8, 25

Ijspeert, J. A., Nakanishi, J., and Schaal, S. (2002). Learning rhythmic movements by demonstration using nonlinear oscillators. In *International Conference on Intelligent Robots and Systems (IROS)*, pages 958–963. 6

Iossifidis, I. and Schöner, G. (2006). Reaching with an redundant robot arm using attractor dynamics. In *International Symposium on Robotics ISR and 4th German Conference on Robotics Robotik*. VDI Verlag. 11, 72

Jeannerod, M. (1984). The timing of natural prehension movements. *Journal of Motor Behavior*, 16:235–254. 24

Katsumata, H. and Russell, D. M. (2012). Prospective versus predictive control in timing of hitting a falling ball. *Experimental Brain Research*, 216:499–514. 26

Kazerounian, S., Luciw, M. D., Richter, M., and Sandamirskaya, Y. (2013). Autonomous reinforcement of behavioral sequences in neural dynamics. In *International Joint Conference on Neural Networks (IJCNN)*, pages 1–8. 104

Kelso, J. A. S. (1984). Phase transitions and critical behavior in human bimanual coordination. *American Journal of Physiology*, 246:1000–1004. 25

Kelso, J. A. S. (1995). *Dynamic patterns: The self-organization of brain and behavior*. MIT Press. 1

Khansari-Zadeh, S. M., Kronander, K., and Billard, A. (2012). Learning to play minigolf: A dynamical system-based approach. *Advanced Robotics*, 26:1967–1993. 6

Kim, S., Gribovskaya, E., and Billard, A. (2010). Learning motion dynamics to catch a moving object. In *International Conference on Humanoid Robots (ICHR)*, pages 106–111. 6, 68, 103

Kirk, D. E. (1970). *Optimal control theory: An introduction*. Englewood Cliffs, Prentice Hall. 5

Kober, J., Glisson, M., and Mistry, M. (2012). Playing catch and juggling with a humanoid robot. In *International Conference on Humanoid Robots (ICHR)*, pages 875–881. 5, 103

Kober, J., Muelling, K., Kroemer, O., Lampert, C. H., Schoelkopf, B., and Peters, J. (2010). Movement templates for learning of hitting and batting. In *International Conference on Robotics and Automation (ICRA)*, pages 853–858. 6, 103

Kohler, M. (1997). Using the kalman filter to track human interactive motion – modelling and initialization of the kalman filter for translational motion. Technical report, University of Dortmund. 65

Kveraga, K., Ghuman, A. S., and Bar, M. (2007). Top-down predictions in the cognitive brain. *Brain and Cognition*, 65:145–168. 103

Lampariello, R., Tuong, D. N., Castellini, C., Hirzinger, G., and Peters, J. (2011). Trajectory planning for optimal robot catching in real-time. In *International Conference on Robotics and Automation (ICRA)*, pages 3719–3726. 5

Large, E. (1997). Scaling the dynamical systems approach to path planning. In *IEEE International Symposium on Industrial Electronics*, pages 21–26. 12

Lee, D. and Reddish, P. (1981). Plummeting gannets: A paradigm of ecological optics. *Nature*, 293:291–294. 27

Lee, D. N., Young, D. S., Reddish, P. E., Lough, S., and Clayton, T. M. H. (1983). Visual timing in hitting an accelerating ball. *Quarterly Journal of Experimental Psychology*, 35:333–346. 27

Luciw, M., Kazerounian, S., Lakhmann, K. Richter, M., and Sandamirskaya, Y. (2013). Learning the perceptual conditions of satisfaction of elementary behaviors. In *Robotics: Science and Systems (RSS), Workshop: Active Learning in Robotics: Exploration, Curiosity, and Interaction*. 104

Luksch, T., Gienger, M., Mhlig, M., and Yoshiike, T. (2012). Adaptive movement sequences and predictive decisions based on hierarchical dynamical systems. In *International Conference on Intelligent Robots and Systems (IROS)*, pages 2082–2088. 8

Matsushima, M., Hashimoto, T., Takeuchi, M., and Miyazaki, F. (2005). A learning approach to robotic table tennis. *IEEE Transactions on Robotics*, 21:767–771. 6

Maybeck, P. S. (1979). *Stochastics Models, Estimation, and Control*. Academic Press, Inc, New York, USA. 109, 111

Mohinder, S. G., A. P. A. (2008). *Kalman Filtering: Theory and Practice Using MATLAB*. Wiley-IEEE Press, New York, USA. 111

Monteiro, S. and Bicho, E. (2002). A dynamical systems approach to behavior-based formation control. In *International Conference on Robotics and Automation (ICRA)*, pages 2606–2611. 9

Morasso, P. (1981). Spatial control of arm movements. *Experimental Brain Research*, 42:223–227. 24

Mueller, M., Lupashin, S., and Raffaello, D. (2011). Quadrocopter ball juggling. In *International Conference on Intelligent Robots and Systems (IROS)*, pages 5113–5120. 5

Muelling, K., Kober, J., and Peters, J. (2010). A biomimetic approach to robot table tennis. In *International Conference on Intelligent Robots and Systems (IROS)*, pages 1921–1926. 5, 68

Nakashima, A., Sugiyama, Y., and Hayakawa, Y. (2006). Paddle juggling of one ball by robot manipulator with visual servo. In *International Conference on Robotics and Automation (ICRA)*, pages 1–6. 4, 68

Oubbati, F., Richter, M., and Schöner, G. (2013a). Autonomous robot hitting task using dynamical system approach. In *IEEE International Conference on Systems, Man, and Cybernetics (SMC)*, pages 4042–4047. 3

Oubbati, F., Richter, M., and Schöner, G. (2013b). A hierarchical, neural-dynamic architecture generating sequences of timed movements. In *Robotics: Science and Systems (RSS), workshop: Hierarchical and Structured Learning for Robotics*. 3

Oubbati, F. and Schöner, G. (2013). Autonomous timed movement based on attractor dynamics in a ball hitting task. In *International Conference on Agents and Artificial Intelligence (ICAART)*, pages 304–311. 3, 13

Pieper, D. L. (1968). *The Kinematics of Manipulators Under Computer Control*. PhD thesis, Stanford University. 72

Rapp, H. H. (2011). 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)*, pages 430–435. 5, 60, 103

Reimann, H., Iossifidis, I., and Schöner, G. (2011). Autonomous movement generation for manipulators with multiple simultaneous constraints using the attractor dynamics approach. In *International Conference on Robotics and Automation (ICRA)*, pages 5470–5477. 11, 72

Richter, M., Sandamirskaya, Y., and Schöner, G. (2012). A robotic architecture for action selection and behavioral organization inspired by human cognition. In *International Conference on Intelligent Robots and Systems (IROS)*, pages 2457–2464. 8, 17, 18, 19, 22, 23

Rosenbaum, D. (2009). *Human motor control*. San Diego, CA: Academic Press. 23

Sandamirskaya, Y. Richter, M. and Schöner, G. (2011). A neural-dynamic architecture for behavioral organization of an embodied agent. In *IEEE International Conference on Development and Learning (ICDL)*, pages 1–7. 18

Sandamirskaya, Y. and Schöner, G. (2008). Dynamic field theory and embodied communication. *Lecture Notes in Computer Science*, 4930:260–278. 16

Santos, C. and Ferreira, M. (2009). Timed trajectory generation using dynamical systems: Application to a puma arm. *Robotics and Autonomous Systems*, 57:182–193. 7

Schaal, S. and Atkeson, C. G. (1994). Robot juggling: An implementation of memory-based learning. *Control Systems Magazine*, 14:57–71. 6

Schaal, S., Ijspeert, A., and Billard, A. (2003). Computational approaches to motor learning by imitation. *Philosophical Transactions of the Royal Society*, 358:537–547. 6

Schaal, S., Sternad, D., and Atkeson, C. G. (1996). one-handed juggling: a dynamical approach to a rhythmic movement task. *journal of motor behavior*, 2:165–183. 25, 27

Schöner, G. (1990). A dynamic theory of coordination of discrete movement. *Biological Cybernetics*, 63:257–270. 25, 27, 35

Schöner, G. (1991). Dynamic theory of action-perception patterns: the moving room paradigm. *Biological Cybernetics*, 64:455–462. 27

Schöner, G. (1994). Dynamic theory of action-perception patterns: The time-before-contact paradigm. *Experimental Brain Research*, 13:415–439. 27

Schöner, G. (2002). Timing, clocks and dynamical systems. *Brain and Cognition*, 48:31–51. 24, 25, 30

Schöner, G. (2008). *Dynamical Systems Approaches to Cognition*. Cambridge University Press. 2, 3, 9, 13, 14, 16

Schöner, G. and Dose, M. (1992). A dynamical systems approach to task-level system integration used to plan and control autonomous vehicle motion. *Robotics and Autonomous Systems*, 10:253–267. 12

Schöner, G., Dose, M., and Engels, C. (1995). Dynamics of behavior: Theory and applications for autonomous robot architectures. *Robotics and Autonomous Systems*, 16:213–245. 9

Schöner, G. and Kelso, J. A. S. (1988). Dynamic pattern generation in behavioral and neural systems. *Science*, 239:1513–1520. 1, 9, 25

Schöner, G. and Santos, C. (2001). Control of movement time and sequential action through attractor dynamics: A simulation study demonstrating object interception and coordination. In *9th Intelligent Symposium On Intelligent Robotic Systems (SIRS)*. 7, 26, 28, 34

Schöner, G. Dijkstra, T. M. H. and Jeka, J. J. (1998). Action-perception patterns emerge from coupling and adaptation. *Ecological Psychology*, 10:323–346. 26

Schutter, J. D., Geeter, J. D., Lefebvre, T., and H., B. (1999). Kalman filters: A tutorial. *Journal A*, 40:538–546. 64

Searle, J. R. (1983). *Intentionality: An essay in the philosophy of mind*. Cambridge University Press. 18

Senoo, T., Namiki, A., and Ishikawa, M. (2006). Ball control in high-speed batting motion using hybrid trajectory generator. In *International Conference on Robotics and Automation (ICRA)*, pages 1762–1767. 5, 60, 68

Shukla, A. and Billard, A. (2011). Coupled dynamical system based hand-arm grasp planning under real-time perturbations. In *Robotics: Science and Systems (RSS)*. 7

Siegler, I., Mantel, B., Warren, W. H., and Bardy, B. (2003). Behavioral dynamics of a rhythmic ball-bouncing task. In *Progress in Motor Control IV meeting*, pages 178–184, Caen, France. 27

Soares, R., Bicho, E., Machado, T., and W., E. (2007). Object transportation by multiple mobile robots controlled by attractor dynamics: theory and implementation. In *International Conference on Intelligent Robots and Systems (IROS)*, pages 937–944. 9

Steinhage, A. and Schöner, G. (1997). The dynamic approach to autonomous robot navigation. In *IEEE International Symposium on Industrial Electronics*, pages 63–73. 13

Steinhage, A. and Schöner, G. (1998). Dynamical systems for the behavioral organization of autonomous robot navigation. In *Sensor Fusion and Decentralized Control in Robotic Systems*, pages 63–73. 12, 13, 18

Sternad, D., Duarte, M., Katsumata, H., and Schaal, S. (2001). Bouncing a ball: Tuning into dynamic stability. *Journal of Experimental Psychology: Human Perception and Performance*, 27:1163–1184. 27

Strogatz, S. H. (1994). *Nonlinear Dynamics and Chaos*. Perseus Books, New York, USA. 12, 24, 29

Thelen, E. and Smith, L. B. (1994). *A dynamic systems approach to the development of cognition and action*. MIT Press. 2, 9

Tresilian, J. R. (1991). Empirical and theoretical issues in the perception of time to contact. *Psychology: Human Perception & Performance*, 17:865–876. 26

Tuma, M., Iossifidis, I., and Schöner, G. (2009). Temporal stabilization of discrete movement in variable environments: an attractor dynamics approach. In *International Conference on Robotics and Automation (ICRA)*, pages 863–868. 7

Van Gelder, T. (1998). The dynamical hypothesis in cognitive science. *Behavioral and Brain Sciences*, 21:615–628. 9

Van Gelder, T., Hobson, J. A., Kihlstrom, J., and Schacter, D. (1999). *Dynamic Approaches to Cognition*. MIT Encyclopedia of Cognitive Sciences. 2, 9

Wagner, H. (1982). Flow-field variables trigger landing in flies. *Nature*, 297:147–148. 27

Warren, W. H. (2006). The dynamics of perception and action. *Psychological Review*, 113:358–389. 24, 26, 27

Welch, G. and Bishop, G. (2006). An introduction to the kalman filter. Technical report, Chapel Hill, NC, USA. 109, 111

Wilson, H. and Cowan, J. D. (1973). A mathematical theory of the functional dynamics of cortical and thalamic nervous tissue. *Kybernetik*, 13:55–80. 13

Yilmaz, A., Javed, O., and Shah, M. (2006). Object tracking: A survey. *ACM Journal of Computing Surveys*, 38:1–45. 63

Yoshikawa, T. (1985). Dynamic manipulability of robot manipulators. In *International Conference on Robotics and Automation (ICRA)*, pages 1033–1038. 5

Zago, M., McIntyre, J., Senot, P., and Lacquaniti, F. (2009). Visuo-motor coordination and internal models for object interception. *Experimental Brain Research*, 192:571–604. 103

Zibner, S. K. U., Faubel, C., Iossifidis, I., and Schöner, G. (2011). Dynamic neural fields as building blocks of a cortex-inspired architecture for robotic scene representation. *IEEE Transactions on Autononous Mental Development*, 3:74–91. 17, 39

Declaration

I herewith declare that I have produced this thesis without the prohibited assistance of third parties and without making use of aids other than those specified; notions taken over directly or indirectly from other sources have been identified as such. This thesis has not previously been presented in identical or similar form to any other German or foreign examination board. The thesis work was conducted from 10.2009 to 03.2014 under the supervision of Prof. Dr. Gregor Schöner at Institut für Neuroinformatik, Ruhr-Universität Bochum.

Bochum, April 21^{th}, 2014

Curriculum Vitae

Personal

- Name: Farid Oubbati.
- Date of birth: July 08^{th}, 1978.
- Place of birth: Laghouat, Algeria.

Education

- Eng.D Electronics, National Institute of Electricity and Electronics (Boumerdes, Algeria), 1996 - 2001.
- M.Sc Automatics, Polytechnic National School (Algiers, Algeria), 2002 - 2005.
- Ph.D student Robotics, Institute for Neural Computation, Ruhr-Universität (Bochum, Germany), 2010 - now.

Employment

- Assistant researcher at the Institute for Neural Computation, Ruhr-Universität (Bochum, Germany), 2009–now.
- Lecturer at the University of Laghouat (Laghouat, Algeria), 2007–2009.
- Maintenance Engineer at SPA Milok, Laghouat (Laghouat, Algeria), 2006–2007.
- Assistant researcher at the National Institute of Electricity and Electronics, University of Boumerdes (Boumerdes, Algeria), 2001–2002.

Publications

Conference Papers

- Farid Oubbati and Omar Stihi, "Robot controllers Design Using the Hardware In the Loop 'HIL' and Rapid Control Prototyping 'RCP' DSP card techniques", in First International Meeting on Electronics, Electrical Science and Engineering (IMESE 2006), University of Djelfa, Algeria.
- Farid Oubbati and Gregor Schöner, "Autonomous timed movement based on attractor dynamics in a ball hitting task", in Proceedings of the 5th International Conference on Agents and Artificial Intelligence (ICAART 2013), p 304-311, Barcelona, Spain.
- Farid Oubbati, Mathis Richter and Gregor Schöner, "Generating & Hierarchically Organizing Sequences of Timed Movements", in Robotics: Science and Systems (RSS 2013), workshop: Hierarchical and Structured Learning for Robotics, Berlin, Germany.
- Farid Oubbati, Mathis Richter and Gregor Schöner, "Autonomous Robot Hitting Task Using Dynamical System Approach", in IEEE International Conference on Systems, Man, and Cybernetics (SMC 2013), p 4042-4047, Manchester, UK.