EU – FP7



# AMARSi

# Adaptive Modular Architectures for Rich Motor Skills

ICT-248311

D 6.3

March 2013 (36 months)

## Technical report on the cognitive architecture

Authors: Herbert Jaeger (Jacobs University), Mostafa Ajallooeian (EPFL- A), Aude Billard (EPFL-B), Jochen Steil (Bielefeld University), Felix Reinhart (Bielefeld University), Francis wyffels (University of Gent), Martin Giese (University of Tübingen)

| | |
|---|---|
| Due date of deliverable | 30.03.2013 |
| Actual submission date | 15.04.2013 |
| Lead Partner | Jacobs University Bremen |
| Revision | final |
| Dissemination level | Public |

# 1 Introduction

"Cognition" is a term which, in the "traditional" top-down perspective of cognitive science and AI, is associated with symbolic knowledge representations, logical/rational reasoning, and goal-oriented planning and decision making. However, in Alife and behavior-based robotics, there has always been the alternative, bottom-up view on cognition as something that emerges from the bodily interaction of an agent with its physical environment. The philosophies behind the bottom-up and top-down routes to understanding intelligence have not yet been ultimately reconciled. The bottom-up engineering of behavior-based autonomous robots has not scaled up to the combinatorial complexity of representations and planning schemes of today's AI systems, nor has, conversely, symbolic AI found ways of how to amalgamate large symbolic knowledge bases into continuous, high-bandwidth streams of sensor information and motor control commands of situated robots.

In this general scientific context, the AMARSi project takes a particular and somewhat unconventional stand. The project aims at "cognitive architectures", but these are framed in terms of the real-time control of a highly redundant, compliant robot – and not framed in the traditional terms of abstract symbolic reasoning. Work in the AMARSi project has led us through increasingly differentiated views on what "cognitive architectures" spell out in this demanding robotics scenario. This development is reflected in the preceding deliverables D.6.1 and D.6.2. Today, after three years, we know much better than at project start time what are the **crucial design dimensions** that need to be addressed when one aims at realizing rich repertoires of motor skills:

1. **Hierarchies** of control: "higher"-level controllers control/**modulate** "lower"-level controllers.
2. **Modularity**: complex skills – because they are just that, complex – can only be realized by integrating a variety of functionalities, which leads to modular architecture composed of a variety of functional modules.
3. **Sequencing and mixing**: complex motor behavior must be structured in time, sequentially and in parallel.

These are the main design coordinates (and obstacles) for achieving what one might call cognitive motor control.

This report is structured as follows. First we provide an overview of technical research in the robotics partner groups which has a bearing on the three themes listed above (Section 2). Then we will provide an in-depth discussion of the lessons learnt from AMARSi research on cognitive architectures (Section 3).

# 2 Overview of relevant technical research

## 2.1 EPFL-A (Ijspeert)

### 2.1.1 Complex skill: walking over unperceived rough terrain

A hierarchical and modular control architecture has been developed for the skill of quadruped walking over unperceived rough terrein. Two particular challenges were addressed. First, the obstacles / surface roughness was random and not to be perceived by the robot, such that a "perceive – model – plan" coping strategy was impossible. Instead, a fast, low-level re-balancing and obstacle-avoiding had to be invoked. Second, the forward speed of the robot was to be fast enough to enforce dynamic walking (up to 2 body lengthes / sec). The architecture for this demanding skill is hierarchical and modular and couples together four mechanisms:

1. A network of coupled CPGs (one per end-effector) generates a periodic, 12-dimensional gait motor pattern.
2. If a leg hits an obstacle (sensed by contact sensor), a reflex mechanism is triggered which (strongly and instantaneously) modulates the shape of the target trajectories emitted by the CPG system.
3. A standard proportional feedback controller is used to for making the 12 joint angles track the output of the reflex-modulated CPG system.
4. In addition to this, a higher-level, model-based control loop monitors the overall pose of the robot and generates re-balancing commands to stabilize this pose when perturbed. These re-balancing commands are additively superimposed on the motor commands issued by the P-controller that is simultaneously active on the lower level.

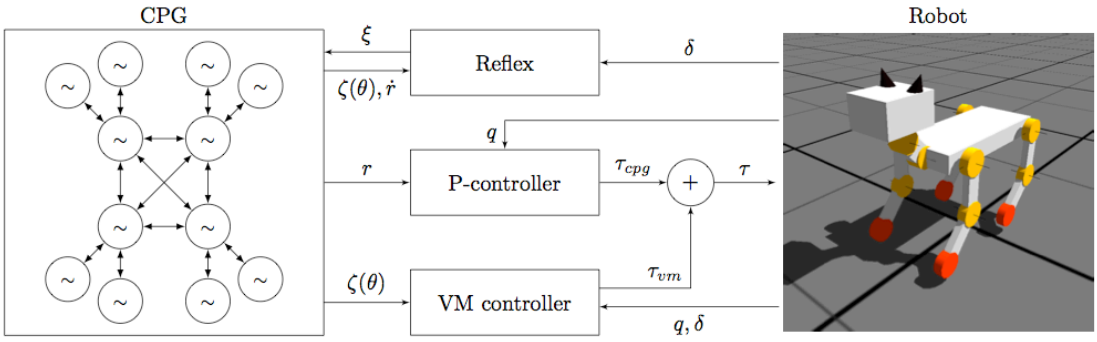



Figure 1. The proposed architecture for locomotion control. A CPG, implemented as coupled oscillators, generates the rhythmic joint angle patterns needed for an open-loop locomotion. The output of the CPG, $r$, is converted into motor torques, $\tau_{cpg}$, through a P-controller. The reflex mechanism receives the contact sensing information, ™, from the robot and phase information from the CPG and generates reflex feedbacks, ⌊, if needed. The CPG torques are augmented with virtual model control torques, $\tau_{vm}$, which are generated based on contact and proprioceptive information received from the robot. The activity of the VM controller can be inhibited by the CPG by /(⌊), a phase-dependent term. Taken from [1].

Figure 1 provides a sketch of this architecture. It has been successfully tested on simulated stiff [1] and compliant [2] Oncilla robots.

This architecture is an instructive demonstration of combining the three crucial design dimensions listed in the Introduction. "Walking over rough terrain" is revealed as a complex skill which needs (at least) a basic gait pattern generation, a fast reactive obstacle coping reflex, and a global posture stabilization mechanism. These mechanisms interact in a way which is dependent on the phase of the walking cycle (a sub-theme of the sequencing design dimension).

## 2.2 EPFL-B (Billard)

### 2.2.1 Complex skill: catching objects in flight

In year 3, a main AMARSi theme at EPFL-B was to continue developing an architecture for the skill of catching an object in flight [3][4][5]. Robotic catching of flying objects has been addressed several times in work outside AMARSi (short review in [5]). However, all of these previous works either assumed the existence of analytical models of the object or focused on ballistic motion (catching balls only). These assumptions are unrealistic as robots are bound to have to manipulate rapidly a variety of objects, whose analytical models cannot be known. For instance, if a robot manipulates bottles filled with liquid, the inertia of the object changes as an effect of the amount and type of liquid it is filled with. Besides, most objects cannot be grabbed at the center of mass and, hence, tracking the grasping point amounts to predicting non-linear translational and rotational motion.

This work considers that the catching point on the object is not located at the center of mass and that the object's dynamics is highly non-linear (e.g. a tennis racket has an asymmetric shape which increases the effect air friction) a half-filled water bottle changes weight repartition during flight) quickly rotating in flight. This requires an online estimation of the object's pose, an adaptive online computation of the catching location and object pose at catching time, and an online adaptation of the hand/finger poses to ready them for the predicted catching act. All of this has to be done in the face of noisy sensor measurements and with very high demands on speed (object and hand pose estimation/prediction cycle 10 ms, final motor control loop cycle 2 ms).

Current realizations of this skill at EPFL-B are rooted in probabilistic modeling for modeling the feasibility region for grasping and in the dynamical systems based approach to robot control, advocated by AMARSi. Specifically, the coordination between the arm reaching motion toward the predicted catching location and the hand/finger pose preparation is done by coupling two dynamical systems that have been trained individually, using the *coupled dynamical systems* (CDS) method developed at EPFL-B. Since CDS has been documented in detail previous deliverables (e.g. D.6.2), we only give a condensed summary here:

Generally, one may consider the task to coordinate two motion patterns A and B, where A and B are using different degrees of freedom of the robot (here: the arm reaching A uses arm joints, the finger preshaping B uses finger joints). For a replication in robots, the CDS method developed at EPFL-B first trains A and B individually from a small number of demonstrations of the combined A + B behavior, using the Lab's *stable estimator of dynamical systems* (SEDS) methodology. Then, use the available training data to estimate an essentially 1-dimensional coupling function $\Psi$, based on a Gaussian mixture model obtained from the same training data.

Ψ maps the current state of the DoF's of A into a scalar *phase* variable for B. Due to its low dimensionality, estimating this coupling function does not blow up the required training data size. The coupling of A with B is directed: while A (here: the reaching motion of the arm) unfolds autonomously, the motor pattern B evolves under the additional influence of the phase variable Ψ. This *coupled dynamical system* (CDS) model can be mathematically set up in a way which (i) ensures that the termination times for both A and B coincide, (ii) preserves the assured stability conditions that the native A and B controllers enjoy due to their SEDS training, and (iii) preserves the original recovery-from-perturbations characteristics of A and B.

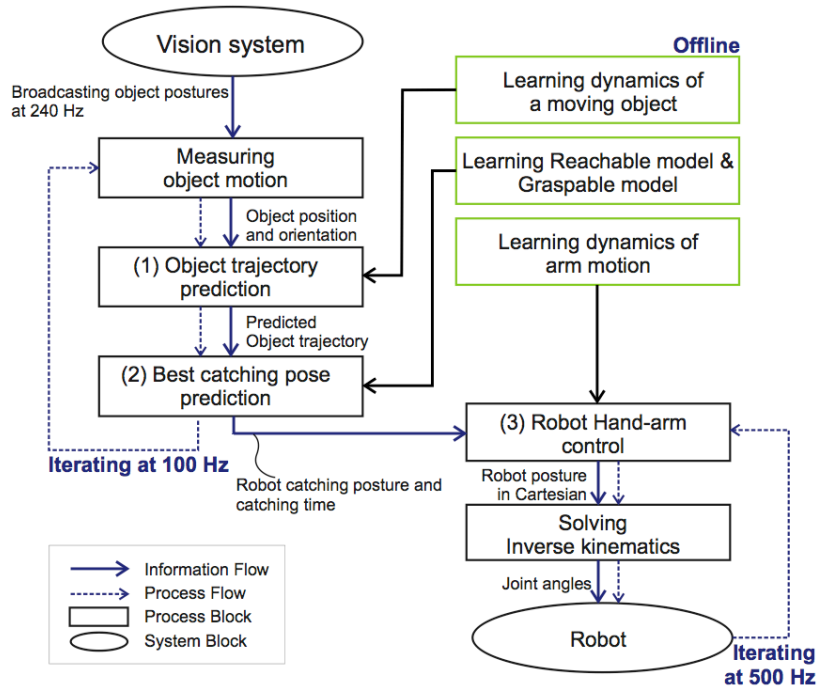



Figure 2. Architecture for skill of catching flying objects. Taken from [5].

Figure 2 shows the overall system architecture. It has been demonstrated on the iCub robot (in simulation; because the real robot cannot produce the required speed of motion) and on a 7 DoF Kuka arm. Current research aims at extending this skill to the COMAN robot. This is a very significant extension, because it includes whole-body orientation motions and stepping motions. Such complex whole-body catching motion patterns are currently being recorded from humans, in order to acquire training data for robot control.

This is a unique example of the strength of dynamical-system based to provide fast and robust adaptation of arm-hand motion under high uncertainty. It is also another example of how realizing a single skill necessitates a modular and hierarchical architecture. The latter aspect (hierarchy) here comes into the play in two instances. First, there is the division of labor between modules which compute a target trajectory, and the "low-level" custom tracking controllers of the robot arm. Second, here we have a directed phase coupling between the arm reach and the hand pose controllers, where the latter is influenced by the former but not vice versa.

### 2.2.2 Stabilizing a dynamical systems based controller

In a more general and theoretical line of work, M. Khansari-Zadeh at EPFL-B has extended the previously developed *Stable Estimator for Dynamical Systems* (SEDS) method for learning a dynamical system (vector field) representation for a directed motion from a small number of human demonstrations. The core idea of the basic

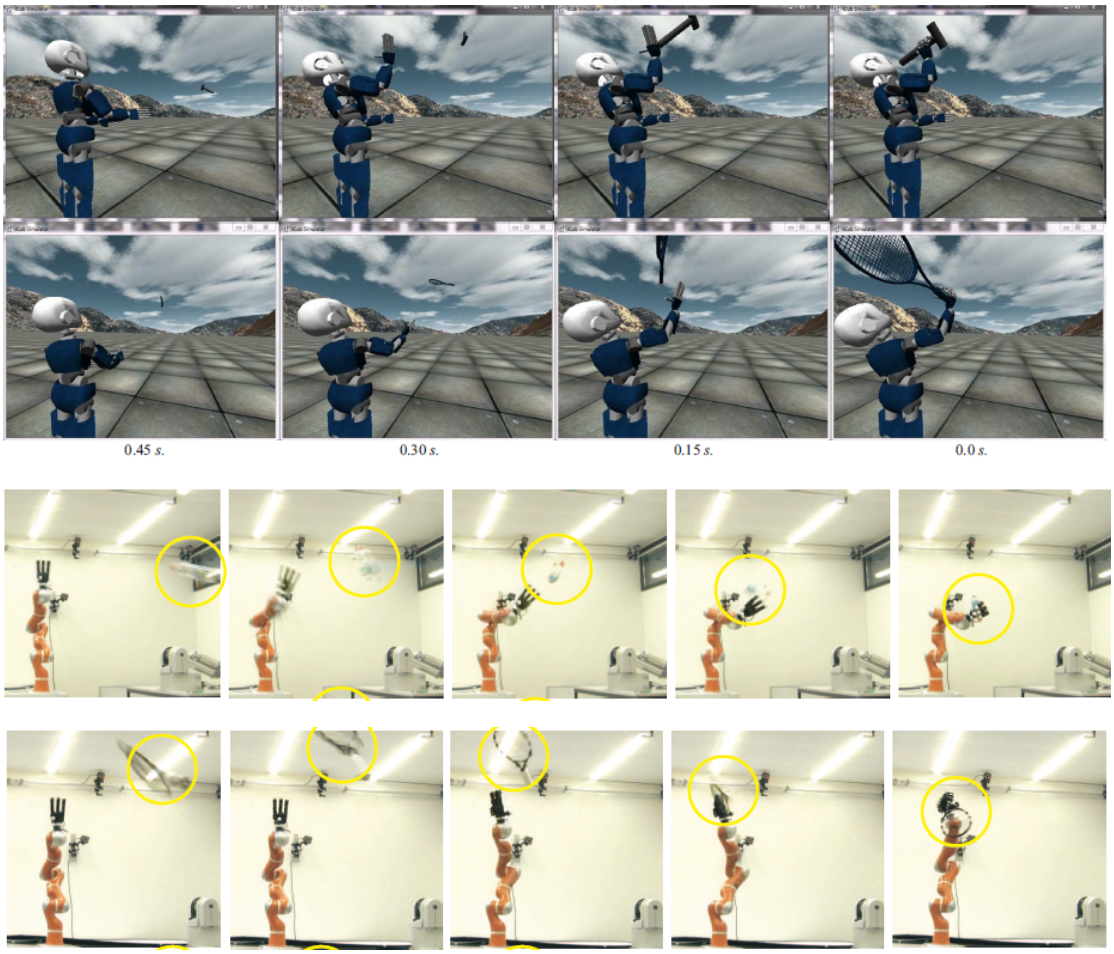



**Figure 1 (two first rows) iCub robot catching a bammer and a tennis racket in simulation; (two bottom rows) KUKA robot catching a bottle half-filled with water and tennis racket**

SEDS is to capture the state/velocity information contained in the demonstrations first in a probabilistic Gaussian Mixture Model (GMM), from which then a differentiably smooth vector field with assured convergence and stability properties is extracted (e.g., [6]). A potential shortcoming of this classical SEDS is that stability is ensured only during training, but not at the time of exploiting the motion model for actual robot control. The extended version of SEDS, called SEDS-II [7], establishes a higher layer of control whose outputs are superimposed on the SEDS trajectory signals. The purpose of this higher layer of control is to guarantee stability of the overall control loop at runtime. It is based on constructing a global Lyapunov function whose minimum coincides with the target end pose of the modeled motion, and which otherwise approximates the temporal evolution of the human-demonstrated trajectories with a decreasing energy gradient.

The SEDS-II methodology has been tested on a physical 7 DoF robot arm in an object placement task. An orange was to be picked from a variety of initial locations, to be

placed on a plate or in a bucket. The requisite trajectories were learnt from a small number of human demonstrations.

With respect to our cognitive motor control theme, SEDS-II is another example of one "higher" control layer modulating a "lower" layer.

## 2.3 UGent

### 2.3.1 Controlling a pattern generator

In Deliverable 6.2 (month 30) a method was reported for modulating the shape and, more difficult, the frequency of a reservoir-based periodic pattern generator. This line of work has been extended in two ways. First, a systematic study of conditions when frequency adaptation becomes disrupted by bifurcations has been carried out [8]. A main finding is that a careful training of the reservoir using the FORCE learning algorithm [9] in the equilibration training scheme [10] (Figure 3) can extend the range of frequency modulation to about a factor of 3. Second, the simple proportional controllers that were used for shape and frequency control have been replaced by online adaptive reservoir-based controllers [11]. These can autonomously optimize the gain matrix that is needed to control a pattern-generating reservoir. Figure 4 shows the architecture layout and one of the Oncilla leg control demonstrations that were run.

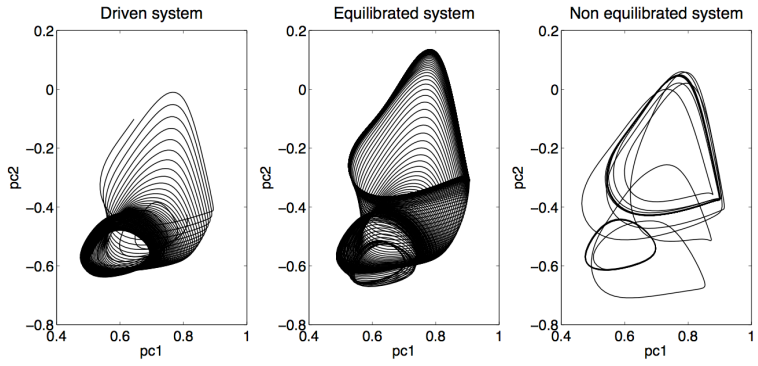



Figure 3. Comparison of the state evolution (2-dim projections of network states) of a driven (left), actively generating equilibrated (middle) and an actively generating, non-equilibrated (right) oscillator network. Without careful equilibration, the network learns two or more periodic attractors of different frequencies (right), which renders a smooth frequency control impossible. Taken from [8].

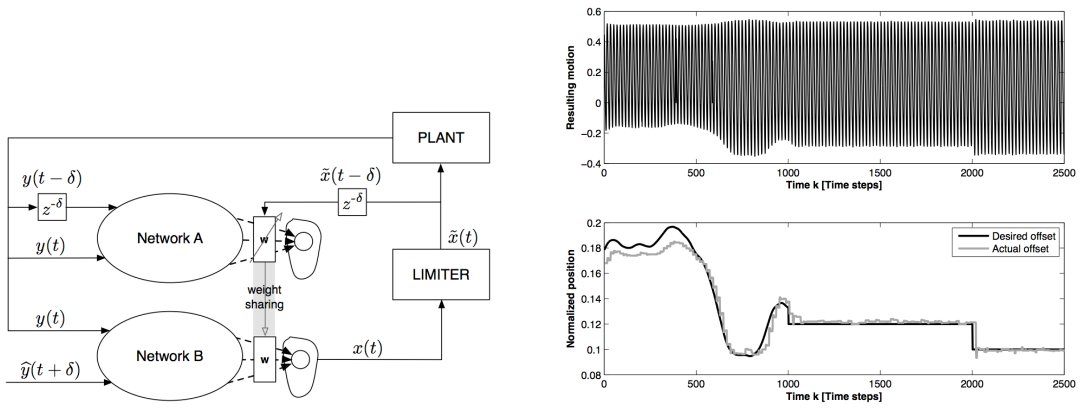


Figure 4. Left: Architecture layout for reservoir-based online adaptive controller for tracking a nonlinear plant (here: a pattern-generating reservoir network). Right: Demonstration: controlling the offset of an Oncilla leg walking pattern. Taken from [11].

This line of work is again an instance of the trend that we witness as AMARSi progresses: when one wants to realize increasingly complex skills, one is naturally led to hierarchical control structures where a basic pattern generator (which in turn drives lowest-level "firmware" tracking controllers) is modulated from a "higher" level.

### 2.3.2 Modular architecture for control with primitives (MACOP)

UniGent has developed a control architecture which was inspired by the well-known MOSAIC control architecture [13], but, in a sense, turns it around. In MOSAIC type of architectures, different controllers are trained for different tasks. At exploitation time, the controller is selected whose associated forward model currently best predicts the sensor information. In MACOP [12], different controllers are trained which specialize not on different tasks, but on different regions in joint space. The partitioning of joint space into subregions is effected by an adaptive online algorithm which optimizes effective joint space coverage and variance of the partition. Dependent on the current joint space partitioning, the controllers then optimize their tracking accuracy with learning rates that mirror the partitioning. The resulting trained system can be used for novel tracking tasks whose trajectories were not contained in the training trajectories, provided the joint space regions visited at execution time are not too different from the joint space regions explored during training. In its current versions, the controllers used are of the same adaptive online, reservoir-based type as the ones used in the pattern modulation studies reported above (Figure 4 left). Figure 5 gives an architecture diagram.

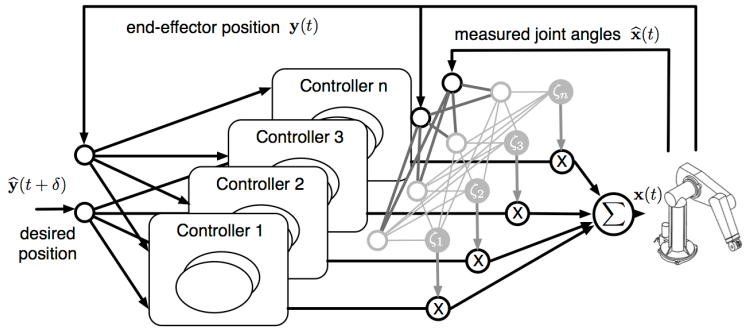



Figure 5. Illustration of MACOP architecture. The desired (objective) and the current end-effector position are used as external inputs to each controller. The controller outputs are weighted by a scaling factor and superimposed with each other such that the resulting motor commands (e.g. joint angles) are given to the robot. The used scaling factors represent the responsibility of a controller and is determined by the actual joint angles, given by the encoders, and the actual end-effector position. Taken from [12].

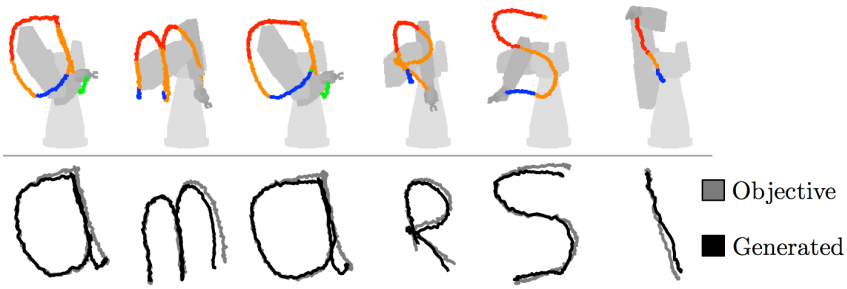



Figure 6. Testing the MACOP control architecture. After 50 training repetitions, a mixture of 5 controllers could reasonably well track the (noisy, hand-written) "amarsi" target patterns (lower row). The five controllers are represented in the top row by colors red, orange, blue, gray, green; color indicates which of the controllers had the maximal "responsibility" weight in a given section of the target path. Taken from [12].

The MACOP architecture was tested, among others, on a Webots simulated robot arm in a task of writing "amarsi" in the air (Figure 6).

The MACOP approach is still young and further tests and refinements need to be performed. However, it offers a quite original way to approach the modularity / mixing challenge for cognitive motor control, which could be called the "responsibility leads" principle for controller training and usage: the primary quantity to determine is the degree of responsibility that is assigned to a controller; only secondly, the controller adapts (in training) or executes (in exploitation) with a share determined by the responsibility value that has been determined for it.

## 2.4 University of Tübingen

### 2.4.1 Complex skill: emotion-expressive walking

A recently started line of work of this partner concerns humanoid walking control which is modulatable so as to reflect emotional states (such as sad, happy) of the walker. This work derives from a long tradition in this Lab to characterise subtle aspects of human motion in terms of dynamical systems, and specifically, to characterise emotional states [14] (collaboration with Weizmann).

The control follows a hierarchical scheme again. A stable walking gait is realized through an implementation of A. Feldman's equilibrium-point controllers for the lower body. This control level is modulated from a higher level through kinematic primitives that have been extracted from human actors' recordings. The integration of the high-level kinematic control is achieved by adding PD force terms superimposed on the forces provided by the equilibrium-point controllers. This coupling results in morphing of the basic low-level generated trajectory towards the desired kinematic trajectory. This two-level dynamic control architecture allows the modifications of the gait style in a predictive way regarding the perturbations in locomotion and navigation.

At the time of writing, this system has been demonstrated in simulation of a planar CoMan model. Current ongoing work concerns the extension to a 3D CoMan model. Figure 7 shows snapshots.

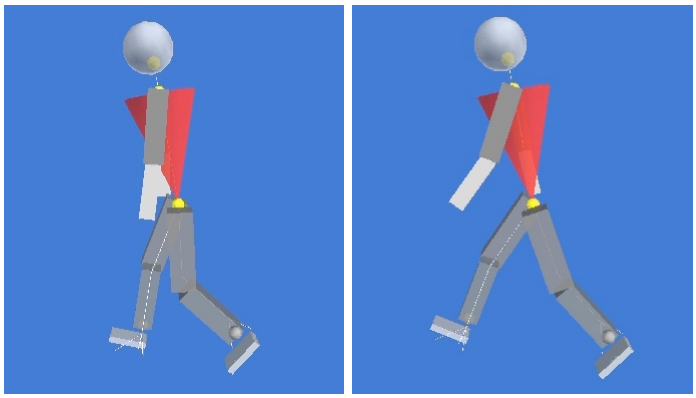


Figure 7. Demonstration of two walking patterns controlled by the low-level equilibrium-point controllers of the simulated planar CoMan. The basic equilibrium-point control (with time-based switching) is combined with two different kinematic trajectories: ‘sad walking’ (left) and ‘happy walking’ (right).

## 2.5 UniBi (Cor Lab)

### 2.5.1 Sequencing within and between skills

In a study [15] which is very instructive for cognitive motor control, UniBi has designed a control architecture for the iCub for three bi-manual skills: "paddling left", "paddling right", and "weight lifting". Since this study has been reported in Deliverable 6.2 in some detail, here we are very brief. Figure 8 (a) shows a sequence of three snapshots where the robot first paddled left (left picture), then made a transitional move (center picture) to paddling right (right picture). The transitional move has to negotiate a travel of the hands to new working areas. The travel trajectory is hand-coded to have bell-shaped velocity profile in agreement with findings from human motions. In figure 8 (b), by contrast, a single "weight lifting" skill is shown in execution, which consists of two alternating discrete movement primitives that are coupled by a sequencer module. The sequencer is implemented as a state machine, where convergence of a discrete primitive, i.e. $\|v\| \approx 0$, triggers the transition to the next state.

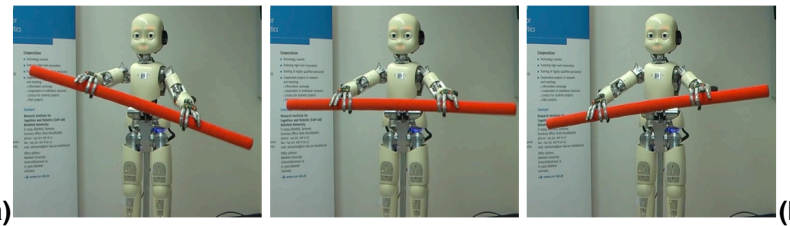
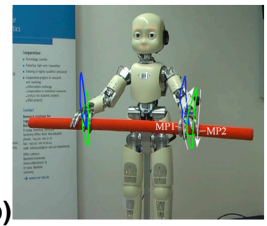
**(a)**  **(b)** 


Figure 8. **(a)** A transition motion (center) between two skills (left and right). **(b)** Sequencing two discrete movement primitives ("up" and "down") within a single skill. See text for explanation. Taken from [15].

This study is instructive w.r.t. cognitive motor control because it (again) raises the question of how to define skills. Here we described the behavior organization as it is done by the authors in [15]. However, it would also be defendable to consider the "up" and "down" parts of the weight lifting as individual skills, or the ability to change between right and left paddling as parts of one complex skill. Furthermore, one may or may not grant skill status to the transition motion between the left and right paddling.

### 2.5.2 Coupling controllers through shared effector variables

Motor tasks are often defined in a way that primarily concerns only a part of the body. For example, humanoid walking, at first sight, mainly concerns the legs and torso while the arms and hands appear less directly involved. Such a primary association of two motor patterns with two separate portions of the body often makes it feasible to perform two motor tasks simultaneously, e.g. pointing while walking. However, the segregation will rarely be perfect. In a natural human walk, the arms become entrained to the walking and contribute to balancing, and when a standing human points s/he will also bend the torso and move the legs, albeit only slighty. Thus, when one wishes to add new motor patterns to a robot control system, one should have an understanding of how its execution interferes with the execution of other motor patterns, even if at first sight there is a segregation of affected body parts.

Addressing this kind of problem, UniBi has studied the arm-torso-arm interaction of motor tasks for a (simulated) iCub robot, where the motor pattern controllers were primarily defined and trained for each arm individually [16]. Since this work was already detailed in Deliverable 6.2, we give only a very summary account. Two subsystem controllers were individually trained in the beginning. The first subsystem comprised the four left arm joints and three torso joints, the second subsystems the right arm and the same three torso joints again. Each of these two (symmetric) modules can be used to compute joint trajectories for the 3 torso plus the 4 arm joints, given a desired task space trajectory. But how can these two control modules be combined? Figure 9 shows how it was done. Both the left arm and the right arm module are executed in parallel. They will typically generate different joint targets for the 3 torso joints that they share. These two commands are simply averaged before being sent to the robot.

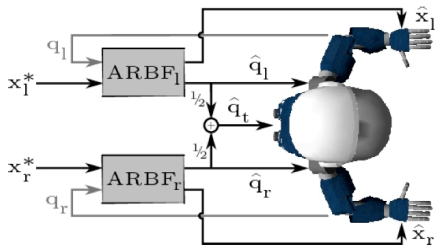
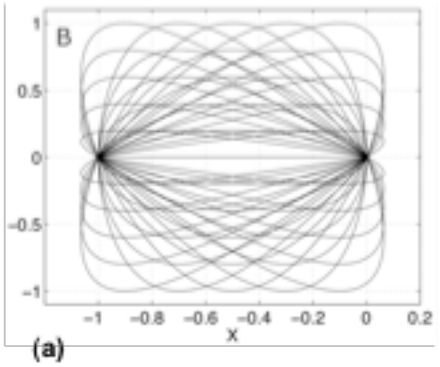



Figure 9: combining the right and left arm/torso controllers. For explanation see text. (Taken from [16])

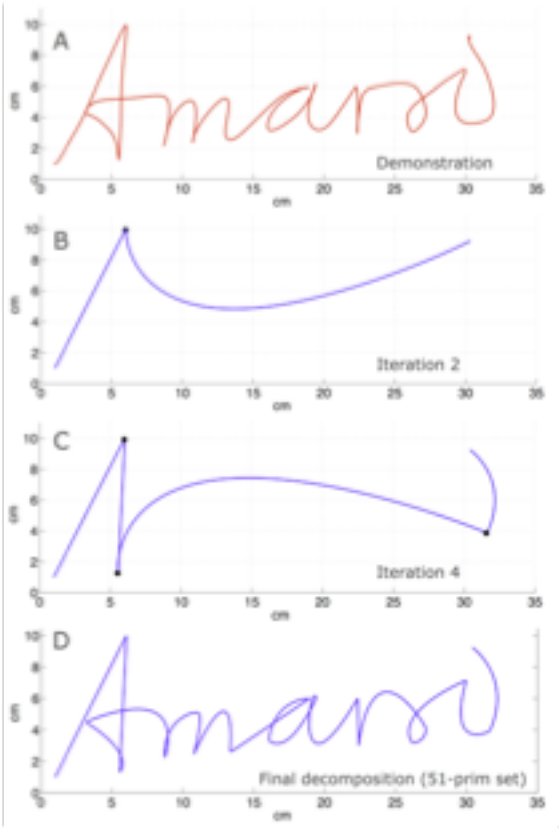
This negotiation scheme leads to natural-looking interactions between the two arms, which could be demonstrated in a number of pointing/body turning tasks. While in these tasks the prescribed reaching targets predominantly concerned the arms and hands, subtle interactions between the right and left arm through the torso led to small modulations of the torso joints which were important for a natural appearance.

### 2.5.3 Hierarchical task decomposition from library of primitives

UniBi developed a method for hierarchically segmenting a task trajectory into segments which can be tracked by activating stored control routines for a library of movement primitives [17]. This is best explained by an example. Figure 10 (a) shows a set of 51 2-dimensional motor primitives. These trajectories have been synthesized with a minimal-jerk model. The top panel in Figure 10 (b) shows a complex planar target trajectory. In the first analysis step (not shown), this trajectory would be approximated by a single element from the library (the best fitting one would be selected). In subsequent steps, the point of maximal error in the current approximation is detected, used as a split point, and the target portions left and right of the split point are approximated by best-matching elements from the library. In panels B and C the first two steps in this hierarchical decomposition of the target are shown, and panel D gives the final result, which consists of 18 parts.




(a)




(b)

Figure 10. Segmenting a complex target trajectory (top panel in **(b)**) using a predefined set of motion primitives **(a)**. Taken from [17].

## 2.6 Jacobs University

### 2.6.1 Fast transient pattern modulation

When a walking or running legged robot perceives an obstacle or a ground hole on short notice, the ongoing leg motion needs to be modulated in a fashion that is quick, strong, adapted and transient. At Jacobs, a scenario has been considered where the obstacle/hole is perceived and classified on higher cognitive levels, and a short-duration signal is sent to lower control levels, of the kind "make stepsize longer", "lift foot higher", etc. These signals are qualitative commands, but do not carry detailed target trajectory information. According to the assumed scenario, they arrive at the lower control level, where a periodic gait pattern is being generated and controlled, at unpredictable phase angles w.r.t. the periodic base pattern. A method has been developed by which the lower-level periodic pattern generator is realized by a reservoir network, which is trained to react to the "alert modulation" signals by (much) slowing down, changing the offset, or changing the amplitude transiently and robustly, independent of the oscillation phase when the alert signal arrives [18]. Figure 11 shows a demonstration of transiently slowing-down a periodic pattern (the most difficult type of modulation to train; amplitude and offset being much easier to influence).

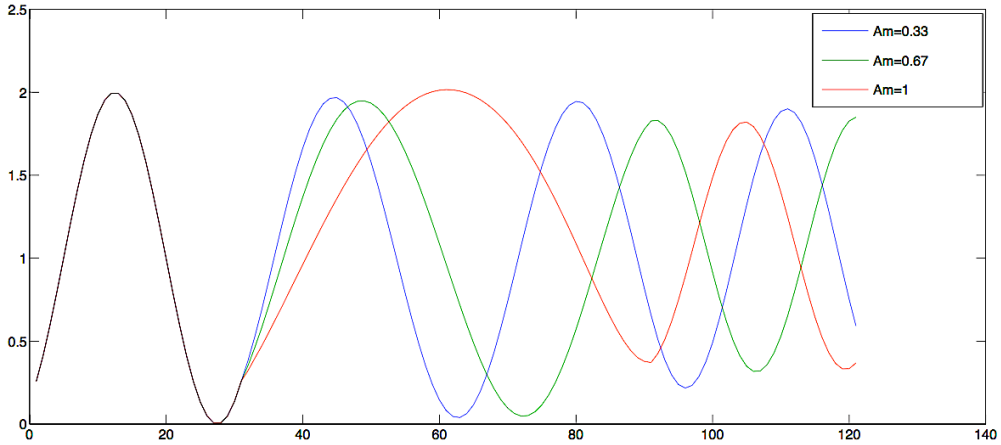



Figure 11: Slowing down a sinewave. The sine pattern is generated by a reservoir network that has been trained to slow down when a rectangular signal arrives at a specific modulation port. The degree of slowing down can be regulated by the amplitude *Am* of the alert input. The modulation starts at timestep 30 and ends at timestep 90. Taken from [18].

### 2.6.2 Morphing between patterns

A recurring basic problem in robotics is to create transitory trajectories which connect a pattern $A$ with a subsequent pattern $B$. Examples are gait transitions in walking robots, or transits to new areas in workspace in manipulation tasks. At Jacobs a generic method to obtain such transitory trajectories has been developed. It is

assumed that pattern $A$ and $B$ are generated by the same recurrent neural network (RNN). When a RNN generates a (periodic or non-periodic) pattern $A$, its network state vectors evolve in a linear subspace $L_A$ which is characteristic for that pattern. This subspace corresponds to a projection matrix which we can likewise denote by $L_A$. When the RNN is generating pattern $A$, its states all lie in $L_A$. Therefore, one may use either of these state update equations: (i) $\mathbf{x}(n+1) = \tanh(\mathbf{x}(n))$ or (ii) $\mathbf{x}(n+1) = L_A \tanh(\mathbf{x}(n))$, obtaining identical dynamics. When one wants to morph between pattern $A$ and $B$ within $k$ timesteps, one can use a linear morph between $L_A$ and $L_B$ to control the network dynamics via a version of (ii) of the kind $\mathbf{x}(n+1) = (\mu L_B + (1-\mu)L_A)\tanh(\mathbf{x}(n))$, where the morph parameter $\mu$ grows linearly from 0 to 1 during the desired morphing period. Figure 12 shows two examples obtained in this way. A publication is in preparation.

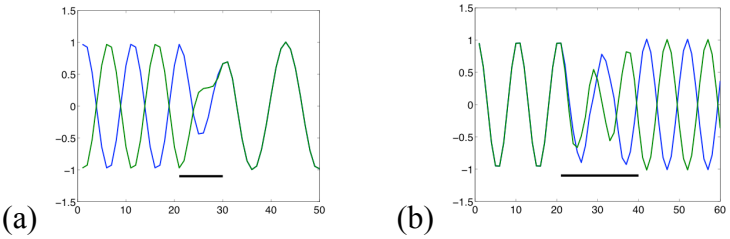



Figure 12: Morphing between two 2-dim oscillator patterns. Two examples are shown where a transition (duration marked by black bar) from an antiphase to an in-phase (panel (a)) or vice versa (panel (b)) pattern with a simultaneous change in frequency was shaped.

### 2.6.3 A Complete Bayesian Agent

Most work carried out in AMARSi concentrates on realizing individual, highly differentiated skills, or combinations of a small number of them. An alternative route, typical for the behavior-based tradition of AI, is to start from a holistic model of an autonomous agent and let it differentiate. This route has been explored at Jacobs, guided by the well-known proposals made by Friston et al. for a "surprise-minimizing" (Bayesian) agent [19]. Friston has proposed that the paradigm of surprise minimization could explain how animals move: actions aims at changing the world so that it corresponds to what is expected by the animal. This idea is compelling because it claims perception and action derive from the same principle.

The Jacobs group has implemented such an agent model to check the validity of Friston's claim: can one derive complex behavior from the surprise minimization principle? Unlike Friston's Bayesian approach, the implementation relies on the well-proven efficiency of reservoir computing architecture to define such agent (Figure ). Indeed, reservoirs are good at reproducing stimuli. We have shown that Echo State Networks (an instance of reservoir computing) can be mathematically reformulated as tuning perception to minimize surprise. As Friston suggested, we have built action generation mechanisms on top of this perceptive architecture under the same principle of surprise minimization. This leads to an extension of reservoir computing as a basis architecture for surprise minimizing agents for model-free robots/environments.

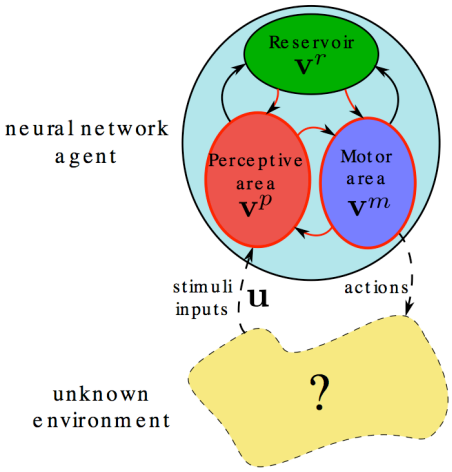



Figure 13: A surprise-minimizing agent architecture. The agent is realized by recurrent neural networks in the reservoir computing tradition.

When such an agent, entirely untrained, is exposed to an unknown environment, it adapts both its perceptions as well as its actions to maximize predictability of sensor feedback. As suggested in the literature, the first results show that surprise minimization alone does not lead to interesting behavior. Indeed, surprise minimizing agents tend to stabilize the agen-environment interaction in an equilibrium point. Although they seem to define a class of universal stabilizers, they do not exhibit rich behavior. In other words, these agents crawl to a dark cave. Thus, there is a need of a goal.

In a second step, it was shown how to make these agents follow a target trajectory. They still minimize surprise but along a desired behavior. This has been successfully tested on synthetic robots / environments. Both batch and online learning modes have been developed. In both cases, the critical computation is the gradient of the environment with respect to the actions. In a batch framework it can be measured by the experimenter and in an online framework it is empirically done by the agent. Ongoing work (collaboration with UniGent) concerns working on controlling the Oncilla robot into walking by asking the robot to follow as fast as possible a kinematic trajectory for the feet of the robot. A publication is in preparation.

## 2.7 Analysis

The development of skill control mechanisms in AMARSi continues to mature. A number of **new complex skills** have been realized recently:

- walking over unperceived rough terrain (2.1.1)
- catching objects in flight (2.2.1)
- air-drawing of script letters ("amarsi", 2.3.2)
- emotion-expressive walking (2.4.1)
- paddling right/left and weight lifting with transitions (2.5.1)

This list is neither comprehensive nor final; work continues to refine and extend these and other skills.

With respect to the cognitive aspects of control, the topics **hierarchy**, **temporal organization**, and **modularity** are of particular interest.

In all of the complex skills mentioned above (and all other skills implemented in the project), there is a **triple hierarchy of control**. On the lowest level (level 0), hardware-adapted feedback tracking controllers transform joint target trajectories into physical motion. In the case of the iCub and commercial robot arm platforms, these controllers are encapsulated pieces of software that come shipped with the robot. In the case of the Oncilla (simulated or physical), typcially simple P or PID controllers are used. On the next level 1, a dynamical system (neural network or analytical) based pattern generator generates a basic form of a target trajectory for the level-0 controllers. In case of quadruped locomotion and walking this is often referr to as CPG, in the other scenarios (e.g. catching, grasping, mainpulation) point attractor based systems are used. These target trajectories become modulated by superordinate mechanisms which we may call level 2 control. A multitude of such modulatory influences has been found useful and has been exploited for robot control (simulated and/or physical):

- A virtual model of the body is used to generate whole-body **stabilizing** forces which are superimposed on the level-1 trajectories (2.1.1)
- A Lypunov function based vector field yields online **stabilizing** additive modulations to level-1 trajectories in a generic method to guarantee global stability (2.2.2)
- A reservoir-based, online adaptive "meta" controller **controls slow observables** (frequency, offset, amplitude) of a likewise reservoir-based CPG network (2.3.1)
- The target trajectories generated by CPGs are **gated** by a superordinate responsibility assignment module (2.3.2)
- A level-1 bipedal walking controller, which by itself generates a stereotypcial basic walking gait, is **modulated** by force terms from higher-level kinematic primitives to achieve emotion-expressive walking variants (2.4.1)
- Level-1 CPG networks are **sequenced** by state-based higher level coordination modules (2.5.2)

This level-0/1/2 picture given here is an extreme simplification, drawn to illuminate the fact that, while hierarchization in general is widely employed, the particular objectives and mechanisms show a great variation. In addition to the hierarchical top-down flows of control just listed, there are "lateral" or bottom-up interactions between control layers. Examples are the phase angle input to a hand / finger pose level-1 CPG which originates from another level-1 controller for **timing** coordination (2.2.1), or bottom-up, touch sensor triggered obstacle avoidance reflexes (2.1.1).

Finally, we have seen several different aspects of the **temporal organization** of complex skills (individual and across skills):

- a **phase-dependent impact of obstacle avoidance reflexes** on a basic walking CPG (2.1.1),
- also, a **phase-independent impact of obstacle avoidance reflexes** on a basic oscillatory pattern (2.6.1),

- the use of a phase variable for the **temporal alignment of arm motion with hand/finger motion** (2.2.1),
- a **slow-timescale** modulation of a **fast-timescale** CPG network (2.3.1),
- a learnt adaptive **temporal mixing of controllers** by continuously varying responsibility variables (2.3.2),
- **state-based switching mechanisms** for alternating discrete movements (2.5.1),
- explicitly designed **transition moves** to connect the executions of different skills (2.5.1),
- **coupling controllers through shared variables** (2.5.2),
- **hierarchical sequencing of tasks** by matching task segments with stored motion primitives (2.5.3),
- a **neural morphing mechanism** between different oscillatory patterns (2.6.2).

As to **modularity**, it is of interest to state explicitly the obvious fact that the complex skills implemented by AMARSi partners are all realized by multi-modular control architectures. An important observation is that at first sight these architectures widely differ from one another, in complexity, nature of modules, and functional logics. We will argue in the Discussion section that this is unavoidable to a certain degree. The picture that emerges from all of this evolving work is characterized, most conspicuously, by the wide variety of functional and behavior organization schemes that have been found to lead to working solutions.

## 2.8 The DSL-view on architectures

Before we discuss the implications of the findings from the last section, we show that from the perspective of the structured representation of these architectures in the DSL-language, we have to qualify the apparent variety in the sense that the functional variety is after all not that large, in particular with respect to modularity. It is one of the two goals of the conceptual framework of the AMARSi-DSL to capture essential features of modularity and to identify common functional units in its vocabulary (the other goal is of course the simplification of development of experiments and software generation, which is discussed in D7.5). The following short analysis shows how useful the common language is. To this extend, the partners have collaboratively restated some of their architectures discussed above in the DSL-language. Figs. 14-19 show already discussed architectures for quadruped locomotion by pattern generation from EPFL-A (Fig.14, Sec. 2.1.1), inverse kinematics learning by UGent (Fig.15, Sec. 2.3.2), catching objects by EPFL-B (Fig.16, Sec. 2.2.1), distributed upper body control through shared effector variables realized by dynamical networks for iCub (UniBi, Fig.17, Sec. 2.5.2), and manipulation of a stick by UniBi in the preliminary first draft from the year 2 review (Fig.18) and the current version (Fig.19) for comparison as discussed in Sec. 2.5.1.

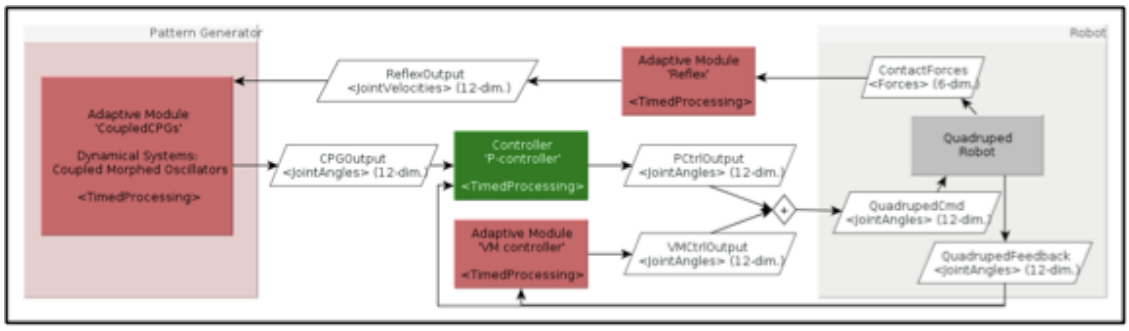



Figure 14: DSL-view on architecture for quadruped walking on rought terrain (EPFL-A, Sec 2.1.1.)

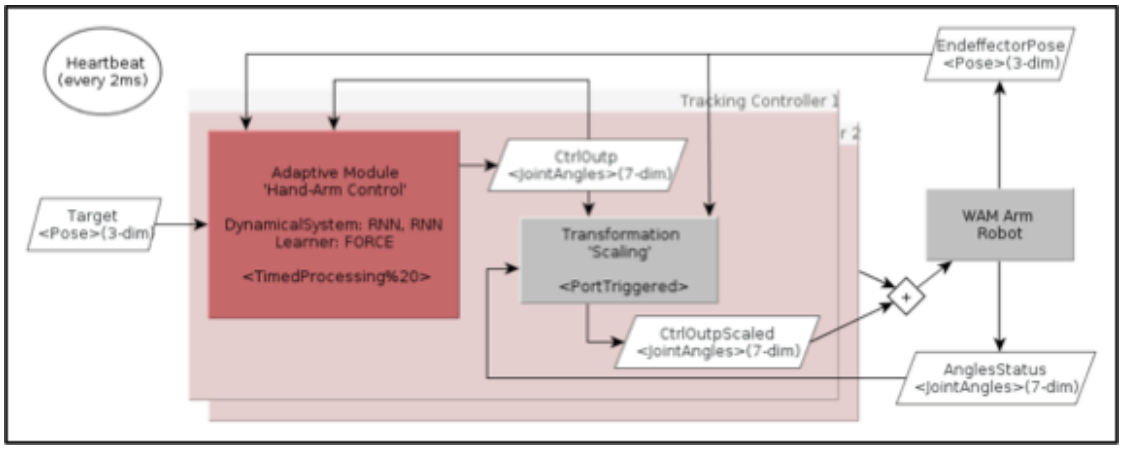



Figure 15: DSL-view on the MOCAP architecture for a mixture of controllers architecture to learn inverse kinematics (UGent, Sec. 2.3.2).

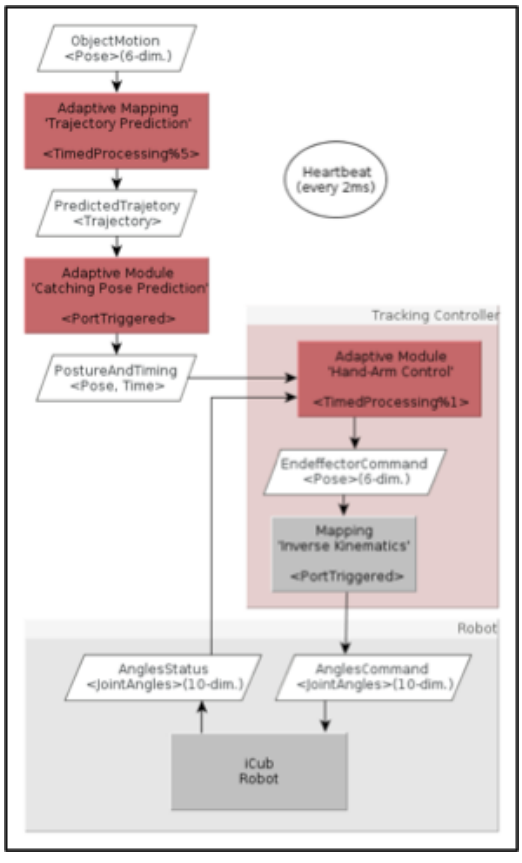



Fig 16 DSL-view on catching objects in flight for the simulated iCub (EPFL-B, Sec

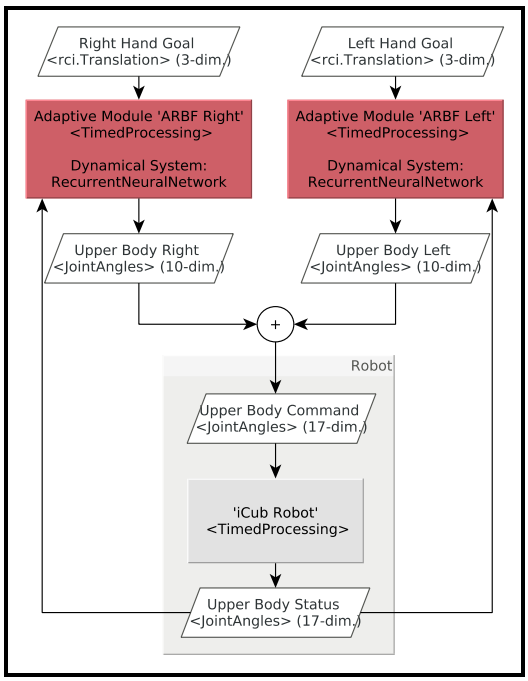



Fig 17 DSL-view on distributed architecture

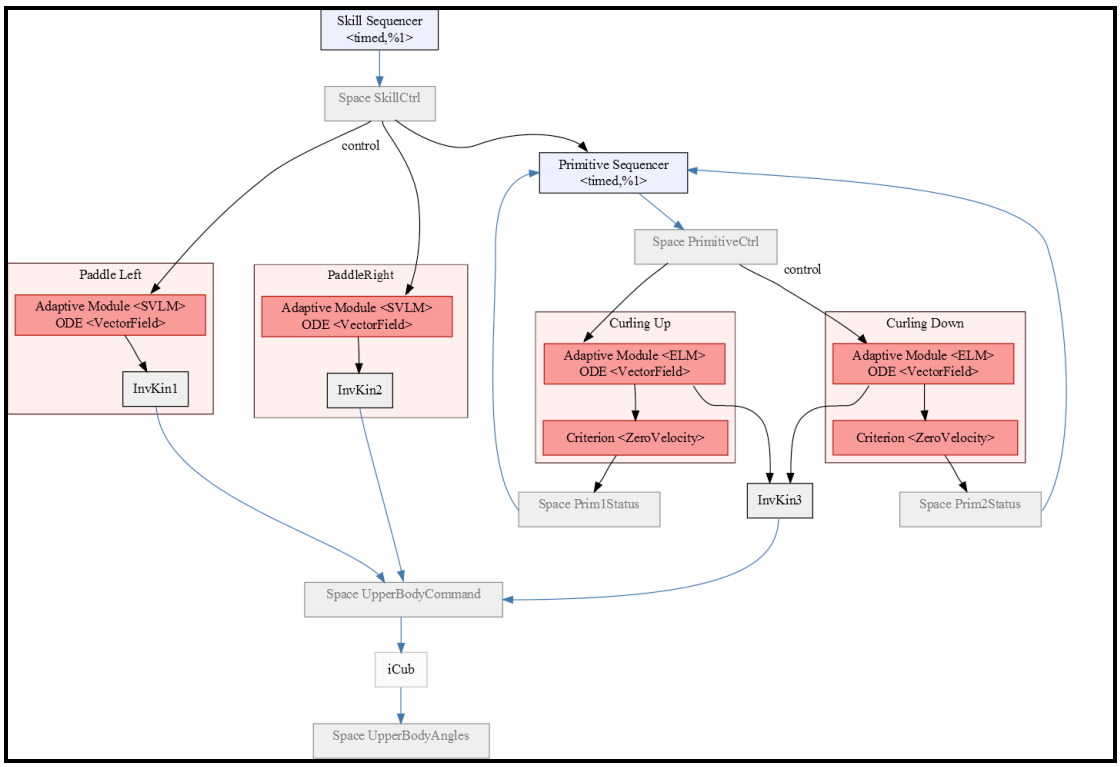



Figure 18: DSL-view on the architecture for bi-manual mainpulation of a stick (UniBi, Sec. 2.5.1). It includes a double hierarchy of sequencers for combination of up-down into one skill on a lower and a skill sequencer on a higher level  The graphics were generated from earlier version of the DSL-tools and shown in the year 2 review meeting.

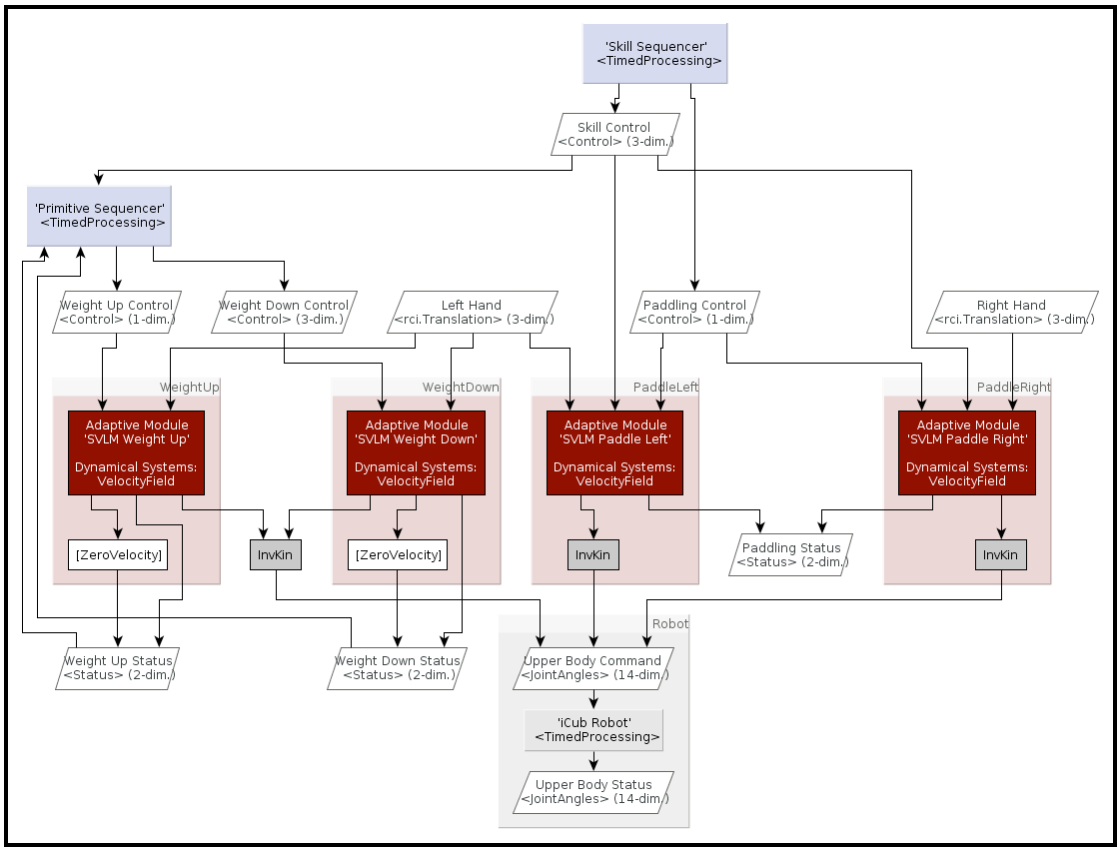



Figure 19: DSL-view on the same UniBi architecture, current version for comparison.

A number of obversations can be made from the DSL formulations. The first and most important one is that the essential concepts of control spaces and the nesting of adaptive modules together with some standardized control logic in adaptive components are sufficient to model all the different architectures, which target very different skills and robots. The DSL therefore captures a level of description where skills are rich but variety of modules is relatively small. This confirms to some degree the overall AMARSi assumption that architectures shall be modular and commonalities can be found if dynamical systems representations are used.

On a more technical level, all the architectueres work on joint-angle space for providing targets for level 0 controllers. However, feedback is very important for bottom up modulation, in Fig. 14 to directly modulate the online adaptive CPG and the adaptive stabilizer, in Fig. 15 to adapt the weighting online, in Fig. 16 and 17 to take into account the inaccuracies of the iCub, in Fig. 18 (and 19) to signal completion of a primitive to the sequencer. The architectures therefore now operate in closed loop conditions. Figs. 16, 18 (and 19) clearly show hierarchical schemes where higher-level controllers modulate (Fig. 16) or sequence (Fig. 17) the level 1 adaptive modules. A further common theme is the additive superposition of control signals in Figs. 14, 15 and 17. This is interesting, because simple superposition of signals representing differrent control objectives is often infeasible in classical control architectures. We suspect that the combination of the compliance in the hardware together with the flexibility of the dynamical systems representations to counteract disturbances allows this simple solution to integrate different signals in the AMARSi architectures. The arguably currently most complex architecture (Fig. 19) involves already a nested hierarchy where one sequencer is used to combine two simple basic movements (move-up and move-down) into a more complex one („weight lifting“), which then is sequenced by a yet heigher level. Nevertheless, the figures show also that all the architectures are not yet very complex on the conceptual level, despite being challenging in the details of the adaptive mechanisms, the learning algorithms and the dynamical systems as has been discussed before and is also apparent from the reports and publications.

Note that the DSL as such does not directly support software and code generation and rather works on a conceptual level. Only where a full tool chain is in place, DSL-based software development can be used to directly transfer ideas and solutions between architectures and platforms. We work hard towards this on the Oncilla (see D7.5), however, it is currently beyond reach to support all the diverse AMARSi platforms in this manner. Nevertheless, from the conceptual framework we can still learn that partial solutions for particular problems, for instance the adaptive partitioning for learning inverse kinematics given in Figs 15 and 17, may quite directly be added to the architectures in Figs. 16 or 18, where an inverse kinematics is used, but not learned. In DSL-language, this is very easy to see and realize.

Nevertheless, this is only one part of the story. The other part is that the overall functional behavior of the robot (the skill) depends a lot on details of the timing, the way the feedback modulates the dynamical systems, the learning etc. UniBi has shown, in a rather systematic evaluation of the architecture in Fig. 17, that even simple and quite subtle changes in the overall architecture can have a strong impact on the realization and learnability of a complex skill [15]. Therefore, the richness in the skill is deeply routed in the detail of the architecture, despite an sometimes misleading conceptual simplicity that is captured by the DSL framework.

# 3 Discussion

## 3.1 An Eagle's Eye View of Cognitive Architectures

When one speaks of a "cognitive architecture", one will typically be guided by an intuitive picture of a hierarchical or modular architecture whose structure follows a coherent organizational plan. Such "orderly principled" architectures have been proposed in numerous disciplines:

- In **classical AI** and **knowledge-based systems**, the prototypical architecture is a *semantic network* which allows to structure knowledge in a hierarchy of concepts which are vertically related by abstraction/subsumption and laterally by relations (which in turn may be ordered by abstraction). A key concept is inheritance of properties along the abstraction links.
- In **connectionist neural networks** and **cognitive science**, the classical AI picture is basically adopted and turned into abstraction networks where the processing dynamics is delegated to "spreading activation" variants (instead of invoking logical theorem proving engines like in AI). A conspicuous example is Shastri's SHRUTI architecture [20].
- In **control engineering**, hierarchical control architectures have been proposed where higher levels of control modulate lower levels, or set targets for them. A prime example is Albus' control architecture [21], which even has become an official U.S. industrial standard.
- Similarly, **behavior-based robotics** architectures typically strive for a principled method to establish a hierarchy of control levels by learning or design. The stage-setting example is Brooks' subsumption architecture [22]. Interestingly, Brooks came from a control engineering background, and used an AI term to name his approach.
- In **complex pattern recognition systems**, especially in speech recognition and computer vision, the raw input pattern is transformed into a hierarchy of increasingly abstract features. Typically, a given engineered pattern recognition system uses a single feature extraction mechanism through all layers, for instance HMM representations in speech recognition or filter convolution and spatial averaging in LeCun's neural networks [23].

There are also prominent cognitive architectures which do not immediately fit into this picture. For instance, Anderson's ACT-R architectures [24] are made of a number of modules with very distinct functionalities (like long-term memory, planner, sensor preprocessing, motor control). However, a closer look will often detect that the authors have strived to work out a unifying operation principle or basic representation format to connect the diverse modules in a cohesive way. In ACT-R, for instance, this unifying role is realized by a universal rule-based representation and processing format.

Of course, there are also elaborate and high-performing cognitive architectures which are made of a multiplicity of modules with heterogeneous representation formats and processing algorithms. An instructive example is Thrun's autonomous desert-crossing car Stanley [25]. One may also consider any industrial plant together with its control and monitoring center as a cognitive system – for example, a power plant or a

chemical processing plant. Such systems exceed in complexity any academic "cognitive system", and they are (like Stanley) thoroughly heterogeneous.

If one boldly abstracts from this brief survey, one is led to the apprehension that academic and theoretical research is naturally geared toward "cleanly principled" accounts of cognitive architectures, while large-scale real-world systems that simply have to perform well are engineered following a "whatever works well will be used" strategy.

## 3.2 Cognitive Architectures @ AMARSi: Status in a Nutshell

At the time when the AMARSi proposal and Annex 1 were written, we were certainly guided by the "academic" mindset. We envisioned that after an initial stage of AMARSi-internal comparison of, and competition between "archetype architectures", a single best such architecture would be determined and used for *the* AMARSi demonstrators on the iCub/COMAN and Cheetah/Oncilla. Quoting from Annex 1, description of task T.6.3: *"... an integration of results from "friendly competition" of partners ... into one "best merge" architecture or, with respect to several targeted tasks and platforms, to integrated architectures."*

If one looks at the achievements at the end of year 3, highlighted in Section 2 of this report, it is clear that this has not happened. In each Lab where robots are programmed, and for each complex skill that is addressed, different and highly differentiated solutions have been worked out. The only (strong and important) connecting link is the integrative software design support (WP7) with its unifying software construct of an adaptive module and the design tools incorporated in the Domain Specific Language (DSL). This unification is however technological, not conceptual. The adaptive software module specification is very general and accomodates the entire variety of theoretical solutions for CPGs and controllers that has been developing throughout the consortium. The impact of the integrative software design support tools is not a theoretical unification of "the" AMARSi architecture, but the technical (highly non-trivial!) possibility to transfer conceptually heterogeneous modules across the consortium.

## 3.3 AMARSi 3.0: Embodied Cognition with 50+ Degrees of Freedom

After Rodney Brooks had staked out the behavior-based research programme for robotics in the 1980ies, the themes of *embodied* or *situated* cognition have permeated the foundational and methodological discourse in AI, ALife and robotics, with repercussions in philosophy. The basic tenet of *New AI* is that the only way to achieve a fundamentally appropriate understanding of intelligence [26] is to follow the directives of natural evolution and re-construct intelligence in a bottom-up way, starting from insect-like robotic creatures that must be, before anything else, *autonomous.* This programme has been very productive and continues to spawn a plethora of "behavior-based" robots.

However, after more than two decades, one is bound to observe that

- typical behavior-based robots today do not look much different from their pioneering prototypes. Specifically, they only have few DoF bodies (often 2DoF wheeled vehicles);
- these robots are not autonomous at all in any serious interpretation of the term;
- any really high-performing (competition-winning or commercial) robot has design elements that are not trained/evolved in a bottom-up way, but rely on classical methods from control engineering or knowledge-based systems;
- lifting intelligence from simple reactive robots to "higher" levels is not theoretically understood. Specifically, higher levels do not *emerge* from a basic reflex repertoire in any reproducible or predictable way.

The situation is, altogether, slightly schizophrenic:

- On the one hand, the embodied/situated cognition programme still has an unfaltering appeal and generates much productive research in fields like ALife, theoretical cognitive/neuroscience, and "bio-robotics", – so there seems to be something about it that is fundamentally right – ,
- while on the other hand, any high-performing robot, which exhibits traits of higher levels of cognition, relies on some "hybrid" design, where some lower levels of controls may conform to the behavior-based paradigm, but higher levels are largely engineered – so there seems to be something fundamentally incomplete in the embodied/situated cognition programme.

At the time of writing the AMARSi proposal, its authors were certainly influenced by the behavior-based view on autonomous robots. The general perspective endorsed at that time can be summarized (simplified) as follows: "the AMARSi project will start from the concept of motor primitives (seen as dynamical systems) and find an architectural principle of how these can be hierarchically and sequentially organized to yield a rich repertoire of skills". This way of thinking has traits of the behavior-based paradigm in that it starts the design *bottom-up* from motor primitives, in that these are realized by *dynamical systems*, in that an *evolutionary development toward increasing complexity* is targetted, and finally, in that a *unifying architectural principle* was sought.

On the other hand, AMARSi was never squarely committed to a pure bottom up framework. Only three of the partners (UniZu, JAC and, arguably, EPFL-A) had a rooting in that school of thinking. The other robotic partners (UniBi Cor Lab, Ugent, EPFL-B, IIT) have their foundations in engineering, machine learning, signal processing and control, and the proposal and Annex 1 bear many traces of this, for instance to enable sufficient control of novel compliant actuation in the first place, which then is the basis for further learning and interaction architectures.

Before we continue discussing the current state of research in AMARSi, we want to review an illuminating case study from ethology, which has some lessons to tell for our project. We refer to work from Auersperg et al. [27], where two bird species were compared with respect to their cognitive skills. The two species were the New Caledonian Crow and the Kea (a parrot), both of which are known for their diligence in using tools, equalling the performance of the great apes in many respects. Altogether nine individuals of these species were submitted to a varied set of cognitive benchmark tasks which involved sequential problem solving and innovative

tool use. Generally both kinds of bird performed (impressively) well on these tasks, with the parrots on average being a little more successful than then crows, or solving the tasks in less time.

As a main contribution, the article by Auersperg et al. describe in detail the differences of the cognitive skills and "approaches" taken by the two species. In sum, the outcome of this discussion is revealing in that it has no definite outcome. The authors trace down a large number of factors that explain the highly variable phenomenology of the bird's task solving behavior. These factors are quite heterogeneous in their nature. To give an impression, here is a selection:

- The interindividual differences within a species are of the same order of magnitude as the interspecies differences.
- The body physics (especially, shape of beak) co-determines which tool manipulations are easy or hard. Specifically, the crows use stick tools in the wild and their beak has evolved to easily handle sticks. Also, because the crows naturally use stick tools, they more often took advantage of available sticklike tools than kea.
- The kea generally exhibit a larger variability in trying out things, and changed to new "ideas" more quickly (tentatively explained by their strong neophilia [attraction to anything new, "curiosity"], which in turn can be explained by keas having no natural predators in the wild). In contrast, a greater degree of neophobia in crows hindered them from high-rate exploration – they were "too cautious to be fast".
- Kea preferred to explore the task apparatus haptically, while the crows depended more on visual inspection. Again, this was tentatively explained in turn by the greater neophilia of the former, and the relatively greater neophobia of the latter, which appears to be related to different predator pressures in the respective natural habitats.
- Quote: *"Another strong difference was that the kea showed greater destructiveness than the crows, as a consequence of their forceful and frequent use of pulling and tearing actions."* The fact that kea frequently pull and tear on objects was related to their natural foraging strategies; the fact that crows don't pull and tear was related to the shape of their beaks which is unsuited for tearing (but in turn is optimized for stick use, which again relates to their natural foraging behavior).

The authors summarize, *"Our study illustrates the difficulties of comparative cognition research and points to some partial solutions. Clearly, no single-task exploration can be used to assess problem-solving ability or make claims for advanced general intelligence or innovativeness. This caveat applies to within as well as between species comparisons. Problem solving is intrinsically multi-dimensional and it is to be expected that individuals or species will outperform each other in different dimensions."*

We extract from this study the following points:

- A key difference in natural habitat (predator pressure) may explain differences on a neophilia/neophobia scale, which in turn co-determines what behavioral options are available.

– Likewise, evolutionary shaped differences in bodily aspects co-determine behavioral options.
– The authors neither expected, nor searched for, or found, general principles of cognitive performance; instead they attempted to track the observed behavioral variability down to differences in the ecological niches and contingent evolutionary development paths.

Returning to our AMARSi discussion, we can learn from this animal study that cognitive systems, which look rather similar from afar (both kea and crows solve the same task battery with comparable success), turn out to be quite differently organized when inspected at close distance. While each individual cognitive system appears internally coherent and analysable, no general principle of "cognition" is apparent which would subsume all such systems.

This view resonates with the experiences gained in AMARSi so far. We witness that our original aspirations toward a unified architecture have been faltering – at least, if the idea of a unified architecture is understood to imply conceptual or design simplicity. It appears that *rich skills require rich architectures*. Furthermore, it appears that robustness and flexibility of a single skill require complex regulatory and adaptation mechanisms. The survey of current developments given in Section 2 amply illustrates the effects of this "evolutionary pressure" toward greater complexity of design.

However, on an abstract level of analysis, our research has been successful in distilling some invariants across functionally heterogeneous architectures. These invariants are the concepts of an *adaptive module* (i.e. that it is possible to modularize rich architectures in the first place), the concept of *control spaces* (i.e. that it is useful to bundle the sensor and actuator variables pertaining to an adaptive module), and the commitment to using *dynamical systems* ([sampled] continuous-time, continuous-value mechanisms) to build adaptive modules. These invariants have allowed us to formulate a *domain-specific language* (DSL) as a shared basis to formalize our respective skill mechanisms, enabling the design of the shared SW development tools which are described under WP 7. While these invariants are not informative about the specific design for individual skills, they represent a technological enabler for joining together different skills in a functioning robot control system.

## References

[1] Ajallooeian, M., Pouya, S., Sproewitz, A., Ijspeert, A. J. (2013), Central Pattern Generators Augmented with Virtual Model Control for Quadruped Rough Terrain Locomotion. IEEE International Conference on Robotics and Automation (ICRA), 2013, Germany

[2] Ajallooeian, M., Pouya, S., Gay, S., Tuleu, A., Sproewitz, A., Ijspeert, A. J. (2013) Towards Modular Control for Moderately Fast Locomotion over Unperceived Rough Terrain. Dynamic Walking, 2013, USA, Submitted.

[3] Shukla, A., Billard, A. (2012), Coupled dynamical system based arm-hand grasping model for learning fast adaptation strategies. Robotics and Autnomous Systems 60, 424-440

[4] Kim, S., Billard, A. (2012), Estimating the non-linear dynamics of free-flying objects. Robotics and Autonomous Systems 60, 1108 – 1122

[5] Kim, S., Shukla, A., Billard, A. (submitted): Catching objects in flight.

[6] Khansari Zadeh, S. M., Billard, A. (2011), Learning Stable Non-Linear Dynamical Systems with Gaussian Mixture Models. IEEE Transaction on Robotics, vol. 27(5), 943-957

[7] Khansari, M. (2012), A Dynamical System-based Approach to Modeling Stable Robot Control Policies via Imitation Learning. PhD Thesis, EPFL

[8] wyffels, F., Li, J., Waegeman, T., Schrauwen, B., and Jaeger, H. (2013). Frequency modulation of large oscillatory neural networks.

[9] Sussillo, D., Abbott, L.F. (2009), Generating Coherent Patterns of Activity from Chaotic Neural Networks. Neuron 63, 544-557.

[10] Li, J., Jaeger, H. (2011), Minimal Energy Control of an ESN Pattern Generator. Technical report 26, School of Engineering and Science, Jacobs University Bremen

[11] Waegeman, T., wyffels, F., Schrauwen, B. (2012), Towards a Neural Hierarchy of Time Scales for Motor Control, in Simulation of Adaptive Behavior (Springer LNCS 7426), pp. 146-155

[12] Waegeman, T., Hermans, M., and Schrauwen, B. (2013). MACOP modular architecture for control with primitives (submitted)

[13] M. Haruno, M., D.M. Wolpert, D. M., Kawato, M. (2001). MOSAIC model for sensorimotor learning and computation. Neural Computation 13(10), 2201–2220

[14] Barliya, A., Omlor, L., Giese, M. A., Berthoz, A., Flash T. (2013). Expression of Emotion in the Kinematics of Locomotion. Exp Brain Res. 225(2), 159-76

[15] Reinhart, F. R., Lemme, A., Steil, J. J. (2012), Representation and Generalization of Bi-manual Skills from Kinesthetic Teaching. IEEE-RAS International Conference on Humanoid Robots, Osaka, 2012

[16] Reinhart, R. F., Steil, J. J. (2012), Learning Whole Upper Body Control with Dynamic Redundancy Resolution in Coupled Associative Radial Basis Function Networks. IROS: IEEE, pp. 1487--1492, 2012

[17] Soltoggio, A., Lemme, A., Steil, J.J. (2012), Using movement primitives in interpreting and decomposing complex trajectories in learning-by-doing. Proc. 2012 IEEE International Conference on Robotics and Biomimetics (ROBIO 2012)

[18] Ivanchev, J. (2013), Fast time scale modulation of pattern generators realized by Echo State Networks. Technical Report Nr 28, Jacobs University Bremen

[19] Friston, K. J., Daunizeau, J., Kilner, J., Kiebel, S. J. (2010), Action and behavior: a free-energy formulation. Biological Cybernetics 102(3), 227-260

[20] Shastri, L. (1999), Advances in SHRUTI – a neurally motivated model of relational knowledge representation and rapid inference using temporal synchrony. Artificial Intelligence 11, 79-108

[21] Albus, J. S. (1993), A Reference Model Architecture for Intelligent Systems Design. In: Antsaklis, P. J. and Passino, K. M. (eds.), An Introduction to Intelligent and Autonomous Control, Kluwer Academic Publishers, 27-56

[22] Brooks, R.A. (1089), The Whole Iguana. In: Brady, M. (ed.), Robotics Science, MIT Press, Cambridge, Mass., 432-456

[23] LeCun, Y., Bottou, L., Bengio, J., Haffner, P. (1998), Gradient-Based Learning Applied to Document Recognition. Proceedings of the IEEE 86 (11), 2278-2324

[24] Anderson, J. R., Bothell, D., Byrne, M. D., Douglass, S., Lebiere, C., Qin, Y. (2004), An integrated theory of the mind. Psychological Review 111(4), 1036-1060

[25] S. Thrun et al (2006), Stanley: The Robot that Won the DARPA Grand Challenge. Journal of Field Robotics 23(9), 661-692

[26] Pfeifer, R., Scheier, Ch. (1999), Understanding Intelligence. MIT Press

[27] Auersperg, A. M. I., von Bayern, A. M. P., Gajdon, G. K., Huber, L., Kacelnik, A. (2011), Flexibility in Problem Solving and Tool Use of Kea and New Caledonian Crows in a Multi Access Box Paradigm. PLoS ONE 6(6): e20231.