EU – FP7

AMARSi

Adaptive Modular Architectures for Rich Motor Skills

ICT-248311

D 6.2

September 2012 (30 months)

Technical report on dynamic extensibility methods

Authors: Herbert Jaeger (Jacobs University), Mostafa Ajallooeian (EPFL- A), Aude Billard (EPFL-B), Thomas Schack (University of Bielefeld), Felix Reinhart (University of Bielefeld), Francis wyffels (University of Gent)

Due date of deliverable	1st October 2012
Actual submission date	1st October 2012
Lead Partner	Jacobs University Bremen
Revision	final
Dissemination level	Public

Introduction

At the time of writing the project proposal and subsequently the Technical Annex, the topic of extending a robot control system by new adaptive modules was primarily considered in the light of the distinction between extension by human design vs. extension by robot-autonomous learning. We quote from the description of deliverable 6.2. from the Technical Annex:

Given some operational architecture, methods have to be developed to extend it by new adaptive modules while keeping it viable. All partners will develop their architecture toward two modes of extension: (i) the addition of externally specified modules – important for engineering and applications, (ii) the autonomous differentiation of the existing architecture into a larger number of adaptive modules. In both cases, the extension will have a structural aspect (how and where to insert the new module and initialize its internal and communication parameters) and a behavioral aspect (in what behavioral setting the new parameters are quickly adjusted).

In the meantime, as Amarsi research has led to increasingly flexible and complex motor behavior systems, this early view has differentiated substantially. In particular, it has become clear that no clear-cut distinction can be made between a gradual enrichment of an existing motor pattern and the insertion of an entirely new one. We will argue below that the growth of a motor repertoire is analog to how evolution proceeds: first there may be a slight differentiation of an existing pattern into variants (the metaphor would be races of a species), which over time may lead to a segregation into clearly distinct patterns (metaphor: branchings in the evolutionary tree of species). Therefore, in our overview of relevant ongoing work in Amarsi, we report on both "gradual enrichment" and "plain addition".

Some further nontrivial issues have become apparent:

- A single adaptive module may be controlled into different functional modes (by inducing bifurcations but also in other ways). This leads to a conceptual distinction between adaptive modules (as encapsulated computational mechanisms) and motor primitives (as external descriptors of functional behaviors).
- When the motor repertoire grows, a motor "primitive" cannot be described or understood in isolation anymore. It will be executed in varying contexts determined by other motor patterns which precede it, follow it, or overlap in time. These variable contexts will induce a rich execution time variability in a motor primitive, again blurring clear-cut definitions.

This report is structured as follows. First we provide an overview of technical research in the robotics partner groups which has a bearing on extensibility (Section 1). Then we will attempt an in-depth discussion of the engineering and conceptual problems which have now emerged more clearly, and suggest a guiding framework to steer our future investigations (Section 2).

1 Overview of relevant technical research

1.1 EPFL-A (Ijspeert)

We report on two lines of research in Ijspeert's group which are relevant for our topic, (i) the differentiation of an existing central pattern generator (CPG) module by stochastic optimization, yielding a "family" of CPGs with a shared "ancestor"; (ii) the creation of behavioral sequences by changing control parameters in a CPG such the qualitatively different behaviors result.

Differentiation of an existing central pattern generator module. This research line arose from experiments with the Cheetah robot. The starting observation was that when one has a working CPG-based gait controller, it is not trivial to use that controller to induce forward speed changes. The naive solution – i.e., to simply speed up the CPG by adjusting its time constant – quickly leads to instable gaits. Different forward speeds, even when the same basic gait is used, require re-adjustment of shape-determining parameters and re-calibration of sensory feedback gains. More specifically, in this line of work gait controllers were used which were designed according to the *dynamical movement primitive* (DMP) scheme (detailed e.g. in Chapter 2 of deliverable 4.1). DMP based gait generators transform the raw oscillation of an underlying Hopf oscillator into a set of target trajectories for the various joints. The DMP parameters allow one to modulate frequency, amplitude, offset, waveform, and sensory feedback gains. For quadruped locomotion, each leg is controlled by a separate such DMP system, which in turn are mutually coupled through phase-coupling their underlying oscillators. Let θ denote the set of parameters for the controller. Then is, if one wishes to control a robot through increasing forward speeds for an experiment duration $t = 0 \dots T$, where at time $t = 0$ the speed is slowest and at time $t = T$ it is fastest, one needs a path $\theta(t)$ through parameter space wich provides efficient and stable gait variants for each speed. The approach taken to obtain such parameter paths $\theta(t)$ is to apply a stochastic optimization search. One starts from a working controller for the slowest speed $V(0)$ at experiment time 0, with an associated functional parameter set $\theta(0)$. By particle swarm optimization search a working solution θ(0 + δ) is found for a slightly higher speed *V*(0 + δ). This search is based on evaluating the performance (speed and stability) of parameters θ in simulation. The procedure is iterated, until a parameter path θ(*t*) is established. Figure 1 shows an example of how the search area in parameter space is being explored, and a progression through the space is established. The work is part of an ongoing PhD project with no publications yet.

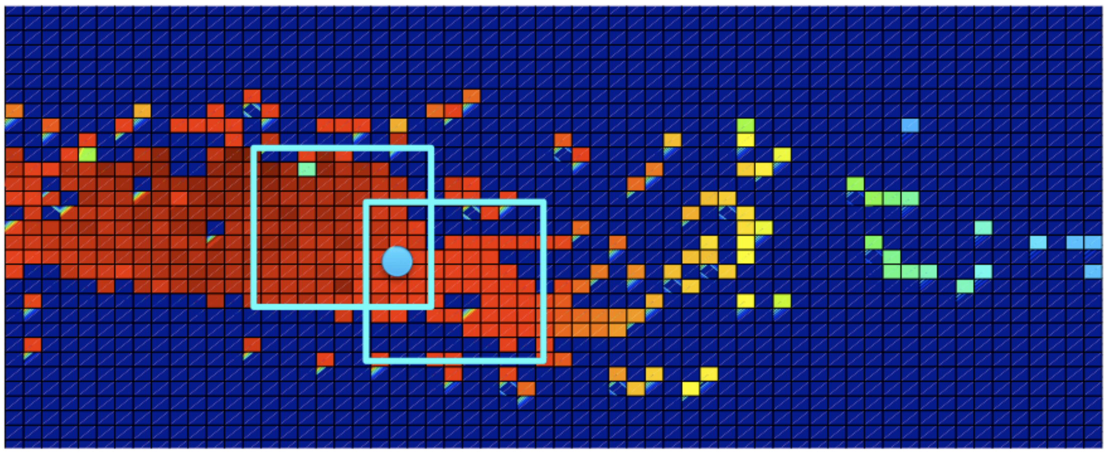


Figure 1: snapshot from swarm-based evaluating parameter settings for increasing speed in a DMP controller. (Taken from a PhD project presentation by Mostafa Ajallooeian)

Creation of behavioral sequences by changing control parameters in a CPG. This work has been detailed before in deliverable 7.2, so we are brief. We mention this work here because it is instructive for our theme. In this research, a simulated Cheetah robot was equipped with a CPG which was obtained by the *dynamical movement primitive* (DMP) method [1]. Such CPGs root in a Hopf oscillator which can be pushed through a bifurcation by adjusting a control parameter r_0. On the one side of the bifurcation, the CPG exhibits a single stable fixed point, which renders the CPG useful, e.g., for controlling a reaching motion. On the other side of the bifurcation a stable oscillation is generated, e.g., as a basis for a trotting gait. A second control parameter *g* allows one to change the location of the fixed point (or the center of oscillation, respectively). Both in the oscillatory and the point stabilization regime, assured stability properties are obtained. A single instance of this "double-faced" CPG is employed to control a sequence of two behaviors, where the robot first trots along, then stops and points a foreleg to a target. A handcoded *sequencer* module generates the necessary slow dynamics of the control parameters r_0 and *g*, the first of which induces the transition from trotting to pointing and the second of which determines the pointing direction in the second phase. Figure 2 depicts how the core CPG (before further transformations needed to steer the motor apparatus) transits between different regimes.

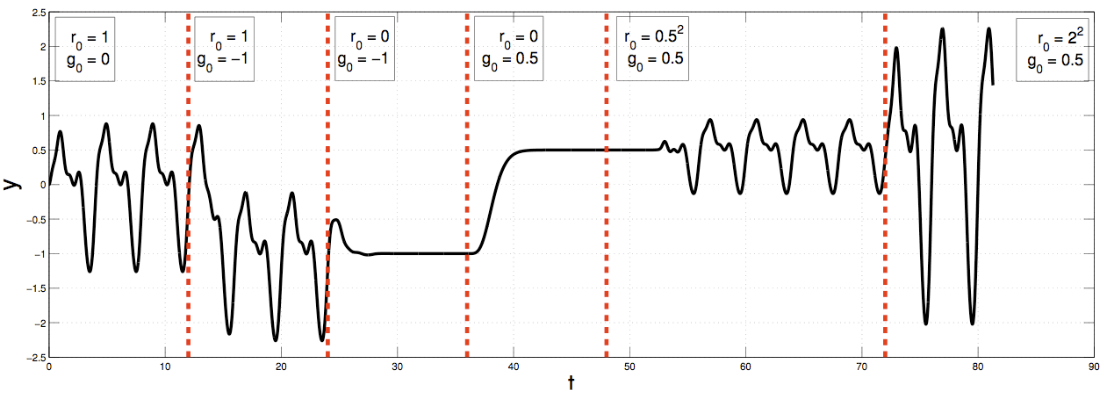


Figure 2: Output of the modulatable "two-faced" core CPG, subject to different settings of the two control parameters. (Re-used from deliverable 7.2)

This example is illuminating in that it points to a conceptual difficulty: namely, a segregation of mechanism from behavior. Intuitively, as an outside observer one would refer to trotting and pointing as two distinct behavioral entities (and naturally assume that they are controlled by different adaptive modules). However, here we see that such apparently very distinct "behaviors" can spring from an identical, simple CPG module. We will comment on this *multifunctionality* theme in Section 2.

EPFL-B (Billard)

In the group of Aude Billard, two lines of work contribute to extending adaptive modules: (i), the addition of obstacle-avoidance capabilities to an existing, functional motor primitive, and (ii) the adaptive combination of separately learnt motor primitives. Both approaches are demonstrated with iCub reaching motions which are designed using the SEDS method, likewise developed in Billard's group. We briefly recapitulate essentials of SEDS and then present the two lines of work.

The SEDS approach to motor behavior learning. This method has been detailed in previous deliverables and a number of publications (e.g. [2]), and we are brief. SEDS stands for "Stable Estimator for Dynamical Systems" and represents a learning method by which a dynamical systems representation of a discrete motion can be learnt from a small number of demonstrations. The core idea 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. This vector field represents a dynamical system (DS) which in the exploitation phase delivers target trajectories for the modeled motion. The crucial benefits of this DS are that due to its rooting in machine learning, it generalizes well from a small number of training demonstrations, assures stability properties, and that it enables a fast recovery from perturbations without the need for explicit replanning.

A gradual enrichment mechanism: incorporating obstacle avoidance into a motion primitive. A situation which arises ubiquitously in robotics is that a learnt motor pattern is confronted, in a particular application situation, with obstacles that have to be avoided. Many solutions to this problem have been proposed in the literature. When the native target trajectories are generated from a DS, these solutions typically place repellor-like objects into the vector field at the obstacle locations. These repellors prevent the target trajectories from approaching the obstacle. In this tradition, Billard's group has developed a particularly refined mathematical approach for adjusting the native vector field. It admits to place convex forbidden regions into the field, such that the resulting dynamics preserves the stability properties of the original system, and also preserves the locations of stable fixed points (e.g. the targets for reaching). Figure 3 gives one schematic example. This work is documented in more detail in [4] and also in deliverable 7.2. The method is independent on how the native vector field was obtained, and applies to both discrete and periodic movements. The critical avoidance objects can be analytically described, matched/fetched from an object database, or be modeled from point clouds on the fly. The computation of the adapted vector field only requires small computational resources and can be done in a few milliseconds, enabling fast reaction to moving obstacles. In [4] numerous demonstrations are provided, analytical ones and with a redundant robot arm (one example in Figure 4).

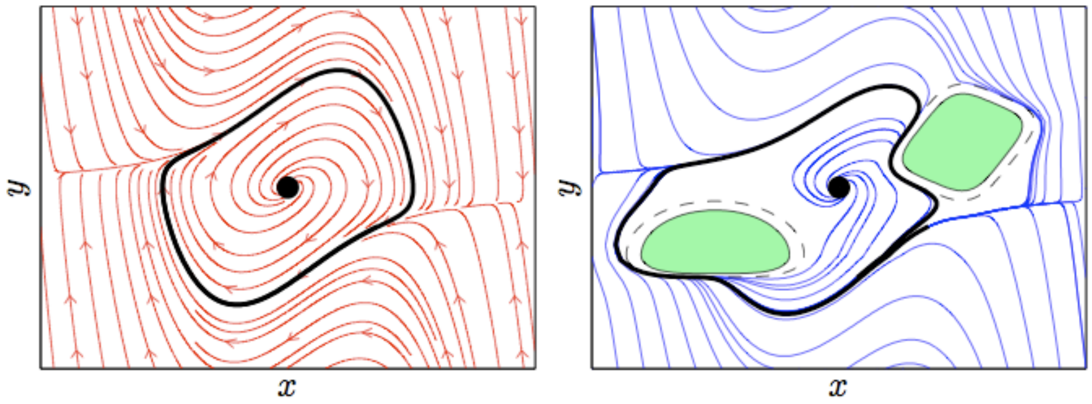
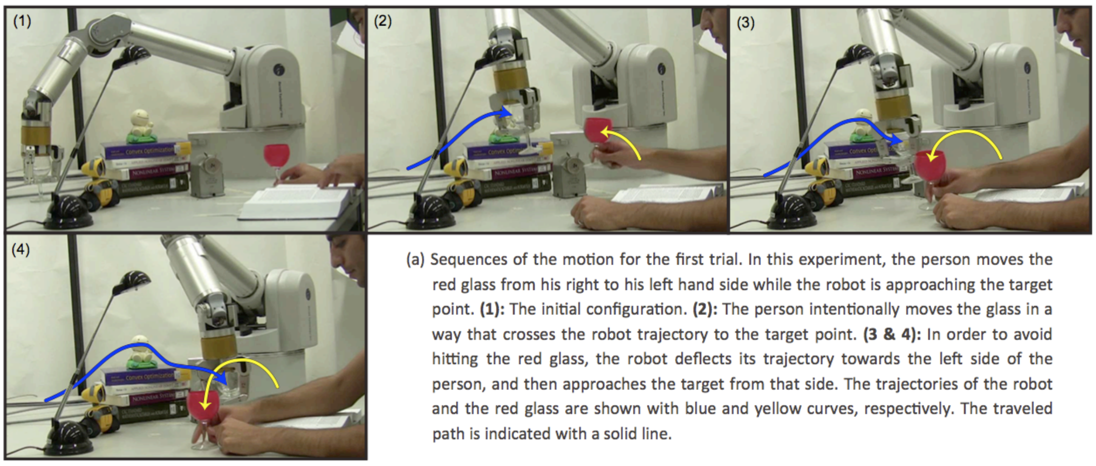


Figure 3: Example of the post-hoc vector field modulation method from EPFL (Billard group). Left: native (originally learnt) vector field, yielding a stable periodic motion. Right: the vector field after two obstacles, with a safety margin each, have been added. (Taken from [4])



(a) Sequences of the motion for the first trial. In this experiment, the person moves the red glass from his right to his left hand side while the robot is approaching the target point. **(1):** The initial configuration. **(2):** The person intentionally moves the glass in a way that crosses the robot trajectory to the target point. **(3 & 4):** In order to avoid hitting the red glass, the robot deflects its trajectory towards the left side of the person, and then approaches the target from that side. The trajectories of the robot and the red glass are shown with blue and yellow curves, respectively. The traveled path is indicated with a solid line.

Figure 4: Robot demonstration of the obstacle avoidance method. See text in image for explantation. (Taken from [4])

Coupling reaching with grasping motor primitives. EPFL (Billard group) has investigated how reaching motions (defined and trained for an entire arm) can be combined with grasping motions (defined and trained on the hand/finger level). It is known that when a human reaches for an object to be grasped, the hand/fingers start preparing for the grasping act while the reaching motion is underway. The two motion control systems are temporally coupled in a systematic way (references in [3]). More 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, this seemingly leads to the following design strategy alternatives:

- First train separate adaptive modules for A and B, respectively, then add suitable coordination mechanisms.
- Train the coupled A + B system directly.

Both alternatives appear not to be ultimately promising. The difficulty with the first one is that the requisite coordination mechanisms can be expected to be nontrivial, especially with regard to stability, requiring substantial insight and engineering in each and every new application case. The problem with the second approach is an explosion of training complexity (the addition of number of concerned state variables leads to a multiplication of required numbers of training samples).

In this apparent impasse situation, EPFL-B explores an intermediate approach. The strategy is to first train A and B individually from a small number of demonstrations of the combined A + B behavior, using the SEDS methodology. Then, use the available training data to estimate an essentially 1-dimensional coupling function Ψ, again based on a probabilistic GMM 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: when A finishes, B finishes too (in the reaching case: when the hand arrives at the target position, the fingers have completed their pre-shaping for the grasp), (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.

The CDS model has been first introduced and demonstrated in a reach-and-grasp task setting where the grasp objects are stationary or slowly moving, and where the pose of the grasp object is fixed ([3], funded outside Amarsi). In one of the demonstrations (on the physical iCub), there were two variants of B to choose from (pinch grasp for a thin object vs. power grasp for a bulk object). When the grasp object is exchanged by the experimenter while the reach-grasp motion unfolds, the finger preshaping adapts on the fly too, simply following the target trajectories supplied by the CDS, with no need for explicit replanning (Figure 5).

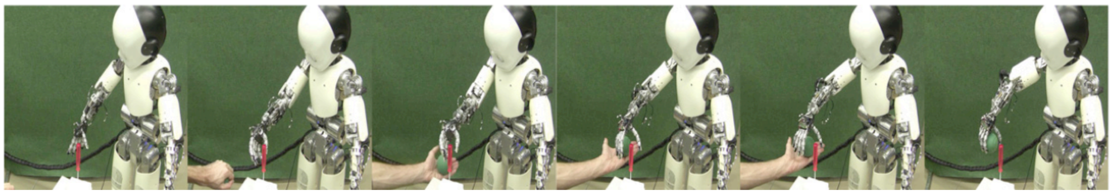


Figure 5: Online adaptive change of finger preshaping from pinch to power grasp, while arm reaching motion smoothly continues. (Taken from [3])

Current research at EPFL-B is extending this CDS approach to tasks where the grasp object is quickly moving while changing its pose. An extreme case is catching flying, rotating objects of asymmetric shape. As an important step toward this demanding task, EPFL-B has developed methods for the fast online estimation of the pose of such objects from video input [5]. In very recent work [6], imitation learning is used to allow learning of skills enabling the robot to catch objects in flight. There, the human user helps the robot to acquire this in an incremental manner by dividing the tasks into different modules. The robot first learns to estimate the dynamics of flight of the object by observing the objects being tossed about 20 times. The robot then learns to compound primitive behaviors for reach and grasp motion through human demonstration. Finally, the robot learns where to place its fingers on the object through static demonstration of a human placing her fingers on the relevant part of the object, while the object lies flat on a surface. Figure 6 shows a demonstration.

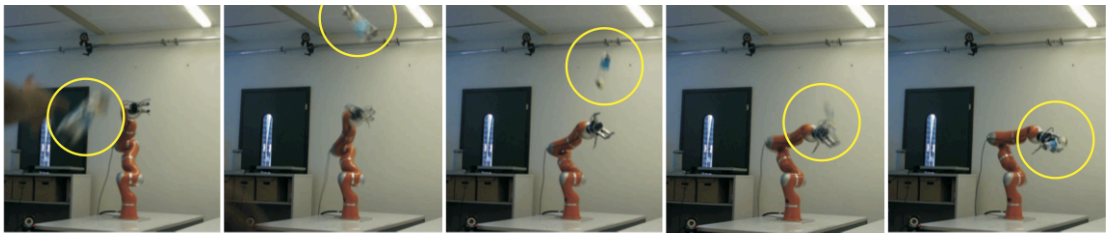


Figure 6: a KUKA robot arm catches a bottle thrown by a human from about 3 m distance. (Taken from [6])

UGent (collaboration with Jacobs)

One important line of work at the Reservoir Lab in Gent concerns methods by which an existing neural CPG can be made modulatable for characteristics like amplitude, shift, waveform or frequency. This general objective is similar to research goals pursued at EPFL-A and EPFL-B, but a crucial difference lies in the mathematical nature of the used CPGs. At EPFL the CPGs are instantiated as analytical ODEs (the DMP methodology at EPFL-A) or as ODEs extracted from training data via intermediate GMMs. In contrast, the pattern generating systems explored in Gent are implemented as (large) recurrent neural networks (RNNs), following the *reservoir computing* (RC) paradigm. ODE based models and RNN based models have complementary benefits:

- ODE based models (often) admit an analytical guarantee of stability properties and are lightweight in their computational costs at exploitation time. Speeding-up or slowing-down of such pattern generators can be easily achieved by modulating the ODE's time constant.
- RC based pattern generators can be easily trained to exhibit almost arbitrary waveforms. Arguably they are closer to biology than ODEs (admittedly a hairy argument). It is possible to train *multifunctionality* into a single neural CPG: by simple control input settings such systems can be made to switch between entirely different output patterns ("neural CPG", documented in D.4.1 Section 9). Furthermore, RC based pattern generating modules can be run bi-directionally, incorporating in a single dynamical system a model for forward as well as inverse kinematics (this line of investigation is pursued at UniBi, see below).

An intriguing difficulty with RC based CPGs is that it is not straightforward how to speed them up or slow them down. Of course this could be easily done if the used neurons are individually modeled by ODEs (e.g. as leaky integrators): then one could tune the neuron's time constants and achieve a proportional speed change in the entire network. But, first, often neuron models without a time constant are employed, and second, it is not biologically plausible that neurons should have a tuneable (and externally addressable) time constant.

Accordingly, it turns out that while it is rather easy to modulate a neural CPG with respect to *geometric* characteristics like waveform, amplitude, or offset, it is hard to find robust methods to modulate its main *temporal* characteristic, i.e. its frequency. Here we report on ongoing work concerning this challenge. This research has a bearing on extensibility because one starts by training a fixed-frequency periodic pattern generator, which then is afterwards made frequency-controllable by adding (and training) a separate control loop.

The basic control scheme is the same as the one that is used for modulating the (easier) geometric output pattern properties. It has been documented before in deliverable D.5.1, so we will only give a brief resume. First, a reservoir is trained as an oscillator: at its output unit it then produces a fixed sinewave pattern. Then an external observer of the quantity that is to be made controllable is added – it outputs an online measurement e.g. of the oscillator's amplitude, shift, or frequency. Based on this observer, an external feedback control loop is added – in early versions a simple P-controller. It is fed with a reference signal for the target observable and can influence the ongoing reservoir dynamics by inserting a bias vector **c** which is weighted with the error signal (in the case of a simple P-controller). The efficiency of this scheme hinges on finding a bias vector **c** which, when weighted-added, influences the critical observable. Figure 7 depicts this general architecture. A number of different methods for training this vector have been developed. The universal finding is that amplitude and shift are easy to control in this way while frequency is hard to master. Very recently however, research at UGent has been successful in also making frequency controllable. The keys to success are relatively large reservoirs (800+ units) and a careful regularization – fostering structural stability of the oscillation – in the training of the native oscillator (the FORCE learning scheme introduced in [8] was found to work particularly well). In this way, frequency became tunable by a factor of 3.

The body of experience collected about modulating neural oscillator patterns by external controllers has now reached a level where it can be brought to bear on robot motor control tasks. In [9] it is demonstrated how the core oscillator is trained by demonstration to control an Oncilla robot leg. The external controller here is not a simple linear feedback controller anymore but is itself realized as a nonlinear, online adaptive RC-based controller (based on the methods described in [10]). The controller generates a multi-component output waveform whose offset and amplitude can be controlled. The adaptivity of the controller allows the system to accomodate to perturbations – in [9], a weight of 100g was attached to the leg at some point. After adaptation, amplitude and offset control were largely re-gained.

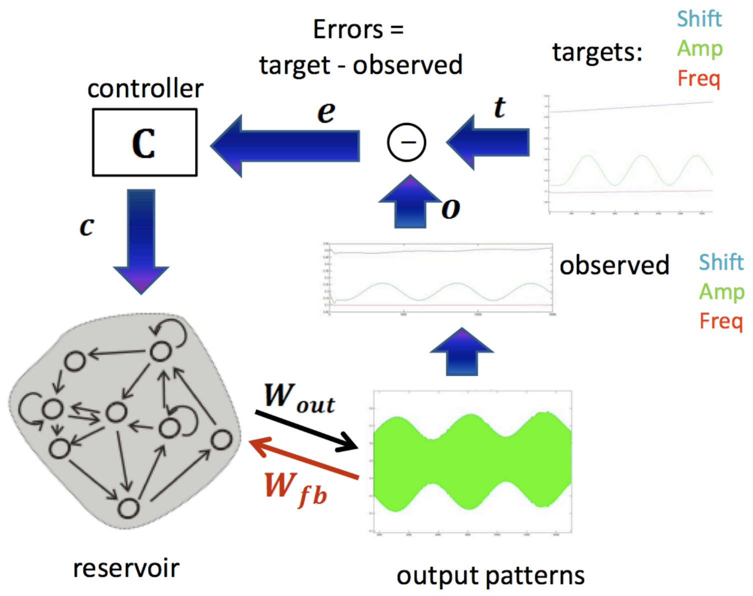


Figure 7: general layout of controlling a reservoir-based periodic pattern generator. (Taken from [7])

UniBi (Cor Lab)

Several lines of work at the Cor Lab have a bearing on extensibility. Generally speaking, like in the current work of other partners, the complexity and modulation richness of motor patterns realized at the Cor Lab is continuously increasing. Here we report on two strands of research. First, a hierarchical analysis of how a complex behavior can be understood to result from a cascade of transformations and modulations was carried out, which led to particularly efficient training from a small number of presentations. Second, the question of how separate motor patterns, which command mostly "private" joint variables but also share some of them, can be coupled in a way that leads to natural-looking combined execution and does not require a separate training for the coupling. We sketch these two lines of work in turn, and conclude with a brief note on sequencing methods employed at the Cor Lab.

Cascaded transformations of movement primitives. In this line of ongoing research [11], UniBi investigated how a maximally reduced (hence, maximally invariant/general) representation of a movement primitive can be expanded and modulated in stages, until it finally yields a runnable motion control loop for a specific task setting. The demonstration scenario involves bimanual object manipulation by the iCub robot: handling a long rod in the fashion of weightlifting or paddling. These tasks are learnt from human demonstrations. UniBi models this control task by a series of task representations of increasing complexity, linked by modulations and transformations. The order of these transitions between representations is variable to some extent. Different orders of cascading lead to different generalization properties and requirements on the amount of training data.

More specifically, the approach taken uses the following representation formats and transformations / modulations:

- The core and most compact representation describes the movement in task space as a *movement primitive* with normalized positions, using only the coordinates **g** of the "leading" hand. Technically, the movement is cast as a dynamical system, whose vector field is trained into a particularly fast-trainable neural network of the "Extreme Learning Machine" (ELM) type [12].
- The execution speed is adjusted by modulating the movement primitive representation by a scalar speed gain α.
- At some point in the cascade, this single-hand based representation has to be *expanded* to full two-handed task space coordinates **p**.
- The geometrical adaptation of the movement primitive to the particular task instance (shift and orientation transforms **H**) can be applied at the level of the core primitive **g** or later in the expanded representation **p**, yielding **g'** or **p'**, respectively.
- Finally, the full-sized task-space representation **p'** has to be transformed to joint coordinates **q** through an *inverse kinematics* solver, which simultaneously performs redundancy resolution. This solver is likewise realized as an ELM.

All three of the core movement primitive, the task expansion, and the inverse kinematics are trained in ELMs, using the same set of human demonstration data. A number of alternatives for sequencing these stages have been investigated. They are shown in Figure 8. In variants (a) and (b), the first two stages are identical and yield full-sized and task instance specific task coordinates **g'**. In variant (a), a single further transformation which combines the expansion with the inverse kinematics is learnt, whereas in (b) these stages are delegated to separate ELM modules. Version (b) can benefit from the structural bias that these stages can be separated and connected by the intermediate representation **p'**, and hence has been found to offer better generalization than (a). Version (c) switches the order of expansion and geometrical adaptation in (b) and thus can benefit from yet another valuable structural bias, namely, that the expansion can be done invariantly from the geometric adaptation. In the ensuing robot experiments, (c) was found to be the superior strategy w.r.t. generalization. Figure 9 gives an impression of the final performance under (c).

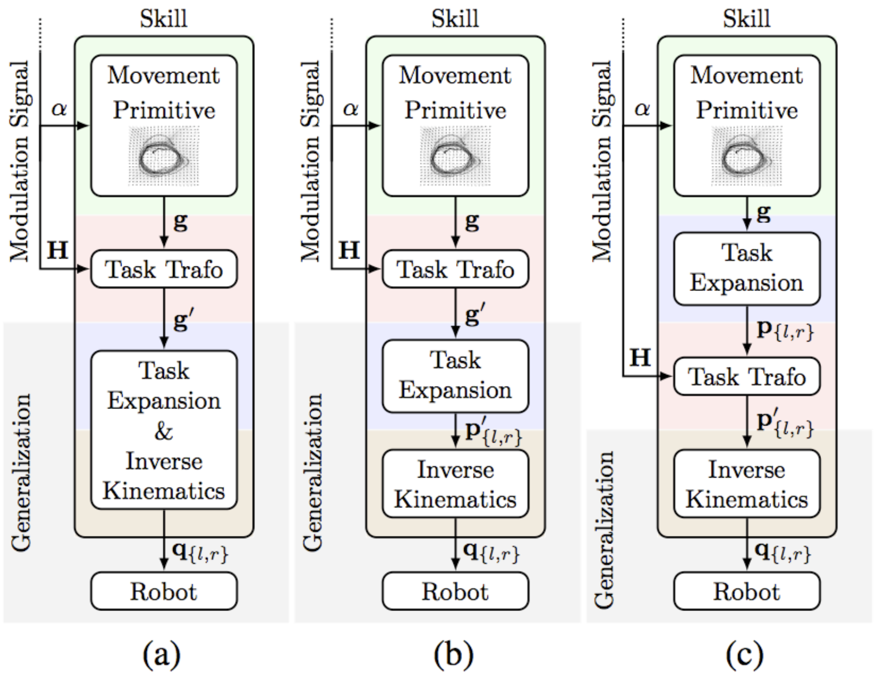


Figure 8: three investigated schemes for cascading transformations which expand and modulate a core invariant movement primitive to a runnable task controller. For explanation see text. (Taken from [11])

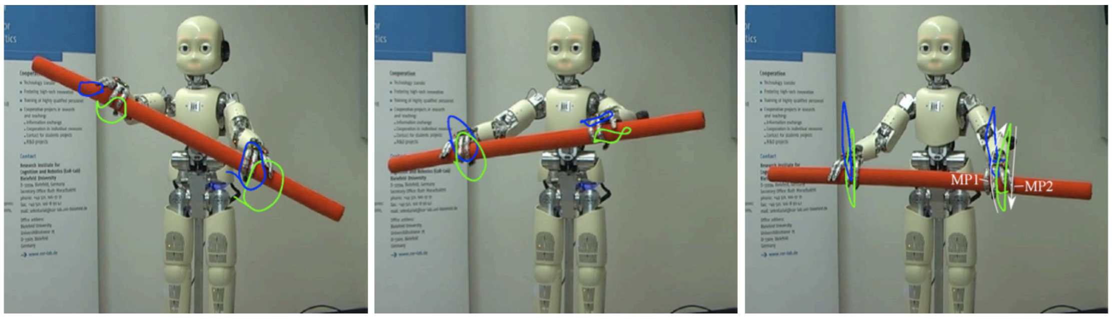


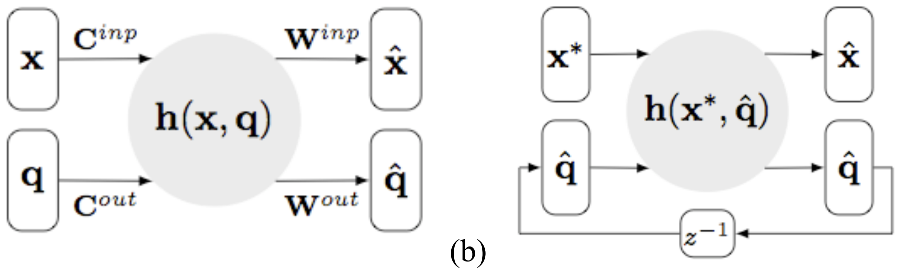
Figure 9: Execution of three learned skills according to the architectural design in Fig. 8 (c): Two periodic paddling skills taught on the left and right side of the robot are depicted in the left and middle image, respectively. A sequence of two discrete motions forms a weight lifting skill (right). Green lines are the end effector trajectories in the training area and blue lines the generalized skill displaced by (0, 5, 5) cm from the training condition. (Taken from [11])

Semi-automatic extension of skill library: The robotic system implemented at UniBi facilitates the semi-automatic creation of skills and automatic addition of the new skill to the overall architecture. For this purpose, a quite general procedure has to be followed: First, training data has to be acquired (here by recording joint angles by kinesthetic teaching through a human tutor). Then, the data is processed and several important features of the motion are detected. For instance, whether the taught motion is a periodic pattern or comprises a sequence of discrete primitives and, also, which bodyparts participate in the motion. This analysis is in principle fully automatic, but - for safety reasons - the human operator is asked to confirm the results. This renders the overall procedure semi-automatic. With the data and detected features, an automatic construction and learning process is triggered (following the ideas of automatic module creation proposed in D6.1). Ultimately, the new skill is added to the architecture and can be exploited immediately.

Coupling body subsystems which share joints. 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; likewise, a pointing gesture primarily involves the arm/hand subsystem and not so much the legs. 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 (Cor Lab) 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 [13]. That is, 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. Thus, the two controllers shared three joints in their respective 7-dim control spaces. The objective of this research was to couple the two controllers in a non-disruptive and natural way in the exploitation phase.

Before we explain the coupling and the demonstrations, we sketch the design of each partial controller. The core component for each 7-dim subsystem is a bidirectional representation of the inverse + forward kinematics. Its basic functioning principles are analog to the RNN based bidirectional controllers explored previously in this group ([16], also documented in deliverable 4.1 Chapter 6 and deliverable 6.1 Section 1.2). For better computational and statistical efficiency, the previously used sigmoid-unit RNN-based representation has now been replaced by a feedforward RBF network. The network is trained on inputs consisting in exhaustively sampled pairs of 7-dim joint coordinates **q** and 3-dim task space coordinates **x**. The trained output values are again the same 7-dim and 3-dim coordinates. When the trained network is used in an output feedback mode where the desired task space target **x*** is clamped to the **x** inputs and the estimated joint values $\hat{\mathbf{q}}$ are fed back to the input, a redundancy-resolved inverse kinematics trajectory is obtained. See Figure 10 for a schematic.



(a) (b)

Figure 10: (a) Setup for the bidirectional task/joint coordinate training. C^{inp} and C^{out} are fixed input weights, W^{inp} and W^{out} are trained output weights. The mapping **h** is

realized by a RBF network and trained on the identity mapping. (b) Exploiting the trained network for computing inverse kinematics by partial output feedback. (Taken from [13])

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 11 shows how it was done at UniBi. 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. Like in Figure 10 (b), the controllers are run in a joint space feedback loop; the values that are fed back are the measured actual joint readings.

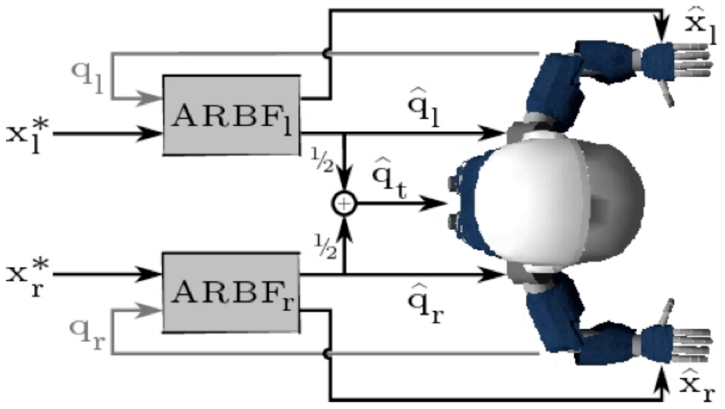


Figure 11: combining the right and left arm/torso controllers. For explanation see text. (Taken from [13])

This negotiation scheme leads to natural-looking interactions between the two arms. Figure 12 depicts some demonstrations. In (a), the task target was to bring both hands to positions left front to the robot. In (b), the same target position was requested for the left hand, while no target was prescribed for the right hand (this is realized by closing the upper **x**-feedback loop in the right arm controller). These two different sets of requirements lead to different degrees of turn and bend in the torso. Similarly, in (c) the two arms had to reach out far in front of the robot, while in (d) only the left arm had to do this. Again, this leads to natural-looking differences in torso posture.

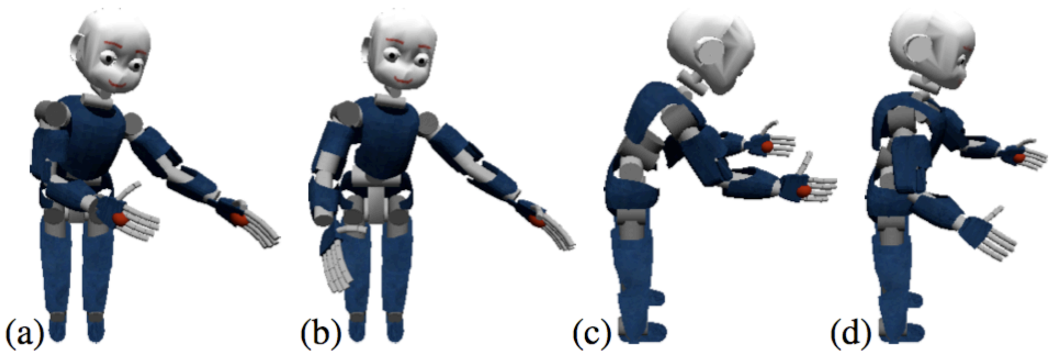


Figure 12: demonstration of combined right/left arm/torso controllers. See text for explanation. (Taken from [13])

Sequencing motor primitives. A short note on how motor primitives are currently sequenced at UniBi. This task arises frequently in various lines of work. It may refer to sequencing very elementary motor primitives, condensing them into what one might call a "behavior" – an example is the weight-lifting pattern sketched in Figure 9c, which is composed of two sequenced directed movements. Or it may refer to organizing sequences of higher-level behaviors. All of these sequencing issues are currently dealt with by hand-coded sequencer modules. A standard finite-state machine serves the sequencing of movement primitives, where convergence of a discrete primitive triggers the transition to the next state. Here, convergence of a movement primitive means the reaching of a target position, which is monitored by a simple criterion (implementing the concepts laid down earlier in D6.1). More sophisticated representations of sequences, e.g. Hidden Markov Models [14], can in principle be plugged into the architecture. However, this simple state transition scheme together with bottom-up feedback of actual joint angles renders the architecture responsive to perturbations and assures that a primitive is finished before the next primitive starts. Smooth transitions in speed between successive primitives can be achieved by additional mechanisms, e.g. switching primitives before they terminate [15].

Resume

Here is a summary of our walk through extension-related research in Amarsi:

1. The primary level of extending motor repertoirs is the *differentiation* of an existing adaptive module. We saw instances of this in
   a. the establishment of a path in module control parameter space by stochastic swarm optimization, enabling walking speed adaptation (EPFL-A),
   b. the ad-hoc modification of DS vector fields by convex obstacle inclusions (EPFL-B),
   c. frequency modulation of a large RNN oscillator by an external control loop (UGent & Jacobs),
   d. transformation cascades which enrich a core low-dimensional DS representation by task- and joint-space variables.
2. On a higher level of extension complexity, Amarsi is investigating two aspects of *coupling* two simpler motor patterns into a more complex one:
   a. coupling hand/finger pose preparation to an arm reaching motion by a coupling phase variable (EPFL-B),
   b. coupling two single-arm motion controllers into a two-arm controller by averaging the control input to the shared torso joints, plus dynamic negotiation by feedback dynamics, a procedure enabled by bidirectional representations of foward/inverse kinematics (UniBi).
3. Having an available repertoire of more than one motor behavior immediately leads to *sequencing*:

a. This can sometimes be achieved by driving a single adaptive module through a bifurcation (EPFL-A).
b. At UniBi, separate sequencer modules (hand-coded as yet) assemble elementary motor primitives into a functional behavior ("weight lifting" example), or assemble high-level functional behavior sequences.

This list indicates that under the headline of "extension" we are tapping a large variety of phenomena, mechanisms, and scientific approaches. In the next section we endeavour to draw a more systematic picture of this field.

2 Some coordinates in the research space of motor repertoire extension

At the time of formulating the Amarsi proposal and the Technical Annex, our view of what it means to extend a motor repertoire was conceptually straightforward:

– One would start with an initial architecture that hosts *a few* adaptive modules, basically engineered by the human designer, with possibly some learnt optimization within or between the modules.
– Then, further modules would be incrementally *added*, either by the human engineer or by some autonomous learning scheme.
– A few obvious challenges were foreseen, especially the two related questions of (i) how new modules should be coupled into the existing system, and (ii) how should the execution of an increasingly rich repertoire of available modules be scheduled / sequenced / planned? The rough early ideas in these two respects were that (i) is just a matter of tuning coupling constants (by hand or by automated learning / optimization), and that (ii) will be effected by some high-level planner (e.g. a finite state machine or an AI-style symbolic planner), or by human commands. At any rate, issue (ii) was regarded as peripheral to the project.

We did not (and could not) build on a detailed analysis of the extension challenge at the beginning of the project. But, as of today, the need for a deeper understanding has become clear. And, we are in a much better position to tackle these issues because we afford of a substantial, shared basis of experimental tools and experiences. In this section of D.6.2 we endeavour to draw a much more detailed and better informed picture of the extension challenge (subsection 2.1), and to offer a unifying perspective (subsection 2.2) which will guide our future efforts in this arena.

2.1 The wide and unwieldy field of "extending" a motor repertoire

Just a list of questions in this subsection... with comments but not with answers.

Q1: How do we define / delimit adaptive modules?

This is certainly not a settled question. The multitude of terms used by consortium partners is instructive*: central pattern generator, neural central pattern generator, neural motor primitive control, neural dynamical motion primitives generator, neural* *dynamic movement primitives, dynamical movement primitives, motor primitives, motor skill, motor pattern, adaptive module, behavior* (overview and discussion in deliverable D.6.1). One reason for this variety is that different researchers use different criteria for defining their objects: functional, implementation-oriented, outward-phenomenal, control-oriented. Another reason is that different names may point to different levels of (hierarchical) organization. In the further discussion we will simply use the generic term *module* if we want to cover any or all of these.

An implicit assumption behind the usage of all of these terms is the *discreteness* of modules. When thinking of a motor repertoire, one thinks of a *finite* collection of well-defined modules. In robot control programs, these modules would be represented by circumscribed functions, subroutines, or objects (in the sense of object-oriented programming). It is worth pointing out that this idea of dealing with well-defined, discrete objects is a very strong assumption, and it may be inadequate. One might also conceive of a picture where the observable variety of motor behavior is organized in a continuum, with smooth transformation chains ultimately connecting every "behavior" with every other. Internally, such a continuum of motor phenomena might be subserved by a comprehensively coupled, extensive neural substrate which would be steerable to produce any observable behavior by smooth parametric changes. A glimpse of this possibility became apparent in the parameter paths investigated at EPFL-A. In this line of complex dynamical systems thinking, one may hope to recover discrete items in terms of attractors – this is a standard scientific move to make. But, neural motor control systems are surely strongly co-determined by sensory input. The bad news, then, is that rigorous attractor concepts for input-driven systems are hard to obtain, and mathematical research on such *nonautonomous dynamical systems* is in its infancy. Ongoing research at Jacobs aims at providing useful mathematical foundations for describing such systems [17].

Closely connected to the question of how to define modules is the question of how to *dissect* or *compose* modules. Only very few of the modules explored and exploited in Amarsi are "atomic" or "unitary" in a sense of being represented by a single mathematical formula or a single computational routine. Most modules which are practically used are composite and can be dissected, e.g. into transformation stages (as in the cascaded transformations presented above for UniBi), or into a core CPG plus some added control or sequencing machinery (as in the controlled RNN pattern generators from UGent/Jacobs or UniBi's weight-lifting controller), or into a basic vector field in which obstacle representations can be immersed on the fly (EPFL-B), or into sub-control loops which are coupled by a few shared controlled variables (UniBi's two-arm controllers) – just an arbitrary choice. These ways to compose parts into larger wholes are well motivated in every single case, but overall there is no systematic understanding of what types of composition mechanisms exist and how to describe them formally.

If we assume a composition hierarchy for modules, how might the "high" end look like? How complex may composite modules become while still being accountable as identifiable, stable units? Is there a highest degree of compositional complexity beyond which behavioral modules cannot stably exist – e.g., for reasons of practical unlearnability or inavailability of sufficiently complex control mechanisms at execution time? What would be plausible examples of most complex human motor skills which a naive observer would still conceive as identifiable "wholes" (as opposed to perceiving the observed action as composite)? Is the external observer perspective appropriate to define "wholeness" of executed skills? This seems questionable. Consider a pianist practicing a demanding passage which requires a non-standard fingering which the pianist cannot recall or assemble from his/her large collection of highly trained, stereotyped fingerings. A large number of exercising repetitions of this passage, starting slowly and increasing in fastness, will be necessary until mastery. The entire passage will then be automatically executed after it is started, and experienced by the pianist as a whole. Figure 13 gives an example, which covers 18 third-notes extending over two bars. An external observer *seeing* the printed score might judge this to be a sequencing job of 18 movement "primitives"; an external observer *listening* to the performance may prefer to segment this into six trioles (corresponding to the main characteristic of the rhythmic experience). The pianist – after the practicing – will rather classify it as a unit. Does this example come close to the limits of complexity of human motor modules? It surely is close to the limits of what the writer of these lines (amateur hobby pianist) can attain.

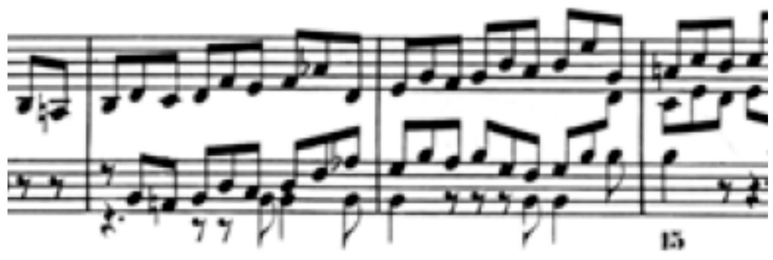


Figure 13: A passage from Bach's prelude in E flat major from the well-tempered clavier, book II. This passage is difficult for the left hand and requires an ideosyncratic fingering which has to be explicitly practiced. (Image taken from www.dlib.indiana.edu)

Q2: How many motor skills does a human have?

This question is not directly critical for Amarsi, because we do not aspire to capture a complete human motor repertoire in our robots. However, it is an illuminating question, closely connected to the first question but shedding a new light on our affair.

To make this question productive, we first have to agree on some criterion of how we delimit motor skills from each other, in order to obtain discrete, countable items.

When I tie my sneakers, the movements and control gains that I employ are different from the ones that are called upon when I tie my heavy hiking boots. But we would rather not want to count this as two different skills – intuitively it seems more appropriate to think of two modulations of one skill. On the other hand, knotting the two ends of a broken kite line together appears intuitively another skill than tying a shoe. Why do we intuitively think so? For the sake of bringing this discussion to life, we propose the following criterion for separating skills: *A motor skill A is different from another motor skill B if mastery of A does not imply good performance in B; B has essentially to be learnt de novo even when A is already mastered.* Of course this learning-effort based criterion leaves gray zones, but it is a helpful first step. For instance, it allows us to clearly separate the following two skills from each other, which on the surface look closely related. The example is taken from the author's hobby world of RC helicopter flying. Learning to remote-control a forward horizontal circling maneuver is one of the first steps in learning to fly an RC helicopter. It is however not simple for a novice and takes some weeks of daily training (and a significant investment in crash repairs). A helicopter is an almost holonomic device and flying it backwards (tail leading) requires almost exactly the same control inputs, except for two sign changes (effects of nick and roll commands are reversed). Now it turns out that for the human RC pilot trainee it is immensely difficult to learn steering circles backward. In fact, the previously acquired skill of forward cycling tends to interfere with the learning, inducing wrongly oriented reactions (and more repair). Also the author can report a distinctly different experiential quality: backward versions of forward maneuvres "feel" entirely different. Altogether, this suggests that backward vs. forward circling should be considered two distinct skills, not modulated versions of one underlying skill. This is an interesting example because from a mathematical or engineering view, the two skills are closely related: one could re-use the same controller, with only two gain signs switched.

Further exploring how far the learning-based definition of motor skills may carry, let us consider sequenced motor behavior. An attack move in the sport of volleyball can be seen as (and is indeed explicitly trained as) a sequence of two or three steps forward toward the net, lowering the CoG while decelerating to prepare a jump, then jump while raising the arm, then hitting the ball, and finally, landing. Compare this to the similarly complex sequence of movements that occur when a polite person hears a knock at the office door: s/he gets up from the chair, takes a few steps to the door, extends the arm to the handle, pushes it down, then opens the door. Interestingly, the attack move has to be extensively trained even though the individual sequence components are well mastered before. In contrast, the door-opening maneuvre does not need any specific training for a person who already knows how to get up from a chair, walk a few steps, etc. The apparent reason for this difference is that the sequencing of component movements for the volleyball attack requires a precise, situation-dependent timing and force control and admits no pausing, while the success of the door-opening maneuvre is almost entirely insensitive to speed and force variations, and allows for pausing. Based on the "definition by required learning" criterion we would classify the attack move a distinct motor skill, but not the door-opening sequence, even though over the years it might be repeated hundreds of times and may look stereotyped to an external observer.

After these propaedeutics, let us return to the question of how many *separately trained* motor skills a human possesses. If one starts thinking about it, one will find that there are numerous "basic" skills which are broadly useful in many situations and which every healthy human acquires very early in life – like crawling, reaching with an arm, bending or shaking the head, sitting up, standing, walking, various grips and hand/finger gestures, etc. It is hard to come up with a narrow estimate of how many of such "generic childhood" skills humans have, but it seems reasonable to say that their number seems larger than 20 and smaller than 1000. Beyond these basic skills, a human is trained or trains himself/herself on a much larger number of composite skills, distinctly addressable by our tentative separate training criterion. These differ with culture and individual life history and would include items like shoelace tying, volleyball attacks, or RC helicopter circling. Modern life environments feature a large number of artefacts, social situations, body care routines, pastime activities, craftmanship and sports involvement, and other dimensions of function and purpose, each of which triggers specific motor learning episodes resulting in distinct motor skills. Again, coming up with a number estimate is daring, but an intuitive guess that seems defendable would put that number larger than 200 and smaller than 100,000. Let us boldly agree that a healthy grown-up human commands of motor skills in the order of 10^3 or more.

Thus, the ultimate challenge for cognitive neuroscientists and roboticists is to understand how a motor repertoire of 1,000+ can be learnt, represented, coordinated and used. This is surely not what Amarsi sets forth to achieve, but Amarsi research should have this ultimate horizon in view and aim for first steps which, in principle, have hooks to scale to this level.

Q3: How many CPGs does a human have?

Of course this question only makes sense to the degree that one specifies what is understood by "central pattern generator". While the term is not precisely defined, its usage in the biological literature (e.g. surveys [18] [19]) is quite consistent and refers to extracortical, localized, small neural circuits which can produce rhythmic output in the absence of rhythmic input. The list of standard examples comprises pulmonary, ingestive and digestive, cardial, eye-saccadic and limb control (mostly locomotor) instances. Only the latter are relevant in our current context of discussion. While we have not found a concrete number stated for humans (or other animals), it appears that the number of attested limb-controlling CPGs is quite small, in the order of magnitude of 10.

Contrasting this small number with the order of 1,000+ of distinct motor skills gleaned from the previous question leads us to some cautionary remarks and further questions:

- Terminological hygiene: roboticists should maybe use the term "CPG" not so liberally for non-periodic motor control modules as they sometimes do.
- Is it likely that all motor skills ultimately root in CPGs? The sheer numerical mismatch in order of magnitude, and the non-periodic nature of most motor skills suggest that this idea is hard to defend. Abstractly, there seem to be two methodological approaches to deal with the CPG vs. motor skill juxtaposition:
    - One may strive to reduce all motor skills to CPGs, ultimately. "Higher" skills, then, would use CPGs as the distal interface to the motor apparatus, modulating them (up to the point where control input to CPGs changes the native rhythmic dynamics to a discrete one), and possibly growing a hierarchical architecture of increasingly complex controllers above the primary CPG level. Much of the work at EPFL-A&B could be seen in this light. Also, the classical engineering perspective on hierarchical control is of this kind ([20]).
    - Alternatively, one may think of architectures where CPGs are just one kind of, but not the basis for all, motor skills. This leads to conceptual heterogeneity and likely diminishes the elegance of architecture models, but may provide more flexibility for learning and design. CPGs may be seen as evolutionary more archaic and stereotyped, while "higher" cortically based skills may be seen as arbitrarily

addable and shapeable. The numerical hiatus between orders of magnitude of 10 vs. 1,000 raises no fundamental question. However, an answer must be found on how the two subsystems (CPGs and cortical control) interact.

Q4: How are modules addressed/selected for execution?

Assuming a discrete, but large (1000+) repertoire of motor skills, it becomes a nontrivial question of how these are selected for execution. This problem does not surface in current-day robotic systems because their repertoire is slim and they have to perform only in limited scenarios for which finite state automata based action schedulers or simple rule-based planners are readily designed by hand. It seems however unlikely that these explicit, discrete schedulers/planners can be scaled to very large repertoires to be used in unmodeled and fast evolving environments.

The problem is aggreviated if motor skills can be modulated (which seems inevitable). Then, in a given evolving situation, not only the applicable skills per se have to be identified, but also their appropriate modulation.

In classical symbolic AI planning systems, action selection is seen as a discrete sequencing task. This task is addressed by creating goal-subgoal hierarchies by means of a symbolic planner. At the finest resolution level, elementary subgoals are attained by executing elementary actions. The set of eligible actions to reach a given subgoal is predefined (and hand-coded). Which candidate action among a set of subgoal-relevant actions becomes "fired" is handled in various ways, using some management scheme for precondition monitoring, heuristics for estimating success, or even random choice. To make all of this work, each available action needs to be tagged by goals that may be achived by the action, and possibly furthermore by enabling conditions and other characteristics. Seen globally, the repertoire of available actions is organized by goals, and it is this goal tagging which renders the actions addressable.

This cleanly-discrete approach of classical AI does not easily transfer to robotic scenarios. Crucial differences to the classical AI scenarios include:

- The environment is partly unmodeled and unpredictable and sometimes requires almost instantaneous, not plannable, reflex-like reactions.
- The robot as well as the environment have a large number of degrees of freedom, which need to be negotiated concurrently and in a graded, gain-calibrated fashion.
- Several motor skills may need to be executed simultaneously, or with gradual fading-in / fading-out.

Challenges of this kind have historically led to the "New AI" approaches of behavior-based robotics, and a rekindled interest in cognitive agent theories that address the immersion of an agent in a dynamic environment, e.g. theories based on concepts of affordances or on concepts of autopoietic self-organization. However, neither behavior-based robots nor such immersive theories have so far been developed far enough to sustain motor repertoires of 1000+ items (multiplied by modulations...).

To make large repertoires function in a situated agent, a computationally efficient structuring scheme for motor skills is needed which makes it possible to identify the current most appropriate skill(s) – plus, possibly, their appropriate modulation – extremely quickly. A hint to be taken from classical AI is that such a structuring may involve some sort of goal tagging. A hint from computer science solutions for database organization might be that a hierarchical tree-oriented organization of the "skill base" may be helpful for fast access.

Considerations of this kind are not immediately relevant for Amarsi, because our motor repertoires will remain small enough to be organized by heuristics on a case by case basis. But it would be desirable to invest in principled organization schemes which have a potential for scalability. Furthermore, a part of the extensibility challenge lies in an answer to the question of "where to place" newly acquired skills.

Q5: How can we define and quantify similarity between motor skills?

One theme in human motor learning research is to investigate *motor transference* and *motor interference*. The first refers to a facilitation of learning a new skill by previously acquired skills, the latter to detrimental effects. A common explanatory strategy is to attribute this to similarities vs. dissimilarities between motor skills. For instance, "learning to crawl could generally facilitate skills like canoeing because of similarities in motor structures (transference), while the learning of higher level, more finely-tuned technique in sports like tennis and badminton may at the same time suffer from interference because of significant differences between the relevant sensory-motor structures" (quoted from [21]; see also).

Such observations lead to considering similarity measures on motor skills. There is no obvious best way to define such a measure. For instance, two motor skills might be deemed similar,

- if they employ the same effectors, or
- if they serve similar goals (which however spins off the question of finding similarity measures for goals), or
- if they are controlled by identical or overlapping neural circuits, or
- if the mathematical descriptions of the resulting mechanical dynamics are similar (leads to the question of finding similarity measures on dynamical systems – topological equivalence of high-dimensional phase portraits might be a guide), or
- if external human observers judge the skills as similar, or
- if they mutually facilitate learning of the respective other.

This list indicates that here we are confronted with a hairy business. The example of controlling RC helicopter circling forward vs. backward demonstrates how involved this issue may be – the two skills would be classified similar by all of the criteria listed above except the last, but there is very strong interference w.r.t. learning between the two.

A similarity measure might be useful for organizing (large) skill repertoires. Feeding a similarity measure to a clustering algorithm could be exploited to arrange the repertoire in a tree whose structure would mirror the similarity clustering of the skills. Such a tree could then become useful for the addressing task pointed out in Q4.

2.2 Toward large and extensible repertoires of motor skills

In this subsection we consider how a truly large (size 1,000+) repertoire of motor skills could be organized in a way that holds some answers to the questions posed above. Such large repertoires are far beyond what is targetted in Amarsi or elsewhere. However, even merely hypothetical thinking in such grand scales may lead to insights that help us to better understand the organization of smaller-sized repertoires, in particular with respect to facilitating their extensibility.

We start by extracting the core challenges from the previous questions section:

- *Discrete set or continuous "skillscape".* How do we delimit skills from each other, given modulation pathways that may gradually transform one skill to another?
- *Complex skills: agglomerates or glues?* Novel skills often arise from the combination of previously available ones. One view on such combinations is to conceive of a limited set of combination operators (sequencing, multi-limb coordination, timed phasing-in-phasing-out) by which available skills can become systematically agglomerated. Another view is that each new skill, if it is worth being called new and worth being memorized, constitutes an innovative whole whose essence lies in a unique dynamical coupling ("glue") of previously available constituents. This coupling has to be explicitly trained and tuned and cannot be understood as a stereotyped application of combination operators.
- *Hierarchical ordering criteria.* It is a common intuition that a motor skill repertoire should be hierarchically organized. There are several natural criteria to define hierarchical "subsumption" relationships. One is *abstraction*: a generic periodic arm swinging (e.g. exhibited by toddlers, presumably due to a "raw" CPG activity) would be considered more abstract than a conductor's highly differentiated swinging of the baton. Another is *composition*: a composite skill is "higher" in the hierarchy than its component skills.
- *Addressability.* There must be almost instantaneous mechanisms by which a situation-adequate skill can be identified. Computer science informs us that an overall hierarchical organization enables fast search. But, it is not clear whether a fast search (e.g. down a decision tree) is what happens in humans. Other options to account for addressability would be attentional mechanisms in which the ongoing stream of situated experience maintains a "torchlight" focus on a small set of currently eligible skills. At any rate, whatever access strategy is deployed, each skill must in some way be connected with or tagged by conditions for executability.

These challenges look strikingly related to issues concerning *conceptual hierarchies* that have been investigated since long in Cognitive Science and AI. A difference is that the hierarchies encountered in those fields are typically made of perceptive concepts, used in pattern recognition tasks and semantic processing of sensory input.

In Amarsi we are confronted with *motor pattern generation hierarchies*. There is a theoretical danger that we might be fundamentally misled if we seek inspiration from conceptual hierarchies. However, the temptation to learn from similarities between the two domains is stronger than a hypothetical doubt. Therefore we will conclude this essay by inspecting one of the main paradigms from Cognitive Science and AI for conceptual hierarchies which may be instructive for our purposes.

Abstraction hierarchies and semantic networks

The core ordering principle for concepts in classical, logic-based AI and numerous CogSci models is *abstraction*. In the extensional view of concept semantics, a concept A is more abstract than a concept B if the extension of A is a superset of the extension of B. This view directly connects concepts to first-order logic, which in turn enables the use of rigorous, powerful and well-understood inference algorithms. The abstraction relation leads to a lattice ordering on the set of all concepts. Along the parent-child links of such trees, inheritance mechanisms enable compact computational representations of large concept spaces. Finally, Boolean operations on concepts allow one to generate new concepts in a transparent and general fashion.

Once a conceptual space is ordered by abstraction, different concepts in the abstraction hierarchy can be "laterally" connected by relations. Relations can often be inherited downwards, leading to additional gains in compactness and transparency. The ensemble of a core abstraction hierarchy, additional relational structure, and inference algorithms forms a *semantic network*. The theory of semantic networks is a traditional field of research in AI and offers riches of insight and tools.

How could robotics benefit from all of this? Are there chances to transfer ideas from semantic networks to motor skill repertoires?

There are two alternative ways of how one could understand this question:

– Given a repertoire of motor skills, one could *describe* each skill by asserting properties that is has; these descriptions then are of the same logical-extensional kind as the concept definitions used in logic-based AI. One could then use available semantic network methods to arrange these *skill concepts* in an abstraction hierarchy, with the associated benefits of inference and search mechanisms. There is a body of evidence from human motor skill research to the effect that humans indeed maintain cognitive representations of motor skills in manual action or other domains which are represented and processed like other object concepts too [22][23][24]. These representations are, so to speak, "about" the skills and mostly elements of the skilled performance itself but must not be confounded with the motor skills at a level of physical entities. They allow a human to reason about his own skills, but they are not directly executable. A blunt example: I know that I have the knee extension reflex, and I know a number of facts about it. But, this explicit conceptual representation is not the patellar reflex itself.
– A more daring modeling effort would strive to order not skill descriptors, but the executable skills themselves (e.g. CPGs, controllers, neural networks, attractors in neural networks) in a hierarchical way that can be formally understood as an abstraction ordering. For sure, engineers have proposed numerous hierarchical control architectures, but the ordering is

defined by composition in these systems, not by abstraction. Potential benefits of an abstraction-analog ordering of motor skills executables are intriguing:

- Such an ordering might reflect a learning history, similar to an evolutionary tree, where more abstract skills are "raw" skills acquired earlier, which then differentiate in a personal learning history, branching into more and more refined, "special" skills.
- In situated action, the abstraction ordering might support a fast adaptation of an ongoing execution of a skill, either by specialization (going down the tree) if the situation allows or requires this; or by abstraction (going up the tree), resorting to a more raw version of the skill in error conditions, enabling a subsequent modified re-try (down again on an alternative specialization branch).
- Acquiring new skills might often be effected by specialization from an existing one. It would be immediately clear where in the overall repertoire the new skill would be placed. Known conflicts or synergies that the parent skill "laterally" relates to other skills may become inherited.

All of these potential benefits are specific to abstraction hierarchies and cannot be obtained from the standard compositional hierarchies.

The crucial ingredient for the second alternative would be to establish a notion of motor skill abstraction. Such a notion should have the same formal properties as extensionally defined concept abstraction. That is, we would need a way to associate with each motor skill a formal *set* of some sort, where the set can be used to *define* the skill, and with set inclusion defining motor skill abstraction.

From defining sets, the Boolean operations of union and intersection would immediately provide elementary constructions of new skills from existing ones. If furthermore there would exist a maximal set that includes all others, we would earn set complement operations, and from that, "negation" of motor skills and the full Boolean logic combinators for creating new skills. This may sound too abstract, and finding a plausible construction of defining sets associated with skills is not immediate.

At Jacobs there is ongoing work close to fruition which aims at exactly this. This work relies on recurrent neural network modules for generating motor patterns. The guiding idea is to use for the abstraction-defining sets, roughly speaking, the set of all network states that may arise during the execution of a given skill. At the time of writing, this approach has already been implemented in proof-of-principle synthetic skill repertoires ordered by abstraction, with the possibility to use Boolean operations to create, on the fly, novel and immediately executable patterns (publication in preparation).

References

[1] Degallier S., Righetti, L., Gay, S., Ijspeert, A. (2011), Towards simple control for complex, autonomous robotic applications: Combining discrete and rhythmic motor primitives. Autonomous Robots 31 (2), 155-181

[2] 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

[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] Khansari Zadeh, S. M., Billard, A. (2012), A Dynamical System Approach to Realtime Obstacle Avoidance. Autonomous Robots, 32(4), 433-454

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

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

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

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

[9] 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

[10] Waegeman, T., Schrauwen, B. (2011), Towards learning inverse kinematics with a neural network based tracking controller. In: Neural Information Processing (Springer LNCS 7064), 441–448

[11] Reinhart, R. F., Lemme, A., Steil, J. J. (submitted), Representation and Generalization of Bi-manual Skills from Kinesthetic Teaching

[12] Huang, G.-B., Zhu, Q.-Y., Siew, C.-K. (2004), Extreme learning machine: a new learning scheme of feedforward neural networks," in IEEE International Joint Conference on Neural Networks, 985–990

[13] Reinhart, R. F., Steil, J. J. (submitted), Learning Whole Upper Body Control with Dynamic Redundancy Resolution in Coupled Associative Radial Basis Function Networks

[14] Kuli, D., Ott, C., Lee, D., Ishikawa, J., Nakamura, Y. (2012), Incremental learning of full body motion primitives and their sequencing through human motion observation. The International Journal of Robotics Research, vol. 31, no. 3, pp. 330–345

[15] Pastor, P., Hoffmann, H., Asfour, T., and Schaal, S. (2009), Learning and generalization of motor skills by learning from demonstration. In: IEEE Intern. Conference on Robotics and Automation, 2009, pp. 763 –768

[16] S. Wrede, S, Johannfunke, M, Lemme, A., Nordmann, A., Rüther, S., Weirich, A., Steil, J. J. (2010), Interactive learning of inverse kinematics with nullspace constraints using recurrent neural networks. In 20. Workshop on Computational Intelligence, Dortmund, 2010. Fachausschuss Computational Intelligence der VDI/VDE-Gesellschaft Mess- und Automatisierungstechnik

[17] Manjunath, G., Jaeger, H. (submitted), The Dynamics of Random Difference Equations is Remodeled by Closed Relations

[18] Ijspeert, A. (2008), Central pattern generators for locomotion control in animals and robots: A review. Neural Networks 21, 642-653

[19] Büschges, A., Scholz, H., El Manira, A. (2011), New Moves in Motor Control. Current Biology 21, R513-R524

[20] Albus, J. S. (1993), A Reference Model Architecture for Intelligent Systems Design. In: Antsaklis, P. J., Passino, K. M. (eds.), An Introduction to Intelligent and Autonomous Control, Kluwer Academic Publishers, chapter 2, 27-56

[21] Schack, T. (2014 – planned), yet untitled contributed article in: Tenenbaum, G. & Eklund, R. Encyclopedia of Sport and Exercise Psychology

[22] Schack, T. & Ritter, H. (2009). The Cognitive Nature of Action – Functional Links between Cognitive Psychology, Movement Science and Robotics. *Progress in Brain research: Mind and Motion - The Bidirectional Link between Thought and Action.* (pp 231-252)**.** Elsevier.

[23] Güldenpenning, I., Koester, D., Kunde, W., Weigelt, M., & Schack, T. (2011). Motor Expertise Modulates the Unconscious Processing of Human Body Postures. *Experimental Brain Research*, 213, (4), 383-391.

[24] Stöckel, T., Hughes, C.M.L. & Schack, T. (2012). Representation of grasp postures and anticipatory motor planning in children. *Psychological Research*. DOI 10.1007/s00426-011-0387-7.