EU FP7

AMARSi

Adaptive Modular Architectures for Rich Motor Skills

ICT-248311

D 5.3

September 2012 (30 months)

Technical report on scaffolding learning by kinesthetic teaching and incremental learning

Authors: Seyed Mohammad Khansari-Zadeh, Seungsu KIM, Andre Lemme, Elmar A. Rückert

Due date of deliverable	1st September 2012
Actual submission date	15th October 2012
Lead Partner	TUG
Revision	Final
Dissemination level	Public

Abstract

In this deliverable we present three novel learning approaches and movement representations which can benefit from kinesthetic support and imitation learning. These different approaches were applied to motor skill learning tasks that can benefit from human demonstrations.

In particular EPFL-B has investigated how incremental learning can be used to adapt a learned forward model of a real robot arm to a new task. The model was initially learned via kinesthetic teaching using a dynamical system based approach. Furthermore EPFL-B has demonstrated which learning mechanisms are involved in teaching a robot how to catch fast inflight objects. Also this challenging task benefits from human demonstrations.

UniBi demonstrated how kinesthetic teaching facilitates motor skill learning in three different scenarios which emphasize that kinesthetic teaching transfers knowledge about tasks but also task constraints: first a inverse kinematics model of a KUKA light-weight robot is learned using recurrent neuronal networks. In a second experiment the iCub robot was taught via kinesthetic teaching to learn pointing movements without depth calculation of camera calibration. Finally, UniBi analysed the trade-offs in a learning and control architecture between representation and generalization on iCub which was used to learn and generalize multiple movement primitives that in turn were acquired via kinesthetic teaching.

At TUG an alternative movement primitive representation based on probabilistic inference in learned graphical models was developed. This approach has interesting and new features, i.e. it allows for an intuitive representation of complex motor skills based on learned cost functions. The cost function is in the most simple case specified by learned via-points. They allow for an easy integration of task specific prior knowledge. This prior knowledge might be observed from human demonstrations, which is currently investigated on a real robot in cooperation with UGent.

1 Motor skill learning using dynamical systems that benefit from human demonstrations

1.1 Incremental Learning

When modeling robot discrete motions with Dynamical Systems (DS), ensuring stability of the learned DS (from a set of demonstrations of the task) is a key requirement to provide a useful control policy. In our previous work [1], we presented an approach, called Stable Estimator of Dynamical Systems (SEDS), to learn the parameters of the DS to ensure that all motions closely follow the demonstrations while ultimately reaching and stopping at the target.

Despite the successful application of SEDS in many robot applications such as playing minigolf [2, 3], obstacle avoidance [4], catching flying objects [5], grasping [6], it is currently limited in that it only supports offline training. Incremental learning is often crucial to allow the user to refine the model in an interactive manner. Without incremental support, if one was to add new demonstrations after training the model, one would have to retrain entirely the model based on the combined set of old and new demonstrations. In this report, we provide preliminary results on extending our previous approach so as to support incremental learning without compromising the global asymptotic stability at the target. The new approach, called *SEDS-II*, which exploits the power of Locally Weighted Projection Regression (LWPR) [7, 8] to perform incremental learning. We evaluate our approach on the 7 degrees of freedom Barrett WAM arm.

1.1.1 Formalism

Consider a state variable $\boldsymbol{\xi} \in \mathbb{R}^d$ that can be used to *unambiguously* define discrete robot motions with a first order autonomous DS:

$$\dot{\boldsymbol{\xi}} = \boldsymbol{f}(\boldsymbol{\xi}) \qquad f : \mathbb{R}^d \mapsto \mathbb{R}^d \tag{1}$$

where $\boldsymbol{f}(\boldsymbol{\xi})$ is a continuous function. Given an initial point $\boldsymbol{\xi}^0$, the robot motion along time can be computed by integrating $\boldsymbol{f}(\boldsymbol{\xi})$ through time:

$$\boldsymbol{\xi}(t) = \int_0^t \boldsymbol{f}(\boldsymbol{\xi}) dt \tag{2}$$

Given a set of N demonstrations $\{\boldsymbol{\xi}^{t,n}, \dot{\boldsymbol{\xi}}^{t,n}\}_{t=0,n=1}^{T^n,N}$, an estimate of $\boldsymbol{f}(\boldsymbol{\xi})$ can be built using different statistical approaches such as LWPR, Gaussian

Process Regression (GPR) [9], Gaussian Mixture Regression (GMR) [10], or a combination of them (as we will show in this report). We assume that the function $\boldsymbol{f}(\boldsymbol{\xi})$ is continuous and differentiable. We are interested in determining a stabilizing command $\boldsymbol{u}(\boldsymbol{\xi}, \boldsymbol{f}(\boldsymbol{\xi})) \in \mathbb{R}^d$ such that the resulting DS:

$$\dot{\boldsymbol{\xi}} = \boldsymbol{F}(\boldsymbol{\xi}) = \boldsymbol{f}(\boldsymbol{\xi}) + \boldsymbol{u}(\boldsymbol{\xi}, \boldsymbol{f}(\boldsymbol{\xi})) \tag{3}$$

is 1) *globally asymptotically stable* at the target $\boldsymbol{\xi}^*$ to ensure the convergence of all trajectories to the target, and 2) an *accurate estimation of the user demonstrations* to satisfy the task requirements. Both of the above requirements are essential for $\boldsymbol{F}(\boldsymbol{\xi})$ to provide a useful control policy.

1.1.2 Stability

In this section, we present our approach on determining the stabilizing command $\boldsymbol{u}(\boldsymbol{\xi}, \boldsymbol{f}(\boldsymbol{\xi}))$ so as to ensure global asymptotic stability of $\boldsymbol{F}(\boldsymbol{\xi})$ at the target $\boldsymbol{\xi}^*$. To simplify the notation, we define:

$$\bar{\boldsymbol{\xi}} = \frac{\boldsymbol{\xi} - \boldsymbol{\xi}^*}{\|\boldsymbol{\xi} - \boldsymbol{\xi}^*\|} \qquad \forall \boldsymbol{\xi} \in \mathbb{R}^d \backslash \boldsymbol{\xi}^* \tag{4}$$

$$\alpha(\boldsymbol{\xi}, \boldsymbol{f}(\boldsymbol{\xi})) = \bar{\boldsymbol{\xi}}^T \boldsymbol{f}(\boldsymbol{\xi}) \qquad \forall \boldsymbol{\xi} \in \mathbb{R}^d \backslash \boldsymbol{\xi}^* \tag{5}$$

where $\bar{\boldsymbol{\xi}}$ is a unitary vector, and α is a measure of the misalignment between $\bar{\boldsymbol{\xi}}$ and the estimated function $\boldsymbol{f}(\boldsymbol{\xi})$. We define the stabilizing command $\boldsymbol{u}(\boldsymbol{\xi}, \boldsymbol{f}(\boldsymbol{\xi}))$, $\forall \boldsymbol{\xi} \in \mathbb{R}^d \backslash \boldsymbol{\xi}^*$ and $\forall \boldsymbol{f}(\boldsymbol{\xi}) \in \mathbb{R}^d \backslash \boldsymbol{\xi}^*$ according to:

$$\begin{aligned} \boldsymbol{u}(\boldsymbol{\xi}, \boldsymbol{f}(\boldsymbol{\xi})) = - \phi\big(\alpha(\boldsymbol{\xi}, \boldsymbol{f}(\boldsymbol{\xi}))\big) \Big(\alpha(\boldsymbol{\xi}, \boldsymbol{f}(\boldsymbol{\xi})) + \kappa_{\boldsymbol{\xi}} e^{-\sigma_{\boldsymbol{\xi}} \|\boldsymbol{\xi}\|} \ \dots \\ + \kappa_{\dot{\boldsymbol{\xi}}} (1 - e^{-\sigma_{\boldsymbol{\xi}} \|\boldsymbol{\xi}\|}) e^{-\sigma_{\dot{\boldsymbol{\xi}}} \|\boldsymbol{f}(\boldsymbol{\xi})\|} \Big) \bar{\boldsymbol{\xi}} \end{aligned} \tag{6}$$

to guarantee the global asymptotic stability of the DS $\boldsymbol{F}(\boldsymbol{\xi})$. In Eq. 6, the parameters $\kappa_{\boldsymbol{\xi}}$, $\kappa_{\dot{\boldsymbol{\xi}}}$, $\sigma_{\boldsymbol{\xi}}$, $\sigma_{\dot{\boldsymbol{\xi}}}$ are positive scalars, and $\phi(\alpha) \in \mathbb{R}$ is a smooth activation function that is defined by:

$$\phi(\alpha) = \begin{cases} 1 & 0 < \alpha \\ 0.5\Big(sin\big(\tau\alpha) + \pi/2\big) + 1\Big) & -\frac{\pi}{\tau} \leq \alpha \leq 0 \\ 0 & \alpha < -\frac{\pi}{\tau} \end{cases} \tag{7}$$

where $\tau > 0$ is a scalar to tune the slope of the activation function (see Fig. 1). Note that for the clarity of the formulation, we have denoted

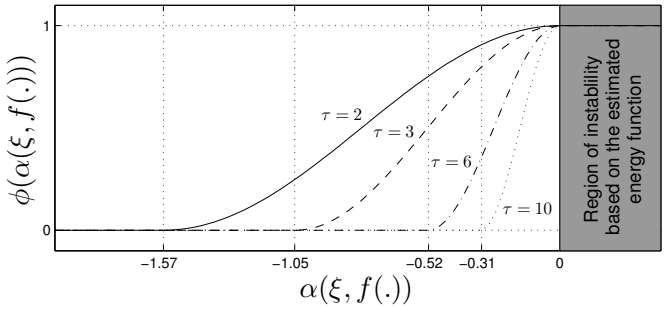


Fig. 1: Tuning the width of the stability margin with the parameter τ. The lower the τ, the larger the width of the stability margin.

$\alpha(\boldsymbol{\xi}, \boldsymbol{f}(\boldsymbol{\xi}))$ with α in Eq. 7. The activation function is used to trigger the stabilizing command when the system is at the edge of instability. Note that for $\alpha(\boldsymbol{\xi}, \boldsymbol{f}(\boldsymbol{\xi})) > 0$, the stability of $\boldsymbol{f}(\boldsymbol{\xi})$ cannot be ensured according to the Lyapunov stability theorem. In these situations $\phi(\alpha) = 1$, and the stabilizing command can be used to ensure the stability of the DS. For $-\pi/\tau \leq \alpha(\boldsymbol{\xi}, \boldsymbol{f}(\boldsymbol{\xi})) \leq 0$, the activation function smoothly rises from 0 to 1. In this region, although $\boldsymbol{f}(\boldsymbol{\xi})$ is stable, it is still slightly modified with the stabilizing command to ensure the continuity of $\boldsymbol{F}(\boldsymbol{\xi})$. For $\alpha(\boldsymbol{\xi}, \boldsymbol{f}(\boldsymbol{\xi})) < \pi/\tau$, the system has a safe stability margin (according to the user preference), hence no stabilizing command is generated, i.e. $\boldsymbol{u}(\boldsymbol{\xi}, \boldsymbol{f}(\boldsymbol{\xi})) = 0$.

Fig. 1 illustrates the effect of τ on the activation function. By decreasing τ, the width of the activation region increases. There is a compromise inherent in setting the value of τ. By lowering τ, we could avoid a sudden change in the direction of motion. On the other hand, by setting a high τ, we could keep the larger part of the stable region intact.

1.1.3 Experiments

We evaluate the performance of the proposed approach in a robot experiment performed on the 7-DOF Barrett WAM arm. In this experiment, we demonstrate that our approach allows combining two different regression techniques in order to benefit from the advantages of both. The robot experiment consisted of having the WAM arm place an orange on a plate and into a bucket. First, the placing task on the plate is shown to the robot seven times via kinesthetic teaching (see Fig. 2a). A DS estimate of this motion is constructed using GMM with 7 Gaussian functions.

The demonstrations and the reproduction of the task from the proposed method are illustrated in Fig. 2b. As is illustrated, the reproductions closely follow the demonstrations, while their global stability is ensured. Though this

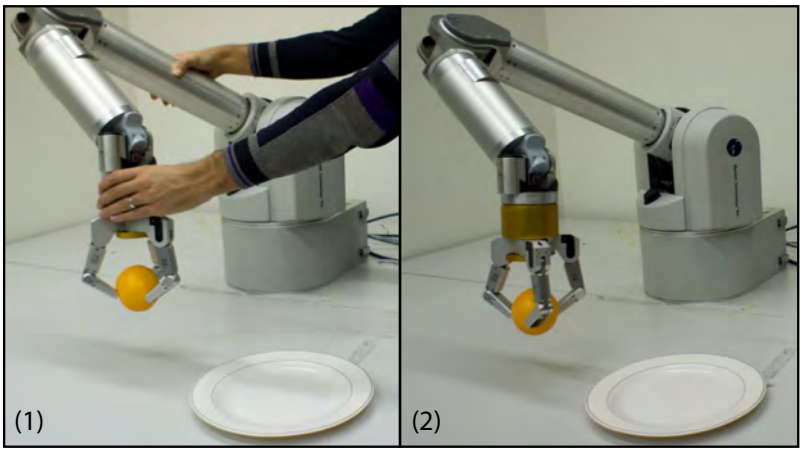
model can successfully generate motions to place the orange on the plate, it cannot be used with the bucket (see the solid black lines in Fig. 3a). In order to adapt the robot motions to this change while avoiding having to retrain the whole model, we exploit the incremental learning power of LWPR to locally modify the DS given by GMR:

$$\dot{\boldsymbol{\xi}} = \boldsymbol{F}(\boldsymbol{\xi}) = \underbrace{\boldsymbol{g}(\boldsymbol{\xi}) + \boldsymbol{l}(\boldsymbol{\xi})}_{\boldsymbol{f}(\boldsymbol{\xi})} + \boldsymbol{u}(\boldsymbol{\xi}, \boldsymbol{f}(\boldsymbol{\xi})) \tag{8}$$

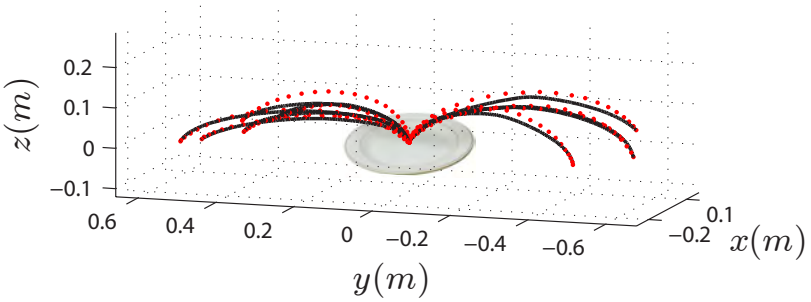
where $\boldsymbol{g}(\boldsymbol{\xi})$ and $\boldsymbol{l}(\boldsymbol{\xi})$ correspond to the DS that are modeled with GMR and LWPR, respectively. Initially $\boldsymbol{l}(\boldsymbol{\xi}) = 0$, $\forall \boldsymbol{\xi} \in \mathbb{R}^d$. The LWPR model is trained incrementally and online by interactively correcting the robot movement while it approaches the bucket (see Fig. 3b). The blue hollow circles in Fig. 3a shows the new training data points that were collected interactively as we have explained. Figure 3c illustrates the reproductions from the combined DS according to Eq. 8. With the new model, the robot can successfully adapt its motion and place the orange into the bucket. Note that in this experiment, the GMR model grants the base behavior for the placing task, and the LWPR model provides the required adaptation to the environment. Anytime when it is necessary, the base behavior can be retrieved by canceling out the LWPR term in Eq. 8. By extension, one can also imagine having several LWPR models, each of which provide the required adaptive behavior for different containers.

1.1.4 Conclusion

In this section, we presented a novel approach to ensure the global asymptotic stability of DS that are estimated from a set of demonstrations. Compared to our previous approach [1], this work allows using incremental learning to refine the behavior of the DS model in an interactive manner. To ensure global asymptotic stability of the system at the target, our approaches generates the stabilizing command online to correct the motion when it shows unstable behavior. At its current form, the stabilizing command is generated from a hard-coded formula to ensure stability at the target. The generated stabilizing command may not be optimal from the perspective of the desired behavior that is defined by the user demonstrations. We are currently investigating possible means to build an estimate of the stabilizing command from the user demonstrations (i.e. the same demonstrations as those that are used in estimating $\boldsymbol{f}(\boldsymbol{\xi})$). This is essential as it would allow to reduce the distortion due to applying the stabilizing command, and thus to generate motions that are more similar to the user demonstrations.

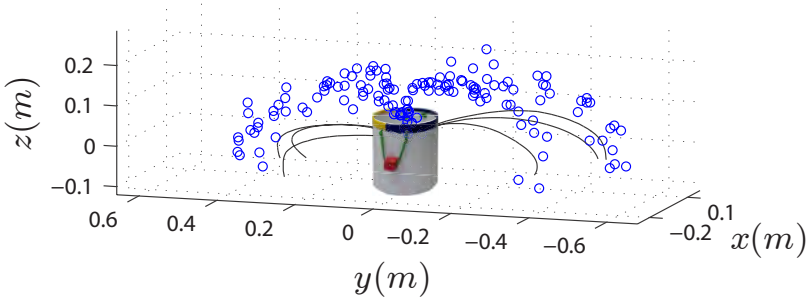


(a) The demonstrations of the task through kinesthetic teaching (left), and its reproductions by the robot (right).

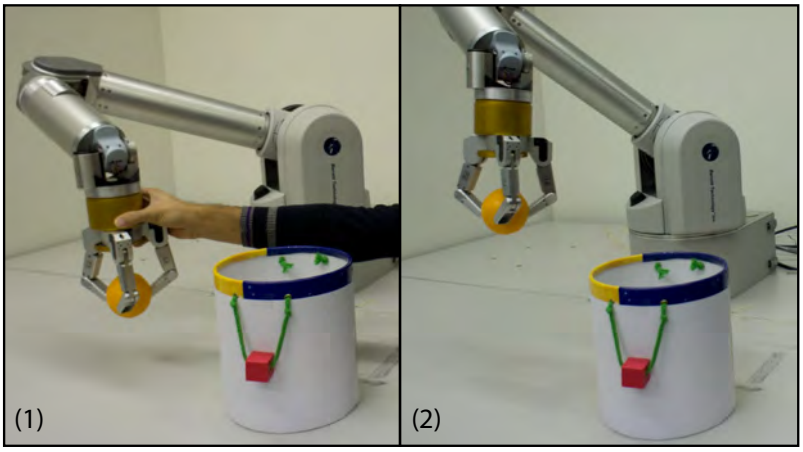


(b) Generation of trajectories from different initial points

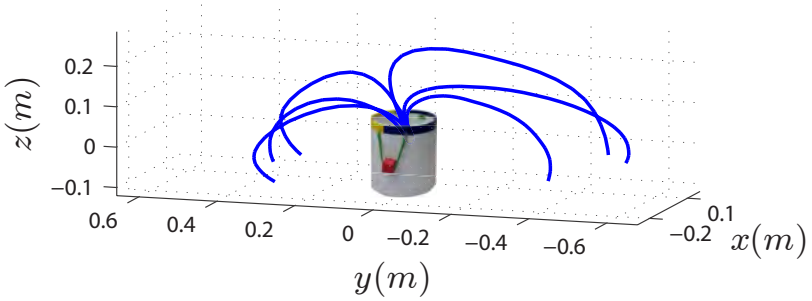
Fig. 2: The robot experiment of placing an orange on a plate. The placing motion is modeled with GMR.



(a) By replacing the plate with a bucket, the previous model can no longer be used. To adapt to the new situation, the user interactively modifies the robot trajectory as the robot approaches the bucket. These data are used to incrementally train the LWPR DS. The new training data points for the LWPR DS are shown with blue hollow circle. The solid black lines represents trajectories generated with solely using the GMR model.



(b) Training interactively the DS model in order to adapt to the new situation (left). The robot reproduction with the combined LWPR+GMR DS (Right).



(c) Trajectories generated with the combined GMR+LWPR DS. The generated trajectories can successfully place the orange into the bucket.

Fig. 3: Adaptation of the DS for the case where the plate is replaced with the bucket.

1.2 Catching an object in-flight

We combined different stages of learning to teach a robot how to catch in-flight fast moving targets. These stages include: (1) learning how to predict accurately the trajectories of fast moving objects; (2) learning how to determine the mid-flight catching configuration (intercept point) and (3) learning motion of arm-and-hand to enable fast planning of precise trajectories for the robots arm to intercept and catch the object on time. The schematic overview and control flow of the task are shown in Fig. 4 and 5.

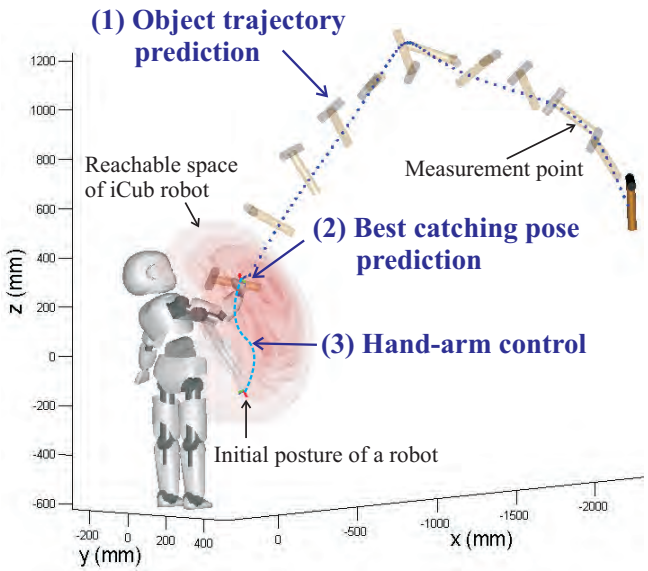


Fig. 4: Schematic overview of the task

1.2.1 Learning the dynamics of a moving object

To predict the trajectory of a flying object, we take a dynamical system based modeling approach. In its most generic form, the dynamics of a free-flying object follows a 2^{nd}-order autonomous dynamical system:

$$\ddot{\xi} = f\left(\xi, \dot{\xi}\right) \tag{9}$$

where, $\xi \in \mathbb{R}^D$ denotes the state of the object (position and orientation vector of the point of interest attached with the object). $\dot{\xi} \in \mathbb{R}^D$ and $\ddot{\xi} \in \mathbb{R}^D$ denote the first and second derivatives of ξ.

We consider complex objects that have non-diagonal inertia matrices and where the point of interest (the grasping point) is not located at the center

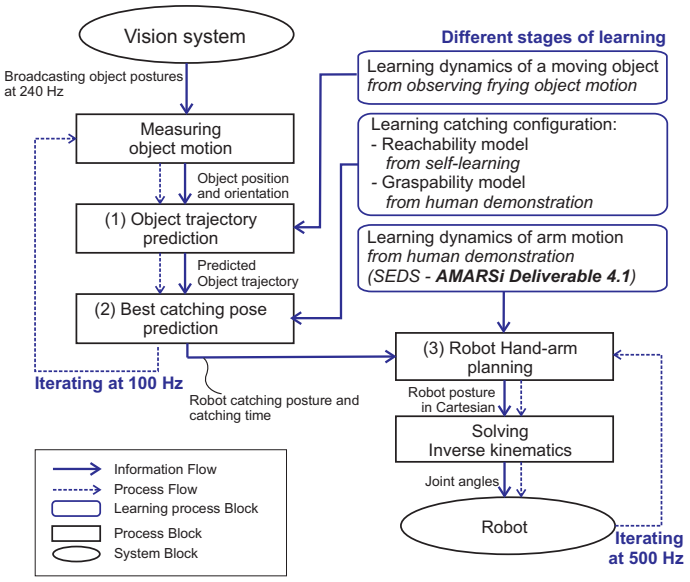


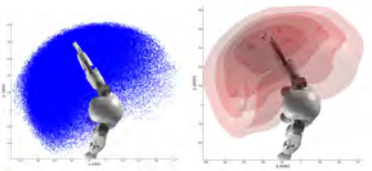
Fig. 5: Control flow of the task and different stages of learning

of mass. Using several examples, in which a demonstrator threw five objects (ball, half empty bottle and empty bottle, hammer and tennis racket), varying the initial translational and rotation speed, we contrasted five state-of-the-art non-linear regression techniques to best estimate this complex non-linear dynamics $f(.)$ [11]. Encoding the demonstrations using a Dynamical System (DS) provides an efficient way to model the dynamics of a moving object solely by observing examples of the objects motion in space. We could further estimate the whole trajectory of the object by integrating the DS in time. To enable real-time tracking, the estimated model of the objects dynamics is coupled with an Extended Kalman Filter for robustness against noisy sensing. A complete description of the method with a detailed comparison across different techniques for the estimation is available at [11].

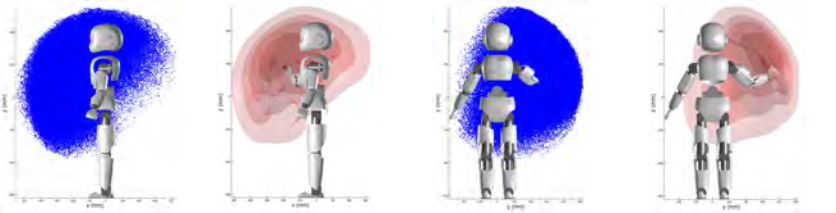
1.2.2 Predicting catching configuration

To predict the mid-flight catching configuration, we developed a data driven probabilistic model of the robots reachable-space of a robot and of the grasping posture (position and orientation of hand with respect to an object). The reachable space is learned through motor babbling, whereas the grasping region on the object is learned from demonstration of grasping on static object. The learned reachable space of iCub humanoid robot and the graspable space for an hammer are shown at Fig. 6 and 7 respectively. By combining the likelihood yielded by the learned workspace model and the model of the

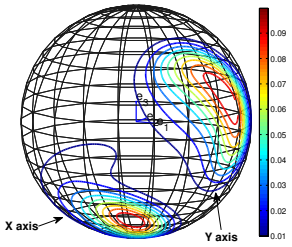
grasping region onto the object, the robot can determine whether the object is catchable or not, as well as determine the best grasping posture without path planning and solving explicitly the inverse kinematics.



(a) samples x-y (b) model x-y

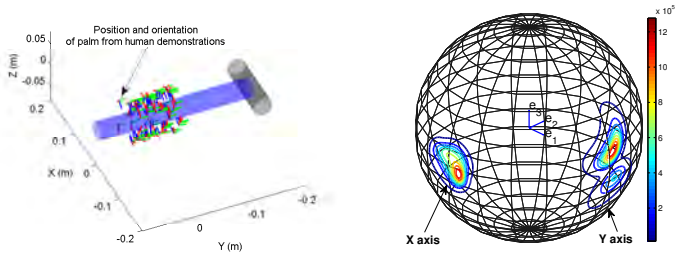


(c) samples x-z (d) model x-z (e) samples y-z (f) model y-z



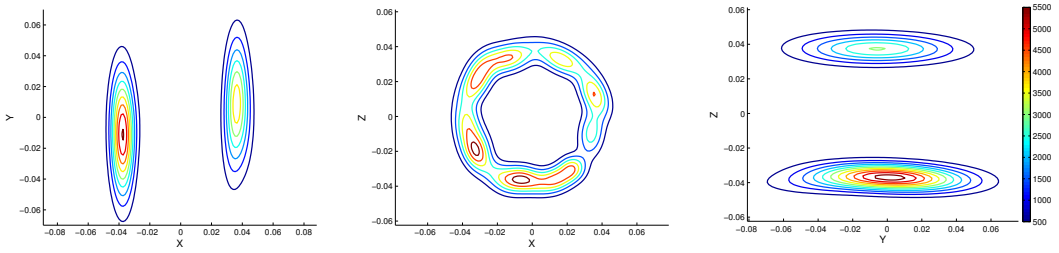
(g) orientation

Fig. 6: Modeling of reachable space for right arm of iCub humanoid robot. (a,c and e) show the reachable 3D cartesian points which are demonstrated from the uniform distribution in joint space. (b,d and f) show the probability contour of the reachable space model which is trained through Gaussian Mixture Model with 10 Gaussians. (g) shows the orientation contour when the iCub's end-effector position is $[-0.2,\ 0.2,\ 0.45]$



(a) A hammer

(b) x and y directional contour of orientation



(c) x-y model at z=0.0

(d) x-z model at y=0.0

(e) y-z model at x=0.0

Fig. 7: Modeling graspable space of a hammer for iCub hand: The graspable space is modeled using GMM with 10 Gaussians. (c,d and e) show Likelihood contour for end-effector position; (b) shows x and y directional likelihood contour of orientation with the fixed position [0.0, 0.0, 0.035].

1.2.3 Hand-Arm Control

We use the CDS model [6] for controlling the arm and fingers in coordination. This approach comprises of learning two different dynamical systems for reaching (arm) and grasping (fingers) trained through SEDS [1] and one inference model that learns a task metric for hand-arm coupling. The two dynamics models are then coupled during task execution using the inference model. In this implementation, we implement the coupling using the metric of *distance-to-target*. Such a spatial coupling between hand and fingers ensures timely closure of the fingers while following the learned dynamics as closely as possible, even in the presence of arbitrary perturbations. In this framework, we use the timed DS controller of [5] for the reaching motion in order to intercept the flying object at the desired instant. It is to be noted that no change in the other member models of CDS is needed to be compatible with the new reaching dynamics. For more details the reader is referred to [6].

The predicted catching posture calculated by the above best catching prediction module is fed as target to the position and orientation dynamical systems. Although this predicted catching configuration is updated at every control cycle, the end-effector and finger joint trajectories generated by our model remain smooth and feasible. The robot continuously adapts the hand and arm motion as the prediction of the final catching posture improve over time. Output of the position and orientation DS is converted into the joint state of a robot using the damped least squares inverse kinematics.

We evaluate the performance of the proposed system using two sets of experiments, iCub humanoid robot simulator and the KUKA LBR 4+ platform; see the results in Fig. 8 and 9.

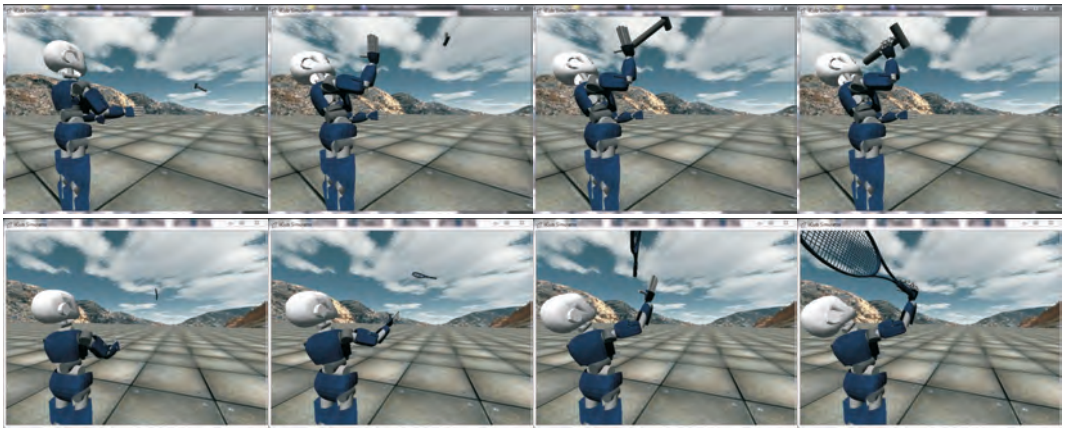


Fig. 8: An hammer and a racket is thrown 3 miter distance from the robot in the iCub simulator. The simulated iCub robot catches the hammer and the racket in the fly.

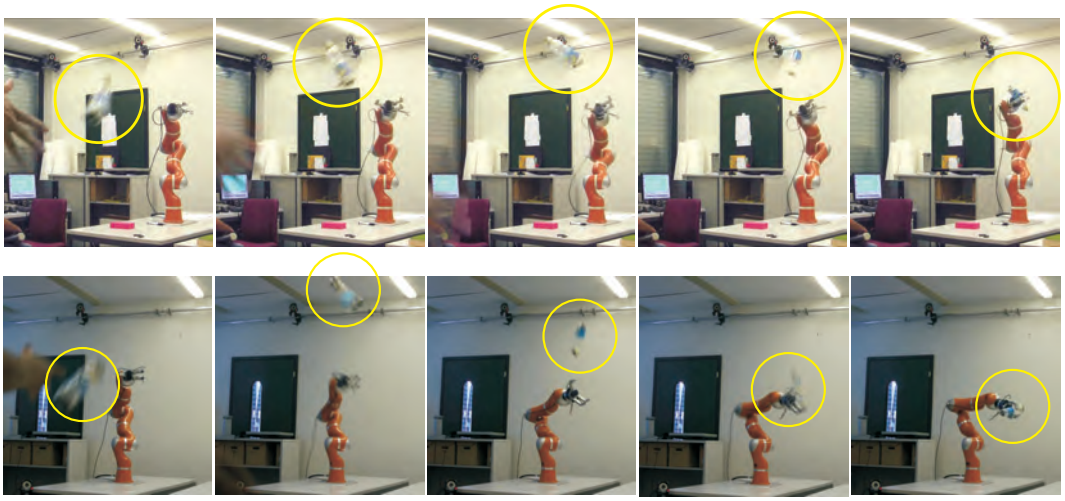


Fig. 9: A thrower throws a bottle 3 miter distance from the KUKA-KWR robot. The robot catches a bottle in the fly. The full video is available in the AMARSi YouTube site, http://www.youtube.com/amarsiproject

2 Scaffolding learning using kinesthetic support and imitation

Currently, kinesthetic teaching is mostly used for rapid initialization of a representation in task space like for instance in section one above. In this function, kinesthetic teaching has been considered as one of the possible interfaces for learning from observation, for instance by substituting visual observation. Much less, however, it has been considered that in particular for redundant robots, the redundancy resolution in joint space provides additional degrees of freedom and in the presence of complex mechanisms, bi-manuals skills, or complex environments actually a particular redundancy resolution encodes information about the task constraints in addition to the actual end-effector task. This information can also be very efficiently transfered to the robot by kinesthetic teaching because the human often intuitively knows how to use the redundancy in the robot reasonably. Thereby, additional modeling efforts for task constraints can be avoided, e.g. an proceeding 3D scene analysis, or the optimal way of avoiding self-collisions.

In the following we discuss the use of kinesthetic teaching for different tasks and consider in particular the representations that need to be derived by the learning process. First, we will start to describe a scenario in which we learn the inverse kinematics for the Kuka LWR [12]. Second, we show a similar scenario in which we added vision to perceive objects and learn a pointing behavior (contribution to Exp. scenario E3). And at last we show a complex movement scenario in which we can learn complex skills which also include tool use. This also contribute to architecture w.r. to WP6.

2.1 Learning inverse kinematics

One major goal of current robotics research is to enable robots to become co-workers that collaborate with humans efficiently and adapt to changing environments or work flows. In [13] we present an approach utilizing the physical interaction capabilities of compliant robots with data-driven and model-free learning in a coherent system in order to make fast reconfiguration of redundant robots feasible. Users with no particular robotics knowledge can perform this task in physical interaction with the compliant robot, for example to reconfigure a work cell due to changes in the environment. For fast and efficient learning of the respective null-space constraints, a reservoir neural network is employed. It is embedded in the motion controller of the system, hence allowing for execution of arbitrary motions in task space. We describe the training, exploration and the control architecture of the systems

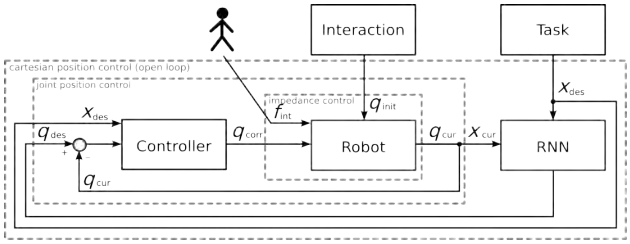


Fig. 10: Block diagram illustrating the control scheme with its three control loops. Physical interaction between the human operator and the robot is modeled as interaction force f_{int}

on the Kuka Light-Weight Robot (LWR).

This work is an example of how to integrate the kinesthetic teaching in a control scheme such that the information can be passed down to the learning algorithms. This control scheme is also similar to the ones used in the work described in the next sections.

In order to explain the control scheme of this work, including the use of compliance features of the robot for kinesthetic teaching and position impedance control with the learned recurrent neural network (RNN) controller, we provide a systems engineering perspective to the system. Fig. 10 depicts the major functional blocks and their control flow. In its two modes, exploration and execution, the system selectively activates different control strategies as explained in the following.

Exploration: In a first phase of exploration the robot is in its *gravitation compensation* mode, where forces applied to the robot are not counteracted by the controller. This phase is used to position the robot manually to a desired joint angle configuration q_{init}. After the configuration is reached, the robot is switched to impedance control for recording training data via kinesthetic teaching. In this phase the Cartesian position control and joint position control is inactive. The only active control loop in this phase is the inner impedance control loop of the LWR. Reference for this control loop is the interaction force f_{int} applied by the operator and the chosen joint angle configuration q_{init}. Parameters for the impedance controller are stiffness and damping in Cartesian coordinates, which were carefully chosen to allow easy physical interaction with the robot.

During kinesthetic teaching the current joint angles q_{cur} and the corresponding Cartesian end-effector positions x_{cur} are recorded and after successful training passed as training data to the RNN to enable the network to learn the taught constraints. Once learning is finished, the trained network is embedded in the hybrid control scheme explained in the following.

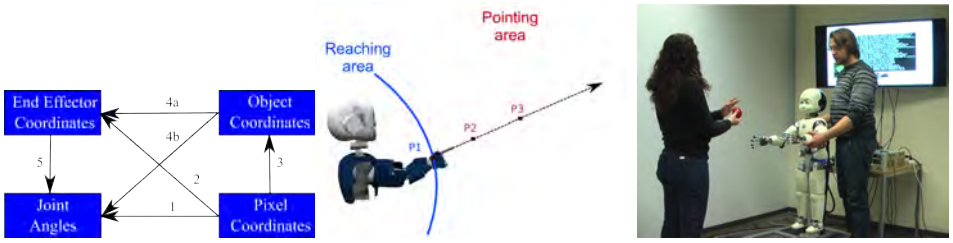
Execution: During the execution phase a 3D task-space trajectory x_{des} is provided by the user and the position controllers are active. The end-effector positions x_{des} are passed to the RNN to map them to desired joint values q_{des}. Note that the the RNN that was trained in the exploration phase, now serves as open-loop controller constraining the redundancy resolution.

Our results show that the learned model solves the redundancy resolution problem under the given constraints with sufficient accuracy and generalizes to generate valid joint-space trajectories even in untrained areas of the workspace. Note that the viewpoint on kinesthetic teaching here is unconventional: we do not teach the actual task in form of end-effector trajectories, but rather teach examples of redundancy resolution that respect the particular constraints for the complex environment and avoid collisions with permanent obstacles in the workspace. This type of encoding of the task constraints by learning first can be combined with the task space learning, e.g. as provided in section one or with other methods developed previously in WP4, and dynamic obstacle avoidance. It replaces an explicit modeling of the 3D-scene and the definition of a respective computational scheme for redundancy resolution that respects the constraints by a simple teaching procedure that transfers the implicit human knowledge on how to deal with a complex environment to the robot.

2.2 Learning redundancy resolution for pointing without depth calculation

Pointing at and gesturing towards something refer to orienting a hand, arm, head or body in the direction of an object and constitute basic communicative abilities for cognitive agents (e.g. humanoid robots). The goal of this work is to show that both approximate pointing by a hand gesture and exact pointing, can be learned as a direct mapping from the object's pixel coordinates in the visual field to hand positions or to joint angles respectively. Again, the goal is to omit a number of modeling steps that would usually be required for this task. We do not compute a 3D position of the object, do not need any camera calibration, and again let the user define the redundancy resolution, this time of the iCub robot's arm. In [14, 15], different neural network paradigms (multilayer perceptron, extreme learning machine and reservoir computing) were explored, to learn pointing and gesturing based real world data gathered on the humanoid robot iCub. Training data are interactively generated and, for exact pointing, recorded from kinesthetic teaching.

Despite the seemingly simple character of the everyday gesturing and pointing, there are substantial differences with respect to how the corre-



(a) (b) (c)

Fig. 11: (a) Illustration of the different mappings possible to solve pointing and gesturing tasks. The numbers are indexing the concrete mapping referred to in the text.(b) Illustration of pointing task characteristics and (b) a learning configuration of kinesthetic teaching for pointing gestures.

sponding mapping is modeled, represented and computed. Fig. 11a illustrates different options, which lead to different approaches. These are given by concatenating mappings along the different paths starting from the lower right box and ending at the lower left box. The lower right box represents the intrinsic visual coordinates given in pixel coordinates for the left and right camera image (i_L, j_L, i_R, j_R), which are provided as input for the different approaches. We refer all mappings, which do not map directly from pixel coordinates to joint angles as non-direct, because to control the robot we would need joint angels, which we get only with additional mappings. We first identify *Mappings 1* and *2*, as follows.

Mapping 1 - It is defined as a direct mapping from pixels to joint angles $(\theta_1, ..., \theta_7)$ for fully controlled positioning of the arm, including orientation for exact pointing.

Mapping 2 - In this case, we have a mapping from pixels to 3D hand positions in task space (x_e, y_e, z_e) for gesturing.

There is a difference in complexity between *Mappings 1* and *2*. The former needs more information and must handle more precisely the redundancies in the arm to provide the correct orientation of the forearm pointing to the object. The latter just assumes that the hand is positioned at the point of the robot egosphere, which is closest to the object. That is, the projection of the 3D-object position along the pointing ray onto the egosphere of reachable positions is the learning target (cf. Fig. 11b), which can be reached with different redundancy resolutions of the arm and therefore different exact pointing directions.

Thus, for learning the *Mapping 2*, less information needs to be provided. However, exact pointing cannot be expected, since the standard inverse kinematics (*Mapping 5*), which is used to provide the joint angles for positioning of the hand, does not possess previous knowledge about the pointing requirement and uses other criteria for resolving the redundancy. Learning the *Mapping 2* from data will constitute the first learning scenario for gesturing.

For the sake of comparison, *Mappings 4a* and *4b* are then defined as follows.

Mapping 4a - It is defined as a mapping from the object position in space to end-effector coordinates.

Mapping 4b - In this case, we have a mapping from the object position in space to joint angles $(\theta_1, ..., \theta_7)$.

It is worth emphasizing the differences between the proposed approaches (i.e. *Mappings 1* and *2*) and the standard approaches available in the literature (i.e. *Mappings 4a* and *4b*). Learning the *Mappings 4a* and *4b* would first require the computation of 3D-object positions (x_b, y_b, z_b) as defined by the *Mapping 3* and then consider pointing as a kind of reaching towards these coordinates through *Mappings 4a* and *4b*. This is feasible only if the 3D-coordinates of the objects can be inferred from the camera images, which requires stereo matching and depth calculation. As compared to the direct approach, the depth calculation provides useful but dispensable information for pointing and is well known to be difficult, because the stereo-matching problem is ill-posed.

In this study different mappings to learn pointing on the humanoid robot iCub have been explored. In the first experiments, a simulated environment is used to generate training data and tested three different data-driven and model-free learning approaches on this dataset. From all networks architectures tested in this study, the Static Reservoir Computing and Extreme Learning Machine, refined with intrinsic plasticity learning, have shown very good performances. As a result gesturing behavior is already possible to implement in this setup. An additional result is that the learning of a direct mapping, from pixel coordinates to arm joint values, for pointing without depth calculation is possible, if respective visuo-motor training data is available.

In the real world experiments a human tutor was teaching iCub how to point, by using kinesthetic teaching to gather training data intrinsically containing constraints given by the robots body structure. The results emphasizes the effectiveness of the kinesthetic teaching. The tutor effortless finds a very good solution to the more complex redundancy resolution to

point at the ball in the first layer, even without any deliberation or explicit knowledge that this part of the task differs from the others. No explicit modeling of the robot's body or the task constraint is therefore necessary, provided the training data is sufficient for all different conditions. Thereby a very effective teaching takes place, from the tutor to the robot. This conveys the task constraint given in form of different redundancy resolutions along with the task itself.

The main advantage of this learning setup is that adaptation to changes in the task, the robot's morphology (e.g. use a stick to point), or the environment can be achieved by quickly relearn the mapping, which takes only few minutes. This is possible, because the data driven learning prevents explicit modeling and calibration of the camera and the robot and rather enables an optimal knowledge transfer from the human to the robot via kinesthetic teaching. Parts of these results are published in [14] and under revision in [15].

2.3 Learning Skills and tool-use

The study presented in [16] on a modular architecture for bi-manual skill acquisition from kinesthetic teaching takes the discussed work on step further in combining architecture, kinesthetic teaching and stable movement primitives [1]. 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. 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.

In [16], 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 movement primitives from demonstration 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 $\mathbf{g}$ 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

[1] Implications on the architecture for rich motor skills are discussed further in D6.2.

Fig. 12: Experimental setup of the physical human-robot interaction: A human tutor teaches iCub a bi-manual skill without tool [left] and with [right].

highest spatial variance during the motion.

Task Expansion: Expansion of the current target $\mathbf{g}$ from the compact movement space, e.g. representation of a end effector coordinates for one arm, to an explicit task formulation, e.g. end effector coordinates $\mathbf{p}_{\{l,r\}}$ 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 for speed and for spatial transformation of task space variables $\mathbf{g}$ or $\mathbf{p}$ by means of homogeneous transformation.

Redundancy Resolution: Mapping from task space specification $\mathbf{p}_{\{l,r\}}$ to joint angles $\mathbf{q}_{\{l,r\}}$ using a particular redundancy resolution scheme.

One particular feature in this representation architecture is the learning of task expansions, which 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.

Multiple skills are combined in an architecture (see Fig. 13), where the skill design with late spatial skill modulation is used to achieve optimal generalization from few kinesthetic demonstrations. The skill representations are enhanced with top-level sequencing, where selection of tasks, i.e. particular sequences of skills, translates to selecting one of the sequencer modules (see top row Fig. 13 "Sequencer Layer"). Fig. 13 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.

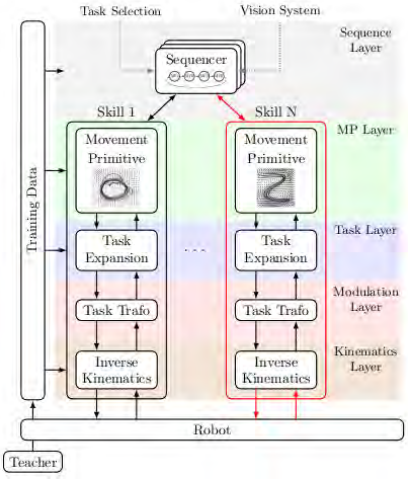


Fig. 13: Modular skill architecture comprising a library of skills and top-level sequencer modules. The teacher provides training data for learning on each representational level by means of kinesthetic teaching. Proprioceptive feedback is propagated bottom-up and enters the movement primitive representation. Visual feedback modulates skill execution in a top-down fashion.

We implemented different phases for teaching, learning, in which we build up the different mappings for the needed skill and a execution phase where we perform the skill.

In the *exploration phase*, a human tutor physically guides the robot in order to generate training data. This kinesthetic teaching 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. 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. 12).

In the *learning phase*, which is following each demonstration of the tutor, the recorded joint angle trajectories $\mathbf{q}_{\{l,r\}}(k)$ for $k = 1, \ldots, K$ of both arms are augmented with the respective end effector coordinates $\mathbf{p}_{\{l,r\}}(k)$ by calculating the forward kinematics. An automated data analysis is conducted to determine the task space coordinates $\mathbf{g}$ 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. Based on the teaching data, the respective movement primitives and their sequencing is learned.

In the *execution 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.

Also in this work, the kinesthetic teaching is used to provide task constraints via the particular redundancy resolution which is needed for the bi-manual manipulation. Additionally and in the same demonstration, the task space trajectories are provided as well. Other than in the first scenario on the LWR, where task constraints were separated from the the actual task and learned separately through a network devoted to redundancy resolution only, here it would in principle be possible to directly learn one holistic mapping comprising both, the task and the task constraints. Kinesthetic teaching can provide the required information, however, the internal representation will differ and therefore the demands on quality and size of training data as well. It shall be noted in this context that kinesthetic teaching data necessarily is sparse. Only a few demonstrations will be feasible on a complex robot and therefore some bias needs to be added in the control architecture to allow for strong generalization. Comparative evaluation of different architectures in [16] shows that a separation of redundancy resolution providing the task constraint and the task and its task expansion is useful to optimally exploit the sparse data obtained from kinesthetic teaching.

2.4 Discussion

The presented scenarios shed more light on the role of kinesthetic teaching and the associated knowledge transfer from the user to the robotic system. Going beyond the usual understanding of kinesthetic teaching as mere interface for demonstration of task space movements, we show that kinesthetic teaching can in particular be used to learn task constraints, which are implicitly encoded in particular redundancy resolution schemes. Whereas this is not necessary if only free movements and simple environments are considered. The examples of movement constraint environment, of self-collision avoidance, and of bi-manual skills all show that explicit modeling steps for encoding this kind of task constraints in redundancy resolution can be avoided by kinesthetic teaching. This is highly desirable from a practical point of view. While explicit schemes are in principle feasible, they always required explicit modeling, additional external sensing and a skilled programmer to implement or change a particular redundancy resolution approach. This is time-consuming and costly and prevent teaching of advanced tasks by naive users, who however at least on humanoid robots easily and intuitively know how to guide an arm even in a complex task. It turns out that even a separate learning only of the task constraints can be useful, for which the only reasonable interface is kinesthetic teaching. In summary, by using kinesthetic

teaching prior knowledge of the task and explicit modeling steps can be reduced because the intelligence of the tutor is used to provides "just right" data for the learning architecture respecting task constraints. This enables a fast reconfiguration and teaching capability for every-day users.

3 Principles for an alternative movement representation

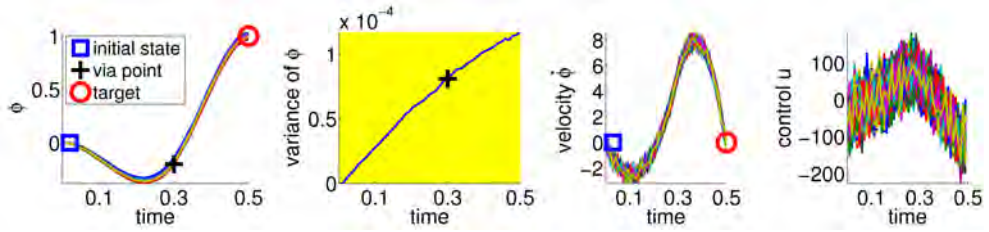
In this section we describe an alternative movement primitive representation based on probabilistic inference in learned graphical models. The presented approach has interesting and new features, i.e. it allows for an easy integration of task specific prior knowledge. This prior knowledge might be observed from human demonstrations in the form of relevant via-points or desired energy states of a dynamical system. However, in this work we only present the basic characteristics of this promising approach for motor skill learning. A detailed description of the approach can be found at the AMARSi project website[2]. The benefit of human demonstrations for motor skill learning is currently investigated on a real robot in collaboration with UGent.

3.1 Learned Graphical Models for Probabilistic Planning Provide a New Class of Movement Primitives

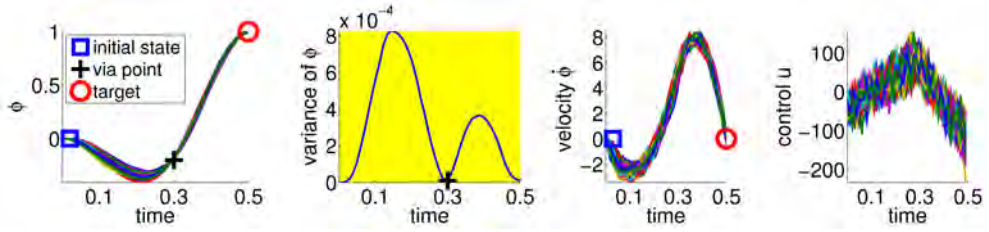
Biological movement generation combines three interesting aspects: its modular organization in movement primitives, its characteristics of stochastic optimality under perturbations, and its efficiency in terms of learning. In this work TUG aims to integrate stochastic optimal control principles within a movement primitive to enable more efficient learning. A common approach to motor skill learning is to endow the primitives with dynamical systems. Here, the parameters of the primitive indirectly define the shape of a reference trajectory. Instead of endowing the primitives with dynamical systems, TUG proposes to endow movement primitives with an intrinsic probabilistic planning system, integrating the power of stochastic optimal control methods within a movement primitive. The parametrization of the primitive is a learned graphical model that represents the dynamics and intrinsic cost function such that inference in this graphical model yields the control policy. TUG parametrizes the intrinsic cost function using task-relevant features, such as the importance of passing through certain via-points. The system

[2]https://redmine.amarsi-project.eu/documents/125

dynamics as well as intrinsic cost function parameters are learned in a reinforcement learning setting. In experiments using complex dynamic balancing tasks TUG has demonstrated that their movement representation facilitates learning significantly and leads to better generalization to new task settings without re-learning. This work has been submitted to a journal. In future research TUG will investigate the benefits of this compact movement representation for imitation learning, where potentially only few characteristic features of movements, i.e. the step length for a walking movement are sufficient for a fast acquisition of new motor skills.



(a) Optimal policy learned with DMPs (average costs over 1000 trajectories: 1286 ± 556)



(b) Optimal policy learned with PMPs (average costs over 1000 trajectories: 1173 ± 596)

Fig. 14: This figure illustrates the best available policies for the DMPs and the PMPs for a simple via-point task. From left to right shown are the point mass trajectories, the variance of these trajectories, the velocity of the point mass, and the applied accelerations. The agent has to pass the via-point at 0.3s and deal with the stochasticity of the system. The plots show 100 trajectories reproduced with the optimal parameters for the DMPs (a) and 100 trajectories with the (handcrafted) optimal parameters for PMPs (b). The PMP approach is able to reduce the variance of the movement if it is relevant for the task, while the DMPs can only suppress the noise in the system throughout the trajectory in order to get an acceptable score. This advantage is also reflected by the average costs over 1000 trajectories. The DMP solution achieved cost values of 1286 ± 556 whereas the PMP result was 1173 ± 596.

Novelty of this work: Due to the use of the intrinsic planning system this novel movement representation complies with basic principles of stochastic optimal control. For example, the developed planning movement primitives (PMPs) are able to account for the motor variability often observed in human motion. Instead of suppressing the noise of the system by following a single reference trajectory, the PMPs are able to learn to intervene the system only if it is necessary to fulfill a given task, also known as the minimum intervention principle. This allows a much higher variance in parts of the trajectory where less accuracy is needed and is illustrated in Figure 14 on a simple via-point task. Current methods like the widely used Dynamic Movement Primitives (DMPs), which rely on a reference trajectory, are not able to reproduce these effects.

References

[1] Seyed Mohammad Khansari-Zadeh and Aude Billard, "Learning stable non-linear dynamical systems with gaussian mixture models," *IEEE Transaction on Robotics*, vol. 27, no. 5, pp. 943–957, 2011.

[2] S. M. Khansari-Zadeh, K. Kronander, and A. Billard, "Learning to play minigolf: A dynamical system-based approach," *Advanced Robotics*, 2012.

[3] K. Kronander, S. M. Khansari Zadeh, and A. Billard, "Learning to control planar hitting motions in a monigolf-like task," in *Proc. of the IEEE/RSJ Int. Conf. on Intelligent Robots and Systems (IROS)*, 2011.

[4] S.-M. Khansari-Zadeh and A. Billard, "A dynamical system approach to realtime obstacle avoidance," *Autonomous Robots*, vol. 32, pp. 433–454, 2012, 10.1007/s10514-012-9287-y.

[5] Seungsu Kim, Elena Gribovskaya, and Aude Billard, "Learning motion dynamics to catch a moving Object," in *10th IEEE-RAS International Conference on Humanoid Robots*, 2010.

[6] Ashwini Shukla and Aude Billard, "Coupled dynamical system based arm-hand grasping model for learning fast adaptation strategies," *Robotics and Autonomous Systems*, vol. 60, no. 3, pp. 424–440, 2012.

[7] S. Vijayakumar and S. Schaal, "Locally weighted projection regression: An o(n) algorithm for incremental real time learning in high dimensional space," in *Proc. of 17th Int. Conf. on Machine Learning (ICML)*, 2000.

[8] S. Schaal, C. Atkeson, and S. Vijayakumar, "Scalable locally weighted statistical techniques for real time robot learning," *Applied Intelligence - Special*

issue on Scalable Robotic Applications of Neural Networks, vol. 17, no. 1, pp. 49 – 60, 2002.

[9] C. Rasmussen and C. Williams, *Gaussian processes for machine learning*, Springer, 2006.

[10] G. McLachlan and D. Peel, *Finite Mixture Models*, Wiley, 2000.

[11] Seungsu Kim and Aude Billard, "Estimating the non-linear dynamics of free-flying objects," *Robotics and Autonomous Systems*, vol. 60, no. 9, pp. 1108–1122, 2012.

[12] R. Bischoff, J. Kurth, G. Schreiber, R. Köppe, A. Albu-Schäffer, D. Beyer, O. Eiberger, S. Haddadin, A. Stemmer, and G. Grunwald, "The KUKA-DLR Lightweight Robot arm – a new reference platform for robotics research and manufacturing Summary / Abstract Stages of research and product development," *Joint 41th International Symposium on Robotics and 6th German Conference on Robotics*, pp. 741–748, 2010.

[13] Arne Nordmann, Christian Emmerich, Stefan Rüther, Andre Lemme, Sebastian Wrede, and Jochen J Steil, "Teaching nullspace constraints in physical human-robot interaction using reservoir computing," in *International Conference on Robotics and Automation*, St. Paul, 2012, pp. 1868 – 1875.

[14] Ananda Freire, Andre Lemme, Jochen J Steil, and G. Barreto, "Learning visuo-motor coordination for pointing without depth calculation," in *European Symposium on Artificial Neural Networks*, 2012, pp. 91–96.

[15] Andre Lemme, Ananda Freire, G. Barreto, and Jochen JSteil, "Kinesthetic teaching of visuomotor coordination for pointing by the humanoid robot icub," *Neurocomputing*, under revision.

[16] René Felix Reinhart, Andre Lemme, and Jochen Jakob Steil, "Representation and generalization of bi-manual skills from kinesthetic teaching," in *2012 IEEE-RAS International Conference on Humanoid Robots*, accepted.