Representation and Generalization of Bi-manual Skills from Kinesthetic Teaching

René Felix Reinhart, Andre Lemme and Jochen Jakob Steil
Research Institute for Cognition and Robotics (CoR-Lab) & Faculty of Technology
Bielefeld University, Universitätsstr. 25, 33615 Bielefeld, Germany
{freinhar, alemme, jsteil}@cor-lab.uni-bielefeld.de, www.cor-lab.de

Abstract—The paper presents a modular architecture for bi-manual skill acquisition from kinesthetic teaching. Skills are learned and embedded over several representational levels comprising a compact movement representation by means of movement primitives, a task space description of the bi-manual tool constraint, and the particular redundancy resolution of the inverse kinematics. By means of comparative evaluation of different architectural configurations, a specific modulation of skill execution with respect to the embedding in the robot’s workspace is identified to achieve optimal generalization performance from few training samples. This representation scheme together with a novel stabilization approach for dynamical movement primitives yields to very robust teaching and execution of a complex sequence of bi-manual skills for the humanoid robot iCub.

I. INTRODUCTION

It is widely recognized that exploiting human knowledge about movement skills either by reproduction of recorded human movements after some remapping [1], [2], by training through teleoperation [3] or by observation and subsequent imitation [4]–[7] is a key to progress towards richer motor skills in robotics [8]. These imitation approaches aim at the generation of an invariant internal representation of the observed movement which needs to be generalized by the robot to adapt to modified or new tasks. Exploiting the compliance of the recent version of the humanoid robot iCub, the presented work contributes to this line of ideas by investigating the learning of complex bi-manual movement skills from kinesthetic teaching.

In most previous work, the observed data is directly given or transformed into a suitable task space [4], [6], [7], [9], in most cases tool tip or hand coordinates or even more task specific coordinates [10], [11]. In this task space, a compact representation of the movement is learned where invariance is achieved by formulation of movements with respect to relative task coordinates. For movement generation, this task space representation is then modulated, for instance transformed according to the visual recognition of an object to be manipulated. The movement is then deployed on the robot via inverse kinematic solvers. This scheme provides high flexibility, in particular for free goal directed movements, and efficient methods have been provided to resolve the redundancies, e.g. [12]–[15].

For bi-manual and even more complex movements involving specific body coordination constraints, the paper argues that the representation of a skill requires more specifications and amounts to at least four aspects: the task space movement, the expansion into bi-manual or other body coordinates respecting particular features of the task, a modulation and transformation binding the abstract primitive to a particular part of the robot’s workspace for generalization, and a redundancy resolution scheme which again respects particular task constraints. In most recent approaches, only the task space representation of the movement is learned [4], [6], [9], [10]. The expansion from the tool tip coordinates, for instance of a manipulated stick, into bi-manual coordinates and a corresponding redundancy resolution scheme must then be incorporated as prior knowledge in form of transformations and constraints in the redundancy resolution for the inverse kinematics solving. In [9] for instance, data from two hands is analyzed for temporal correlations of certain key points. The correspondence between the observed trajectories and the robot’s joint configuration nevertheless needs remapping and standard inverse kinematics.

The advent of compliant robots provides a natural interface for data driven learning of such complex skills on all needed levels of representation. By kinesthetic teaching of the robot, a human tutor can directly guide movements for new skills in physical interaction (compare Fig. 1). By recording of both the task space trajectories and the corresponding internal joint angles, the specific redundancy resolution scheme applied by the tutor and necessary task space expansions, transformations and constraints can potentially all be learned from training data instead of being explicitly and a-priori given.

Note that learning of a complex skill then has to rely on
few data, which calls for non-local learning methods with strong extrapolation capabilities. This paper therefore chooses the Extreme Learning Machine (ELM), a high-dimensional single hidden layer feed-forward neural network with constant and random input weights [16] as representational format, because it can be very efficiently trained by standard regression techniques. Generalization abilities in such data driven schemes naturally scale with the amount and quality of training data. This work will show, however, that generalization also strongly scales with variations of the architecture. It depends on the way how the different partial representations are embedded and with the degree of invariance the partial representations achieve. The paper argues in a qualitative evaluation of different architectures for a particular configuration of representational layers in favor of optimal generalization from few demonstrations. It supports this design decision by comparative evaluation of quantitative learning performances.

Based on these results and a novel neural network approach to stabilization of dynamic movement primitives, it is shown how a complex bi-manual sequence of skills can be learned, blended and generalized on the humanoid robot iCub.

II. REPRESENTATION AND GENERALIZATION OF BI-MANUAL SKILLS

By means of an exemplary skill learning with the humanoid robot iCub (see Fig. 1), we first discuss general aspects of representation for bi-manual movements. The tutor kinesthetically teaches several skills with a stick-like tool comprising “paddling” and “weight lifting” (compare also Fig. 3). Handling the stick requires to respect skill-specific constraints of both hands, i.e. position and orientation of both hands, caused by the coupled kinematic chain in order to master the bi-manual manipulation of the tool.

A. Representation of a Bi-Manual Skill

For this paper, a bi-manual skill is constituted through an interplay of several partial representations, which address particular features of the skill and have different degrees of invariance. Subscribing to the learning from demonstrations approach, a robot and coordinate system invariant representation of the task space movement is aimed at, which then can flexibly executed and generalized. Four representational issues need to be solved:

- **Movement Representation:** Encoding of movements in a task space, which is sufficient to explain the entire motion pattern. The elements of this space are denoted by \( \vec{g} \) and describe the bi-manual motions in a compact way by the end effector trajectory of the guiding hand, that is the hand with the highest spatial variance during the motion.

- **Task Expansion:** Expansion of the current target \( \vec{g} \) from the compact movement space to an explicit task formulation, e.g. end effector coordinates \( \vec{p}_{(t,r)} \) of both arms.

- **Modulation:** Temporal and spatial modulation of the movement, e.g. modulation of the movement’s speed and the embedding of the movement into the robot’s workspace. We denote modulation signals by \( \alpha \) for speed and \( \vec{H} \) for spatial transformation of task space variables \( \vec{g} \) or \( \vec{p} \) by means of homogeneous transformation.

- **Redundancy Resolution:** Mapping from task space specification \( \vec{p}_{(t,r)} \) to joint angles \( \vec{q}_{(t,r)} \) using a particular redundancy resolution scheme.

In simpler scenarios, learning methods for these partial representations have been proposed. For instance, inverse kinematics learning to implicitly represent kinematic constraints present the training data has been shown to be feasible in complex cases [17], [18], including bi-manual manipulation [19]. The learning of task expansions has not been addressed before (to the best of our knowledge), either because simpler skills do not need a task expansion to model constrained movements of body parts or because task expansions were modeled explicitly. An exception is [11], where the expansion from invariant coordinates to robot kinematics is not constant and guided interactively by a tutor. For bi-manual skills, all levels have to be addressed implicitly or explicitly, either by learning or by formulation of particular constraints or optimization criteria.

B. Generalization of Bi-manual Tool-Constraints

Now consider different learning architectures to combine the four partial representations, their impact on the generalization ability, and the respective need for training data. Note that there are two generic interfaces for modulation and generalization that are directly introduced through the chosen representations. Firstly, movement primitives based on dynamical systems can adjust to local perturbations and can be modulated in speed and shape [8]. If sophisticated methods are used, also obstacle avoidance [20] or timing constraints can be implemented [21]. Secondly, the transformation of the movement primitive into a concrete coordinate frame in the robot’s workspace offers the opportunity to scale and rotate it at this stage, e.g. relative to perceptual cues [6], [11]. This generalization across positions in task space can be accomplished before
Fig. 3. Execution of three learned skills according to the architectural design in Fig. 2 (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.

or after expanding the compact movement representation to a full task space representation, which leads to the architectures shown in Fig. 2 (b) and (c). An alternative approach is to learn both, task expansion and inverse kinematics, in a holistic fashion with a single adaptive mapping as shown in Fig. 2 (a). We consider these three cases in turn to discuss generalization ability and learning requirements.

The architecture in Fig. 2 (a) performs modulation first and then combines task expansion and inverse kinematics in a holistic fashion. This amounts to data driven learning of a mapping from 3D task space coordinates $g$ to the full joint space $q_{(l,r)}$ that must be generalized over positions $g$ in 3D robot task space. It represents task expansion and inverse kinematic redundancy resolution implicitly and non-separable. In this case it is in principle also possible to represent a skill involving different tools with different bi-manual constraints, e.g. curved sticks or one-handed and bi-manual execution of the movement. The representational power is largest as the holistic fashion of representation uses the least bias for the learning. However, this advantage needs to be balanced by sufficient training data from all relevant parts of the task space. This is problematic for a scenario of kinesthetic teaching, where each possible configuration must then be manually provided by the teacher. The generalization ability in case of learning from few samples is thus rather limited.

In the second architecture (Fig. 2 (b)), task space expansion and redundancy resolution are now separated, but still each subject to generalization. In principle, the representational power is the same (gray shaded area) as before, but only if data allows for a separate training of the two layers and generalization abilities of both of them are commensurable. That is, if the task expansion $g' \rightarrow p'_{(l,r)}$ generalizes e.g. over different positions in task space, then also the inverse kinematics $p'_{(l,r)} \rightarrow p_{(l,r)}$ from the expanded 6D bi-manual hand coordinates to the joint angles has to be trained on respective compatible data $p'_{(l,r)}$ that are sufficient for generalization. In the experiments, we will see that it is not trivial to get data from kinesthetic teaching, which are sufficient for such generalization of the task constraint, even though the learning task is in principle quite simple for this partial representation because more bias, i.e. prior knowledge about the need of a task expansion, has been added.

Finally, the architecture in Fig. 2 (c) switches the order of performing task expansion and task transformation. In this case, the task expansion is coordinate invariant and performed relative to the coordinate frame of the movement primitive. There is no need for generalization at this stage and the task expansion can be extracted from the data with high accuracy through an initial learning or statistical analysis. This architecture exploits that transformations can easily be done on the expanded coordinates as well, which adds the largest bias to the design of the three architectures. This implies the largest generalization capability for few samples because data driven generalization needs to be established only for the inverse kinematics.

To summarize, from Fig. 2 (a) to Fig. 2 (c) more bias in form of knowledge about how to decompose the skill representation is added. We thereby decrease the demands for training data and accordingly increase the potential for generalization from few training samples, which is typical for kinesthetic teaching. This hypothesis from the qualitative discussion is supported by comparing the generalization performance for the designs (a)–(c) at the end of the next section.

III. Experiments with the Humanoid Robot iCub

In this section, we describe the teaching, learning and exploitation phase in more detail and demonstrate the capability to learn and generalize bi-manual skills from few training samples. Details of the neural learning scheme to implement the learned mappings are moved to the appendix.

A. Kinesthetic Teaching of Bi-Manual Skills

In the teaching phase, a human tutor physically guides the robot in order to generate training data. This phase comprises the following steps: First, the robot grasps a stick-like object by means of a preprogrammed procedure and with the assistance of the tutor. Next, the controllers for both arms are switched to joint impedance mode such that the tutor can move the robot’s arms while it holds the stick, i.e. the
TABLE I
GENERALIZATION ERRORS OF THE ARCHITECTURE CONFIGURATIONS IN FIG. 2 FOR THREE EXEMPLARY SKILLS.

<table>
<thead>
<tr>
<th>Architecture Configuration</th>
<th>Error Measure</th>
<th>Skill</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
<td>Paddle Left</td>
</tr>
<tr>
<td></td>
<td></td>
<td>Left Hand Guiding</td>
</tr>
<tr>
<td></td>
<td></td>
<td>Train</td>
</tr>
<tr>
<td>Fig. 2 (a)</td>
<td>left hand [cm]</td>
<td>0.20</td>
</tr>
<tr>
<td></td>
<td>right hand [cm]</td>
<td>3.55</td>
</tr>
<tr>
<td></td>
<td>hand distance [cm]</td>
<td>0.55</td>
</tr>
<tr>
<td></td>
<td>left orientation [deg]</td>
<td>2.61</td>
</tr>
<tr>
<td></td>
<td>right orientation [deg]</td>
<td>7.50</td>
</tr>
<tr>
<td>Fig. 2 (b)</td>
<td>left hand [cm]</td>
<td>0.06</td>
</tr>
<tr>
<td></td>
<td>right hand [cm]</td>
<td>3.39</td>
</tr>
<tr>
<td></td>
<td>hand distance [cm]</td>
<td>0.49</td>
</tr>
<tr>
<td>Fig. 2 (c)</td>
<td>left hand [cm]</td>
<td>0.24</td>
</tr>
<tr>
<td></td>
<td>right hand [cm]</td>
<td>3.44</td>
</tr>
<tr>
<td></td>
<td>hand distance [cm]</td>
<td>0.50</td>
</tr>
<tr>
<td></td>
<td>left orientation [deg]</td>
<td>2.44</td>
</tr>
<tr>
<td></td>
<td>right orientation [deg]</td>
<td>7.14</td>
</tr>
</tbody>
</table>

iCub complies to the bi-manual tool constraint using the force controller implementation from the iCub software repository [22], [23]. The tutor actively moves both of iCub arms to teach a bi-manual skill and the joint angles of both arms are recorded (compare Fig. 1).

In the learning phase, which is following each demonstration of the tutor, the recorded joint angle trajectories \( q_{(l,r)}(k) \) for \( k = 1, \ldots, K \) of both arms are augmented with the respective end effector coordinates \( p_{(l,r)}(k) \) by calculating the forward kinematics. An automated data analysis is conducted to determine the task space coordinates \( g \) of the guiding hand, i.e. the end effector trajectory with the highest spatial variance. It further checks whether the data can be modeled with a single, periodic movement primitive, or whether the skill comprises a sequence of discrete motions. For this purpose, the trajectory of the guiding hand is segmented based on zero-crossings of the velocities. If we cannot detect points of (almost) no motion, we classify the recorded trajectory to describe a periodic pattern and train a single movement primitive together with the task expansion and inverse kinematics. Otherwise, we extract the discrete motions according to the segmentation and group similar motions together. Based on the segmented teaching data, the respective movement primitives and their sequencing is learned. For instance, the trajectory of the left hand in Fig. 3 (left) has the higher spatial variance of both hands and is therefore selected to be the guiding hand, which then serves as compact representation for the movement primitive. Details on learning the movement primitive based on the guiding hand’s trajectory are described in Appendix D. Learning of the task expansion and inverse kinematics proceeds as described in Appendix C and Appendix B.

In the exploitation phase, skills are performed by unrolling the movement representation to joint angles. Modulation signals shift, rotate and scale the skill in the robot’s workspace and control the speed of the motion.

B. Generalization Results

We now present generalization results in TABLE I for the three different architecture designs displayed in Fig. 2. The movement primitive learning is equal in all three designs. The task transformation is performed on the movement representation \( g \) through the homogeneous transformation \( H \) in designs (a) and (b) to obtain \( g' = Hg \). In case of late transformation according to design (c), the task space coordinates \( p_{(l,r)} \) of each end effector are transformed according to the same task transformation into \( p'_{(l,r)} = Hp_{(l,r)} \). In the holistic model of the task expansion and inverse kinematics (Fig. 2 (a)), the direct mapping from the movement representation \( g' \) to joint angles \( q_{(l,r)} \) of both arms is learned with an Extreme Learning Machine (see Appendix A).

We record three exemplary skills for evaluation comprising paddling on the left and right, as well as a weight lifting skill (see Fig. 3). In the test condition, the skill is displaced by 5cm along the y- and z-axis from the training area. Note that this requires considerable extrapolation from the learners. We evaluate end effector position errors in centimeters by applying the forward kinematics to the estimated joint angle trajectories. We further calculate the deviation of the learned skill from the tool constraints present in the training data, i.e. distance between end effectors (in centimeters) and end effector orientations (in degrees). Note also that the end effector orientations display considerable variance in the training data due to the slightly compliant stick we used to protect the robot hardware. In case of the early task transformation according to Fig. 2 (b), we only display the errors of the estimated task expansion without the subsequent kinematic mapping. To account for the random initialization of the learners, we repeat each learning step 100 times and present the average performances.

TABLE I displays training and test errors for the three architecture designs and data sets. The training errors show that the respective mappings between representations can be
learned very accurately. This includes the tool constraints like relative distance and orientation of end effectors. One can further detect a biased performance between end effectors with smaller errors for the guiding hand of the respective skill. The training performance is certainly sufficient for recall and reproduction of the taught skills in all three architectures.

However, the generalization errors in the test condition differ strongly with respect to the chosen architecture design: In case of the holistic task expansion and inverse kinematics (Fig. 2 (a)), we observe a moderate generalization performance. But if the task expansion is made explicit, generalization accuracy depends heavily on whether the task specification is transformed before or after task expansion.

In the first case (Fig. 2 (b)), the training data is not sufficient for a wide-ranged generalization of the task expansion (see large errors for the hand positions in TABLE I), which shows the difficulty of extracting tool constraints from few training samples. In case of a late task transformation applied to the expanded task specification of both hands (Fig. 2 (c)), generalization errors fall below the moderate errors of the holistic variant (Fig. 2 (a)). The improved generalization performance can be explained by the added prior knowledge about the tool constraint, i.e. homogeneous transformation of both end effector coordinates does not corrupt the tool constraint, and generalization is reduced to the subsequent inverse kinematics learner.

Fig. 3 illustrates the performance of the architecture configuration Fig. 2 (c) which achieves the best generalization results. The skills are generalized with considerable accuracy and the skill is successfully extrapolated to the new location in the robot’s workspace. The quantitative evaluation of the generalization capabilities between different architecture designs supports the qualitative argumentation in Sec. II: The explicit representation of tool/task constraints is advantageous for generalization in comparison to a holistic representation (Fig. 2 (a)), if task modulation is postponed as much as possible in order to offload complexity on to lower levels.

IV. MODULAR SKILL ARCHITECTURE

Finally, multiple skills are combined in an architecture (see Fig. 4), where the skill design depicted in Fig. 2 (c) with late spatial skill modulation is used to achieve optimal generalization from few kinesthetic demonstrations. The skill representations are enhanced with top-level sequencing and feedback, where selection of tasks, i.e. particular sequences of skills, translates to selecting one of the sequencer modules. Other than in [7], where sequencing is autonomous and learned with an HMM or in [11], where novel learned movement primitives are embedded in a fixed sequencer, the current interactive architecture relies on user feedback to select the sequence. The system, however, is reactive to proprioceptive feedback from the robot, which is directly used and fed back through all levels.

A. SEQUENCING AND BLENDING

We distinguish two cases of sequencing: (i) the sequencing of entire skills, and (ii) the sequencing of movement primitives using the same task expansion and kinematics.

In the first case, the entire stack of representations has to be exchanged which typically requires to move from one part of the robot’s workspace to another. For this purpose, we generate linear transition movements for both end effectors from the current configuration $p^{cur}_{l,r}(t)$ to the target configuration $p^{new}_{l,r}(t)$ of the next skill according to

$$ p_{l,r}(t) = p^{cur}_{l,r} + b(t)(p^{new}_{l,r} - p^{cur}_{l,r}) $$

$$ b(t) = 0.5 \tanh(6t/T - 3) + 0.5 \text{ for } t = 0, \ldots, T, $$

where $T$ specifies the movement duration. We blend between the respective inverse kinematic modules by setting

$$ q_{l,r}(t) = b(t)q^{cur}_{l,r}(t) + (1 - b(t))q^{new}_{l,r}(p_{l,r}(t)). $$

This open-loop blending between the execution of two skills is loosely inspired by human reaching movements which feature bell-shaped velocity profiles. Note that this is a hybrid approach that combines explicit modeling of the linear blending movement with the generalization from the learned inverse kinematics such that there is no need to explicitly specify a redundancy resolution for the blending.

The second case considers the execution of several, typically discrete, movement primitives that together define a skill but are deployed using the same chain of task and kinematic
Fig. 5. Skill transition from paddling left to paddling right: Blending results in a smooth trajectory which complies to the bi-manual tool constraint.

representations. This applies for instance to the weight lifting skill, which is decomposed into two movement primitives (compare Fig. 3 (right)). In this case, sequencer modules are implemented as state machines, where convergence of a discrete primitive, i.e. $|\dot{\mathbf{v}}| \approx 0$, triggers the transition to the next state. This simple state transition scheme together with the 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 blending between primitives can be implemented by switching primitives before they terminate [6]. The sequencing of two discrete primitives for the weight lifting skill is illustrated in Fig. 3 (right).

B. External and Proprioceptive Feedback

Fig. 4 further indicates how external feedback can be integrated into the architecture by providing modulation signals to the current sequencer module, which then pass modulation signals on to the skills. This could be visual feedback, input from a planner or user input, as it is used here. An essential part is proprioceptive feedback from the robot, which renders skill execution responsive to perturbations. To this aim, feedback of joint angles is propagated through the different representation levels by computing the respective inverse mappings. These could in principle also be learned, but are computed analytically for simplicity in the current setting. The analytic forward kinematics transforms joint angles to task space coordinates and reduces the task specification to the compact movement space by selecting the guiding hand position only.

V. Conclusion

We present and discuss representations for learning bi-manual skills from kinesthetic teaching and present a respective architecture for flexible sequencing of such skills. Thereby iCub can be taught to manipulate a stick with quite some versatility to switch between left and right hand paddling. To make the right choices on the architectural level is particularly important, because kinesthetic teaching of complex skills can naturally provide only few data in form of a handful of trajectories for each skill. The discussion and comparison of different approaches to model and generalize bi-manual skills shows subtle trade-offs between architectural design, generalization capabilities and according need for training data. While the discussion in this paper is restricted to still only a few variations and bi-manual skills, it is apparent that similar arguments apply also to other complex skills that require particular constraints to be respected by the redundancy resolution of full-body movements. The presented work shows that in order to lift learning to the architectural level, much care must be taken to make the right choices of representations and that it should be made more transparent what these choices are in different example architectures.

After deciding for a particular representational and learning scheme, the paper also shows that the architecture can successfully deal with few and very noisy data from kinesthetic teaching. Its robustness is achieved by using throughout a fast neural network learning scheme with strong extrapolation capabilities and a novel approach to stabilize dynamical system implementations of movement primitives by superposition of a contractive vector field without compromising the original movement shape. Future extensions need to investigate the inclusion of external, e.g. visual, feedback and more sophisticated sequencing schemes. Detailed analysis for other complex skills need to be provided in a similar manner to obtain architectures that can incorporate complex full-body skills.

APPENDIX

A. Extreme Learning Machine

We employ the fast, one-shot and feed-forward neural network learning scheme known as Extreme Learning Machine (ELM, [16], see Fig. 6) on all representational levels. In principle, more specialized machine learning approaches could be applied but we commit to a single learner implementation for simplicity on the one hand, but also because the ELM is a global learner with good extrapolation capability.

Fig. 6. Extreme Learning Machine (ELM).
ELMs combine a non-linear and high-dimensional random projection of inputs \( x \in \mathbb{R}^D \) into a hidden layer \( h \in \mathbb{R}^{H \gg D} \) with efficient linear regression learning of a perceptron-like read-out layer \( W^{\text{out}} \). The outputs \( y \) are calculated according to
\[
y = W^{\text{out}}h(x) = W^{\text{out}}\sigma(W^{\text{inp}}x),
\]
where \( \sigma(a_i) = (1 + \exp(-a_i - b_i))^{-1} \) are sigmoidal activation functions applied to the neural activities \( a_i = \sum_{j=1}^{D}(W^{\text{inp}})_{ij}x_j \) for all hidden neurons \( i = 1, \ldots, H \).

Learning is restricted to the read-out weights \( W^{\text{out}} \). The input weights \( W^{\text{inp}} \) and biases \( b_i \) are randomly initialized according to a uniform distribution in \([-1, 1]\) and remain fixed. Learning proceeds off-line by linear regression. For all training samples \( x(k) \) with \( k = 1, \ldots, K \), the hidden states \( h(x(k)) \) as well as the target outputs \( y(k) \) are collected row-wise in matrices \( H \) and \( Y \), respectively. Then, the optimal read-out weights are given by the (regularized) least squares solution
\[
W^{\text{out}} = (H^T H + \lambda I)^{-1}H^T Y,
\]
where \( I \) is the identity and \( \lambda > 0 \) a regularization parameter. We normalize inputs and outputs of the learner to the range \([-1, 1]\) according to the distribution of the training data. This yields to good learning results without further tuning parameters. We use \( H = 200 \) and \( \lambda = 0.5 \) if not stated otherwise. The rather strong regularization \( \lambda = 0.5 \) prevents over-fitting of the learner to the training data and improves the extrapolation outside the convex hull of the training data. Note that for the demanding generalization requirements, the global representation of inputs in the hidden layer is advantageous.

In the following, we describe the input and outputs to the learner for each representational level.

**B. Learning Inverse Kinematics**

Learning of direct inverse kinematics is applied in the bottom layer of the skill architecture. The recorded joint space data from the kinesthetic teaching is transformed via the analytic forward kinematics of iCub such that we obtain pairs of end effector coordinates \( p_{(l,r)} \) with corresponding joint angles \( q_{(l,r)} \) for both arms. Learning is conducted according to (1), where inputs \( p_{(l,r)} \) are 6D vectors with end effector coordinates of both hands and outputs are the 14D vectors \( q_{(l,r)} \) with the joint angles of both arms (compare Fig. 2 (b) and (c)). In case of learning the combined task and kinematics expansion in a single network (compare Fig. 2 (a)), inputs are the 3D positions of the guiding hand while the outputs remain to be the joint angles of both arms.

**C. Learning Task Expansions**

Task expansions map outputs of the movement primitives, for instance tool tip positions, to a complete task specification, e.g. end effector coordinates of both hands in case of bimanual skills. In the presented scenario, learning of task expansions is applied on the kinesthetic teaching data by calculating the analytic forward kinematics on the recorded joint angle trajectories yielding 6D vectors \( p_{(l,r)}(k) \), i.e. the end effector coordinates for both hands. The movement primitives are defined in the end effector space of the respective guiding hand \( g(k) \) which is determined by analyzing the variance in the end effector trajectories for both arms (compare Sec. III-A). The network is trained to expand the compact movement representation \( g(k) \) to the task specification \( p_{(l,r)}(k) \), \( \forall k = 1, \ldots, K \).

**D. Learning Movement Primitives**

Movement primitives (MPs) are "pieces of motion" \( g(k) \) with \( k = 1, \ldots, K \). In principle, any implementation of MPs can be considered in the proposed architecture. We commit to a dynamical system implementation which renders movement generation responsive to perturbations by integrating feedback. We propose a novel variant of dynamical movement primitives that modularizes the representation of the movement’s shape and the velocity characteristics along this shape. This approach has proven to cope well with noisy and very few data.

1) Movement Representation: We represent movements in terms of velocity fields, i.e. mappings \( v(g) \) from positions \( g \) in the movement primitive space to velocities \( v \), which are imprinted into ELMs with \( H = 200 \) and \( \lambda = 0.001 \). Training data comprises the guiding hand trajectory \( g(k) \) for \( k = 1, \ldots, K \), i.e. the inputs to the neural network. Target outputs for the learning are the corresponding velocities \( v(k) \). After training, movements are generated by integrating the velocities, i.e.
\[
g(t+1) = g(t) + \alpha v(g(t)),
\]
where $\alpha$ enables to modulate the speed of motion. The learned velocity field with exemplary flows for a paddle skill (projected into 2D for visualization purposes) is depicted in Fig. 7 (top left). We address the issue of spurious attractors by additionally learning a velocity field which implements a contraction to the desired movement shape (see Fig. 7 (top right)).

Stabilization of the movements is based on the idea to extract a dynamical system representation of the movement shape, that is the set of all points along the movement. The dynamics of this system are trained to contract to the movement shape: For any point $g^*$ on the movement shape and some initial condition $g(t)$ in the vicinity of the shape, we have

$$||g^* - g(t+1)|| < ||g^* - g(t)||.$$  

We imprint this contractive behavior into an ELM following the ideas from [24]: For each sample $g(k)$, we synthesize trajectories that start from random perturbations $g(k) + \epsilon$ and linearly contract to the original sample in a predefined number of steps (see [24] for details). We synthesize 10 trajectories for each sample with 5 steps each and imprint the contraction fields into ELMs with $H = 200$ and $\lambda = 10^{-6}$. The resulting contraction field for an exemplary trajectory is shown in Fig. 7 (top right).

3) Combining Movement and Contraction Fields: Robust movements are generated by superimposing velocity and contraction fields according to

$$v(g) = \alpha v(g) + \beta c(g),$$

where $\beta > 0$ weights the contribution of the contraction field. Very robust systems are obtained for $\beta \gg |\alpha|$ or $\beta \approx 1$, whereas decreasing $\beta$ results in slower contraction to the original movement after a perturbation.

A typical pitfall of superimposing two vector fields, one for the movement and one for improving stability, is to compromise the original movement features. Note, however, that the proposed approach leaves the original movement unaffected because $c(g) \approx 0$ for all $g$ in the vicinity of the training data (compare Fig. 7 (top right)).

The combined velocity field with exemplary flows depicted in Fig. 7 (bottom) confirms the stabilizing effect of the contraction field while the original movement shape is unaffected.

ACKNOWLEDGMENT

This research has received funding from the EC’s FP7/2007-2013 – Cognitive Systems, Interaction, Robotics – under grant agreement No 248311 - AMARSi.

REFERENCES