Genet Program Evolvable Mach (2012) 13:137–157
DOI 10.1007/s10710-011-9147-0

Learning local linear Jacobians for flexible and adaptive robot arm control

Patrick O. Stalph · Martin V. Butz

Received: 6 May 2011 / Revised: 1 August 2011 / Published online: 16 September 2011

Abstract Successful planning and control of robots strongly depends on the quality of kinematic models, which define mappings between configuration space (e.g. joint angles) and task space (e.g. Cartesian coordinates of the end effector). Often these models are predefined, in which case, for example, unforeseen bodily changes may result in unpredictable behavior. We are interested in a learning approach that can adapt to such changes—be they due to motor or sensory failures, or also due to the flexible extension of the robot body by, for example, the usage of tools. We focus on learning locally linear forward velocity kinematics models by means of the neuro-evolution approach XCSF. The algorithm learns self-supervised, executing movements autonomously by means of goal-babbling. It preserves actuator redundancies, which can be exploited during movement execution to fulfill current task constraints. For detailed evaluation purposes, we study the performance of XCSF when learning to control an anthropomorphic seven degrees of freedom arm in simulation. We show that XCSF can learn large forward velocity kinematic mappings autonomously and rather independently of the task space representation provided. The resulting mapping is highly suitable to resolve redundancies on the fly during inverse, goal-directed control.

Keywords Robot kinematics · Inverse problems · Actuator redundancy · Learning classifier systems · XCSF

P. O. Stalph · M. V. Butz (✉)
Computer Science, Cognitive Modeling, University of Tübingen, Sand 14, 72076 Tübingen, Germany
e-mail: butz@informatik.uni-tuebingen.de

P. O. Stalph
e-mail: stalph@informatik.uni-tuebingen.de

1 Introduction

Robot control is a challenging task and a large variety of methods with different prerequisites have been developed over the last decades. With the aim of mimicking the flexibility and adaptiveness of humans, we present an approach that *learns* to control the kinematics of a robot arm from scratch. In particular, we are aiming at a system that satisfies the following requirements:

1. *No external knowledge*
   The structure, capabilities, and representations (sensory and motor) of the robot are assumed to be generally unknown.
2. *Online learning*
   Learning takes place online while controlling the device.
3. *Self-supervised learning*
   The learning system learns self-supervised, generating the learning data by itself in online interaction with the environment.
4. *Redundancy preservation*
   Instead of learning an inverse model directly resolving the actuator redundancy during learning, the available redundancy is learned and resolved during movement execution, allowing maximum behavioral flexibility.

In this paper, we are concerned with learning and controlling the *kinematics* of a robot arm. Learning a full, accurate model of the *dynamics* is not considered here as it is more complicated due to several reasons. The state space becomes extremely large (velocities and accelerations in addition to positions) and some regions (e.g. low velocity) are relevant while others may not be used at all. However, dynamics can be treated completely separated [9, 25] from the kinematics approach taken here. We focus on learning the velocity kinematics, that is, Jacobians, for directional control. Planning mechanisms for, for example, obstacle avoidance are not considered.

To meet the above requirements, we learn a model of the velocity kinematics of an arm with the XCSF learning algorithm [31, 34, 36]. In our setup, XCSF learns to predict end-effector movements given the current arm configuration and the current motor command—where the arm configuration is specified in joint angles and the motor command comes in the form of small angular changes. Due to the typical redundant degrees of freedom (DoFs) of a robot arm, to move towards a certain point in space a manifold of movement commands are possible. Thus, given a certain goal location, the target direction has to be transferred into the manifold of potential motor commands (inverse velocity kinematics mapping), which can move the arm end effector in the desired direction. To account for additional task constraints, this mapping needs to allow the effective, constraint-dependent resolution of redundant alternatives on the fly upon task invocation. Our XCSF approach solves the inversion problem by learning locally linear forward velocity kinematics models, which can be easily inverted for inverse control with redundancy resolution [6, 10]. A similar approach has also been realized with the *Locally Weighted Projection Regression* (LWPR) algorithm [24]. In either case, the result is a flexible control mechanism that can easily take additional task constraints

into account. An explicit comparison of XCSF and LWPR on kinematic arm control is not considered, as comparable performance was shown elsewhere [27].

While the basic setup of XCSF for arm control was introduced previously [6, 10], here we give a comprehensive introduction to the overall system. More importantly, we extend the previous setup by introducing fully autonomous, goal-oriented behavior during learning using goal-babbling [20]. In contrast to *motor-babbling*, where the system learns from randomly generated *motor commands*, in *goal babbling* random *task goals* are generated and the system tries to reach those goals using its current knowledge. We show that the setup scales up to the control of an anthropomorphic arm with seven degrees of freedom moving through a three dimensional workspace. We also show that successful model learning and resulting control works in other frames of reference, such as a distance and angular encoding of the end-effector location relative to a head-centered frame of reference.

The remainder of this article is structured as follows. First, we introduce robot kinematics background and give an overview of the relevant related work. Section 3 starts with a short introduction to XCSF and then explains how XCSF is applied to learn the velocity kinematics of a robot arm and how this model is used for robot arm control. Section 4 gives the details of a seven DoFs arm simulation and validates the framework in several experiments. The article ends with final conclusions.

2 Background

This section poses the addressed problem of learning a forward velocity kinematics model and using it for inverse control in a general way. Furthermore, the advantages and drawbacks of related work are briefly discussed. We focus on a robot arm, although the proposed framework is generally applicable to any robotic device.

2.1 Robot arm kinematics

To control a robot arm, at least two spaces have to be considered: the arm *configuration space* $\mathcal{C} \subset \mathbb{R}^n$ and the *task space* $\mathcal{T} \subset \mathbb{R}^m$ of the end-effector, with $m < n$ if the arm has redundant DoFs. The task space is often encoded in a Cartesian coordinate system, but other encodings may be used as well.

Due to the robot arm kinematics, a particular configuration $\mathbf{q} \in \mathcal{C}$ fully determines the corresponding task space location $\boldsymbol{\xi} \in \mathcal{T}$. This mapping from configuration space to task space is called the *forward kinematics* mapping and can be expressed as a typically non-linear function

$$\boldsymbol{\xi} = f(\mathbf{q}). \tag{1}$$

Since movements take place in small steps, it is useful to look at the velocity kinematics. Given the current configuration $\mathbf{q}$, the first-order derivative of (1) with respect to time can be written as

$$\dot{\boldsymbol{\xi}} = \mathbf{J}(\mathbf{q})\dot{\mathbf{q}} \tag{2}$$

where $\dot{\boldsymbol{\xi}}$ is the velocity vector in task space, $\dot{\mathbf{q}}$ is the joint-space velocity vector, and $\mathbf{J}(\mathbf{q}) = \partial f / \partial \mathbf{q}$ is the $m \times n$ Jacobian matrix.

To control a robot device, a translation from the given task to the required control command is required. However, the inverse is not uniquely defined if $n > m$. Often this is called the *problem of inverse kinematics*. More precisely, the inverse of $\mathbf{J}$ represents an under-determined system with infinitely many solutions.[1]

One way to pick a solution from this set of solutions is the so-called *pseudoinverse* or *Moore-Penrose* matrix [1], which represents the solution with minimum norm. If $\mathbf{J}$ is lower-rectangular and of full rank, the pseudoinverse is given by

$$\mathbf{J}^{\dagger} = \mathbf{J}^{T}(\mathbf{J}\mathbf{J}^{T})^{-1}. \tag{3}$$

The general solution to the inverse velocity kinematics of (2) is given by

$$\dot{\mathbf{q}} = \mathbf{J}^{\dagger}\dot{\boldsymbol{\xi}} + \underbrace{(\mathbf{I} - \mathbf{J}^{\dagger}\mathbf{J})\dot{\mathbf{q}}_0}_{\text{null space velocity}}, \tag{4}$$

where $\dot{\mathbf{q}}_0$ is an arbitrary joint space velocity. The first summand represents the translation of the desired task space velocity into configuration space. The second summand yields zero task space velocity, since $\mathbf{I} - \mathbf{J}^{\dagger}\mathbf{J}$ is the null space projection matrix that represents available redundancy of the current configuration. Omitting the second summand (or setting $\dot{\mathbf{q}}_0 = 0$) results in zero null space movement. This is one widely used solution to the problem of inverse kinematics [18].

Advanced methods exploit the redundancy in order to avoid angular boundaries or singularities. For example, let $\dot{\mathbf{q}}_0 = s[(\mathbf{q}_{\min} + \mathbf{q}_{\max})/2 - \mathbf{q}]$, where $\mathbf{q}$ is the current configuration, $\mathbf{q}_{\min(\max)}$ is the minimum (maximum) joint angle configuration, and $0 < s \leq 1$ is a scaling factor. The scaling factor should be set such that resulting velocities stay in acceptable bounds. Here, $\dot{\mathbf{q}}_0$ describes a movement towards the center of each joint's range. Applied in (4), the available redundancy is used to avoid extreme angular configurations. While other strategies are possible, this constraint is applied throughout the main experiments in this work. Later, we also analyze another constraint and how it affects the trajectory.

To sum up, one way to robot control is given by the first-order kinematics, which is uniquely defined via the Jacobian matrix $\mathbf{J}$. In this case, the pseudoinverse $\mathbf{J}^{\dagger}$ plus an additional constraint $\dot{\mathbf{q}}_0$ yield a unique solution $\dot{\mathbf{q}}$ given a desired task space velocity $\dot{\boldsymbol{\xi}}$. Since we learn locally linear approximations of the Jacobians with XCSF, these equations are directly applicable for redundancy resolution within our XCSF-based learning and control framework.

2.2 Related work

We restrict the discussion on related work to those approaches that fulfill at least some of our targeted requirements, namely, approaches that learn a model

[1] The discussion of singularities is beyond the scope of this article.

autonomously online without any prior model knowledge and that *preserve redundancies* while doing so.

Learning models that resolve redundancies during learning and thus have only one inverse solution available for a given goal can be clustered into *direct inverse modeling* (DIM) approaches [15], *resolved motion rate control* (RMRC) approaches [15, 30], and *feedback error learning* (FEL) approaches [16]. Also, *direct reinforcement learning* was applied to model human-like arm control development [3, 4]. DIM approaches learn the inverse kinematics of a robotics device directly and are known to possibly not converge given non-convex configuration subspaces for specific goal states. RMRC approaches learn first the forward kinematics of the partial derivatives, that is, an approximation of the Jacobian of the device, and then apply constraints to also learn a suitable inverse model. Distal supervised learning is one form of RMRC [15] where the redundancy is implicitly resolved by the back-propagation mechanism employed. FEL approaches rely on the pre-existence of a simple controller. These approaches optimize the control commands and eventually take over most of the control requirements. Redundancy is resolved by the structure of the given simple controller. FEL thus develops hierarchical control structures building upon a simple controller, improving behavior efficiency and smoothness [12, 38].

A more recent approach that was shown to be able to learn various kinds of direct inverse models is the LWPR algorithm [11, 28]. In the control application case, LWPR was trained to mimic an observed dynamic control behavior, such as drawing a figure eight. Again, redundancies were immediately resolved during learning, mimicking the observed behavior. Nonetheless, similar to XCSF, LWPR builds a population of receptive fields (RFs) that employ locally linear models to approximate a multi-dimensional, non-linear function. LWPR is a statistics-based machine learning algorithm, while XCSF is based on an evolutionary algorithm. The comparison of LWPR with XCSF on general function approximation problems showed that XCSF often yields better spatial structurings while LWPR yields slightly faster convergence [27]. Despite the close resemblance, in most applications published, LWPR learns a direct inverse model [29, 28], that is, a one-to-one mapping from the task space to the configuration space, thus loosing knowledge about available redundant alternatives.

Our learning approach stores the redundant actuator capabilities during learning and can flexibly resolve the available redundancies on the fly during goal-directed behavior, thus being distinct from all the above approaches and their more recent derivatives. Storing redundancies and exploiting behavioral alternatives is a typical human property. Psychological models of these capabilities were introduced with the posture-based (PB) theory of reaching and grasping [21–23]. In this architecture, a set of exemplary postures is stored and evaluated given a desired goal state. Directional movements are then executed towards the goal using the provided kinematics model.

The so-called SURE_REACH framework [7] is a population-based neural approach, which subsumes the PB theory. It learns the kinematics model used for model-based reinforcement learning control online. RFs. Hebbian and temporal Hebbian learning form connections between configuration and task space and within

these spaces, respectively. The latter mapping allows for flexible movement planning whereas the former encodes the redundant configuration alternatives given a particular goal location. Despite its appealing features as a neural-psychological model of reaching behavior, the model has the drawback of scaling exponentially in the number of DoFs modeled. One approach to avoid this scaling problem is to learn locally linear models that apply in extended subregions of the high-dimensional input space, which is further explored in the remainder of this paper.

Another approach using LWPR that preserves actuator redundancies learns locally linear forward kinematic models instead of inverse models [24]. Here, LWPR learns the forward kinematics from (1). The derivation of the velocity kinematics is straightforward due to the locally linear mapping [24]. Since the derived Jacobians are linear, they are easily inverted, while the redundancy is exploited to fulfill additional constraints. However, for the three DoFs planar arm that was tested, up to 8000 RFs appeared necessary. Thus, it remains an open research challenge to apply LWPR to a more complex robotic device and learn a compressed representation of the full kinematics mapping. That is, it essentially remains unclear to what extent LWPR is able to learn in large, fully-sampled problem spaces [27]. Moreover, LWPR optimizes the spatial clustering for an accurate kinematics model, but the first-order derivative is required for control—thus, the structuring is not optimized for control.

We now turn to our XCSF approach, which learns the forward velocity kinematics of an arm, thus storing redundancies in the locally linear Jacobian mapping. Locality is defined within the configuration space, since a small change in the configuration yields a unique change in task space given a particular current configuration. Thus, our system resolves redundancies on the fly and does not require the determination of derivatives during goal-directed behavior.

3 Directional control with XCSF

This section briefly introduces the general framework of the XCSF Learning Classifier System. Most importantly, we describe how XCSF learns a kinematic model and how this can be used for control.

3.1 XCSF: A learning classifier system

Learning Classifier Systems were introduced by John H. Holland [13]. Its most prominent implementation is XCS[2] [31, 32], which has been successfully applied to various applications such as binary classification tasks [5, 31], data mining problems [2, 33], and function approximation [8, 26, 35, 36]. Since we are interested in learning a kinematic model, we apply the function approximation mode of XCS—then called XCSF.

XCSF is able to approximate a non-linear, multi-dimensional function $f{:}X \rightarrow Y, f(\mathbf{x}) = \mathbf{y}$ using piecewise, linear models. Therefore a population of receptive

[2] Sometimes XCS is said to be the "eXtended Classifier System", but this acronym was not intended.

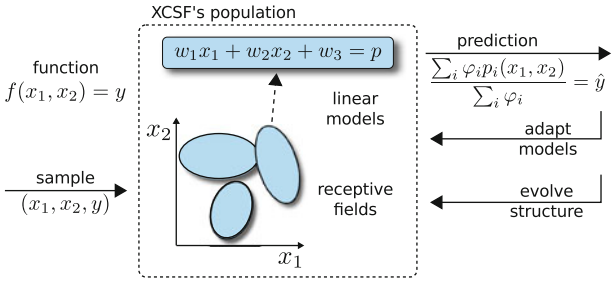


Fig. 1 Illustration of XCSF's workflow for an exemplary two-dimensional function with just one-dimensional output $f(x_1,x_2) = y$. Every iteration XCSF determines the RFs that match the current sample input x_1,x_2. Matching RFs generate a prediction $p_i(x_1,x_2)$ based on their respective internal linear models. The final prediction $\hat{y}$ is computed as the sum of individual predictions, weighted by fitness estimates φ_i. Using the prediction error $y - \hat{y}$, those models are updated and the structure is optimized via evolution with the goal of accurate and maximally general RFs

fields[3] is evolved, where each RF covers a subspace of the input space X. In its respective subspace, the RF learns a linear model $p(\mathbf{x}) = \hat{\mathbf{y}} \approx \mathbf{y}$ to approximate the underlying function. The prediction error $f(\mathbf{x}) - p(\mathbf{x}) = y - \hat{y}$ is used to adapt the linear models in a gradient descent fashion, typically using recursive least squares. In turn, a genetic algorithm (GA) is applied to search for a better clustering of the input space by modifying the position, size, and shape of the RFs, where a scaled inverse of the prediction error yields the fitness. An illustration of the workflow is given in Fig. 1. In sum, XCSF evolves a population of RFs with the goal of accurate and maximally general approximations.

3.2 XCSF-based control via local Jacobians

With small modifications, XCSF is applicable to learn the velocity kinematics from (2). In other words, the goal is to learn a mapping from configuration space velocities $\dot{\mathbf{q}}$ to task space velocities $\dot{\boldsymbol{\xi}}$, depending on the current configuration $\mathbf{q}$. While any function approximation method could be applied to learn a mapping $\mathbf{q} \times \dot{\mathbf{q}} \mapsto \dot{\boldsymbol{\xi}}$, not every method might be suitable. First of all, the input space is $2n$-dimensional, but there is a strong relation from configuration space to its first derivative and it is possible to reduce the complexity to an n-dimensional mapping. Another requirement is a suitable representation to quickly invert the model during movement execution.

XCSF is well-suited for the task, since the algorithm is able to cluster a context space while learning a function that operates on a different space, but depends on the context. In our task the current configuration is the context and XCSF clusters the configuration space with RFs, e.g. rotating ellipsoids [8]. In turn, each RF approximates the Jacobian matrix using linear recursive least squares approximation (see Fig. 2). Together, the complexity is linear in the dimensionality of the

[3] Usually called classifiers in the XCS literature.

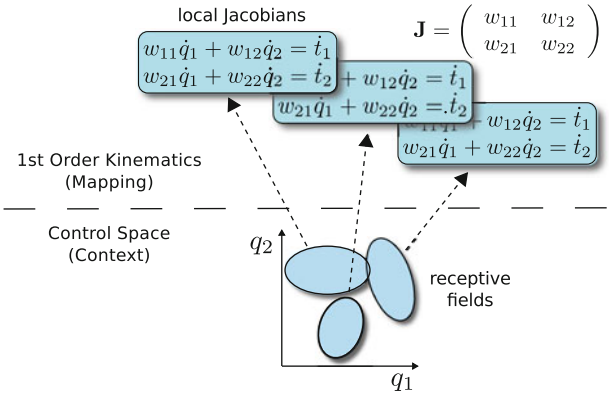


Fig. 2 In order to approximate the velocity kinematics, XCSF generates RFs that cover the configuration space. Over time, each RF learns a localized Jacobian in its respective context by means of recursive least squares. The accuracy of the approximation guides the evolutionary search for a better context space clustering. For illustrative reasons, both, the configuration space and the task space are two-dimensional

configuration space. Given a particular configuration, the linear Jacobian is easily inverted.

Given the current configuration $\mathbf{q}$, a desired task space velocity $\dot{\boldsymbol{\xi}}_{\text{des}}$, and a constraint $\mathbf{q}_0$, the workflow is as follows. First, XCSF's population is scanned for *experienced, matching* RFs, that is, RFs that cover the current configuration and have seen at least $2n$ samples.[4] The Jacobian $\mathbf{J}$ of each active RF i is inverted with respect to the constraint $\dot{\mathbf{q}}_0$. This yields the predicted configuration space velocity $\dot{\mathbf{q}}_{\text{i,pred}}$ from the ith RF. The final configuration space velocity is computed as an activity-weighted average

$$\dot{\mathbf{q}}_{\text{exc}} = \frac{\sum_i a_i \dot{\mathbf{q}}_{\text{i,pred}}}{\sum_i a_i}, \tag{5}$$

where $a_i = \exp(-(\mathbf{q} - \mathbf{c}_i)^T \mathbf{D}_i (\mathbf{q} - \mathbf{c}_i))$ is the Gaussian activity of the ith RF, where the shape is described by the matrix $\mathbf{D}_i$ at center $\mathbf{c}_i$. A positive, semi-definite, quadratic matrix $\mathbf{D}_i$ describes an n-dimensional, not necessarily axis-aligned ellipsoid [8, 27]. In sum, with increasing distance, the influence of an RF decays as also its accuracy decreases. Finally, $\dot{\mathbf{q}}_{\text{exc}}$ is executed and the actual resulting task space velocity $\dot{\boldsymbol{\xi}}_{\text{act}}$ is stored. The previous configuration, the executed angular motion, and the resulting task space velocity yield the next learning sample $(\mathbf{q}, \dot{\mathbf{q}}_{\text{exc}}, \dot{\boldsymbol{\xi}}_{\text{act}})$ for XCSF.

Initially the population is empty and task space velocity requests cannot be answered. If no RF covers a particular configuration, a new one with random shape is generated (so-called covering). However, the Jacobian is initially a zero matrix and consequently a zero motor command is predicted. While this would prevent learning in an environment without noise and external forces (e.g. gravity), in more realistic environments the problem does not occur. We model a small motor noise

[4] An n-dimensional linear approximation of less than $2n$ samples is probably inaccurate.

by adding Gaussian noise to the angular control command that is executed. The standard deviation $\boldsymbol{\sigma}$ is set to $0.05(\mathbf{q}_{\max} - \mathbf{q}_{\min})$, where $\mathbf{q}_{\max}$ ($\mathbf{q}_{\min}$) is the upper (lower) bound of the configuration space. This merely disturbs the movement, but helps to get initial learning data and also improves the exploration of the state space: Without noise on the control signal, the controller will follow one trajectory, which prevents learning of weights for dimensions orthogonal to that trajectory. In contrast, noisy movements also tackle orthogonal directions.

As the robot device moves along a trajectory, the learning samples are highly biased. With high frequency sampling, XCSF would tend to accurately represent the current configuration and configurations experienced lately, while RFs that cover the remaining configuration space get lost. This is due to the GA that reproduces RFs at the current input while deleting others from the population. We approach this issue by "learning only from failures", that is, if XCSF's approximation is sufficiently accurate,[5] no updates to the RFs model nor the genetic algorithm are triggered. On the other hand, if the approximation is insufficient, the Jacobians of currently active RFs are updated via recursive least squares and the GA further optimizes the contextual clustering at the current configuration. Over time the full space is well approximated and XCSF's model is just used for prediction but rarely modified.

4 Experimental validation

We use a simplified anthropomorphic, seven DoFs arm to validate the XCSF based framework. Details of the kinematic specification are depicted in Fig. 3. Throughout all experiments, the configuration space is the seven-dimensional space of joint angle configurations $\mathbf{q} = [q_1, \ldots, q_7]^T$ and the end effector location defines the task space. Thus, XCSF learns to map joint angle velocities to end effector velocities, where the context is the current joint configuration. In turn, XCSF's model of the velocity kinematics can be used to control the end effector location.

Most of XCSF's parameters were set to standard values.[6] Two values for the maximum population size are tested, namely 500 and 2,000. The number of movement iterations is set to 500,000—however, as mentioned above active learning does not occur at every iteration, but only when the approximation is inaccurate. The threshold θ_{GA} specifies how frequently the GA is activated and is increased to 200 in order to compensate for the imbalanced sampling as suggested in [19]. Towards the end of a run, condensation [32] is activated, that is, reproduction without mutation and crossover to remove evolutionary overhead. During condensation highly fit classifiers are strengthened (reproduced but not modified) while

[5] The approximation is said to be accurate, if the prediction error is below a target error ε_0, which is one of XCSF's parameters.

[6] XCSF settings: $\alpha = 1$, $\beta = 0.1$, $\delta = 0.1$, $\nu = 5$, $\chi = 1$, $\mu = 1/42$, $\theta_{\text{del}} = 20$, $\theta_{\text{sub}} = 20$. The target error is set to $\varepsilon_0 = 0.001$. Uniform crossover, GA subsumption, and tournament selection with $\tau = 0.4$ are applied. Condensation [32] is applied after 450,000 iterations. Rotating ellipsoidal RFs [8] and linear recursive least squares prediction [17] are used.

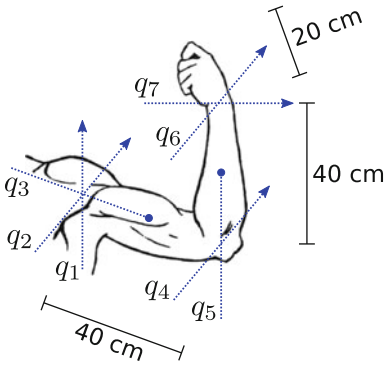


Fig. 3 Specification of the seven DoFs arm in simulation. The arm has a total length of 100 cm. Rotation axes $q_1, \ldots, q_7$ are drawn as *dashed lines*; the two rotary joints are depicted with a *circle*. Joint angles are restricted to $q_1 \in [-1.0, 2.9], q_2 \in [-1.5, 1.5], q_3 \in [-1.0, 1.0], q_4 \in [-0.0, 2.8], q_5 \in [-0.7, 1.0], q_6 \in [-1.5, 1.5], q_7 \in [-0.5, 0.7]$. For each joint the maximum rotation velocity is restricted to 0.01 radians per step. Similar to a human arm, the shoulder has three DoFs, namely flexion-extension (q_1), abduction-adduction (q_2), and internal-external rotation (q_3). The elbow allows for two DoFs: flexion-extension (q_4) and pronation-supination (q_5). Another two DoFs, that is, abduction-adduction (q_6) and flexion-extension (q_7), are located at the wrist. With all angles $q_i = 0$ the arm is fully extended, while the *picture* shows an almost centered configuration

inaccurate classifiers get deleted due to the limited population size. All experimental results are averaged over 20 independent trials.

Goal locations $\boldsymbol{\xi}_{\text{des}}$ are generated randomly during learning and a simple greedy control scheme computes the desired task space velocity $\dot{\boldsymbol{\xi}}_{\text{des}} = \boldsymbol{\xi}_{\text{des}} - \boldsymbol{\xi}$ from the current task space location $\boldsymbol{\xi}$. Thus, requests to XCSF always assume that the goal can be reached within one step, but the executed angular movements—predicted by XCSF's population—are constrained to a maximum velocity, dependent on the the underlying robot device. It is important to note that *task space goals* are generated, that is, the system acts in a *goal directed* way from the beginning. This is called *goal babbling* [20]. The system autonomously uses its learned model to generate motor commands which stands in contrast to *motor babbling*, where random motor commands are generated externally.

4.1 First experiments

A typical graph of XCSF's learning performance is depicted in Fig. 4, but due to trajectory learning the error measurement is misleading: The local linear models are quickly adapted to the current configuration and only minor updates are required to maintain a low error along a trajectory. Due to this fact, it is also misleading to measure the online reaching performance during learning. Eventually, the local model around the current configuration gets updated fast enough to show suitable reaching performance, but the remaining configuration space is not even covered.

In order to measure reaching performance thoroughly, we deactivate learning every 10,000 iterations and run 1,000 randomly created reaching tasks *offline*. The arm is set to a random configuration and a nearby task space goal is activated for XCSF, but the population is not updated during testing. A task space goal is said to be *successfully*

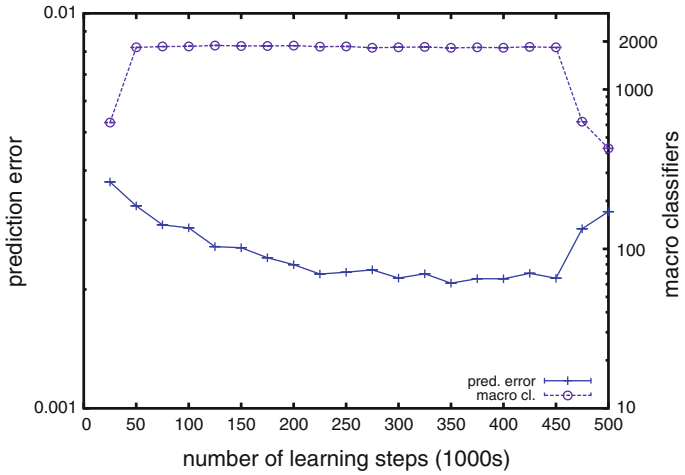


Fig. 4 XCSF's performance during learning. While the population quickly reaches its maximum size (here $N = 2{,}000$), the mean absolute error is intriguingly low (less than 0.004 cm from the beginning) and does not reflect the average prediction error of the whole model due to learning along trajectories. With condensation starting at iteration 450,000, the population size drops drastically, while the error increases slightly. The *left* and *right vertical axis* is log-scaled

reached, when the distance from end effector to target is less than 5% arm length. The percentage of successful tasks is stored along with the *task space efficiency*, that is, the minimal task space distance from initial to target location divided by the actual task space distance traveled. Unfortunately, not every task space location can be reached successfully from any configuration, even if XCSF's population of RFs accurately models the velocity kinematics. The simple greedy control scheme assumes that there is a direct path in task space, but this is not always true. More sophisticated planning algorithms could be used, but this goes beyond the scope of this article. Instead we avoid deadlocks by generating reaching tasks that bridge just a short distance in configuration space. Additionally, generated start configurations lie within the inner 80% of each joint's range. Furthermore, we do not expect to see 100% path efficiency, as the shortest path is not a straight line when angular boundaries are hit. Since percentage measurements are not normally distributed, mean and variance are not suitable to visualize the data. Instead, we report the median as well as first and third quartile.

The experimental results over 20 independent runs are depicted in Fig. 5, where 50 offline tests are conducted over the learning time. With a maximum population size of 500 RFs the reaching performance is not as stable as with a size of 2,000 RFs. Increasing the population size allows for higher precision and more stable performance in offline tests. With 2,000 RFs, the worst percentage of successful reaching tasks (out of the 20 experiments) in the final population was 99.1%. The same tests would always yield 100% reaching performance in the online environment, when RFs were continuously updated. It is important to note that the final population size is reduced by means of condensation [32] during the last 50,000 iterations in order to clean up evolutionary overhead. Figure 4 represents a typical graph for the experiments with $N = 2{,}000$. After condensation 510.85 ± 27.31 RFs remain in the

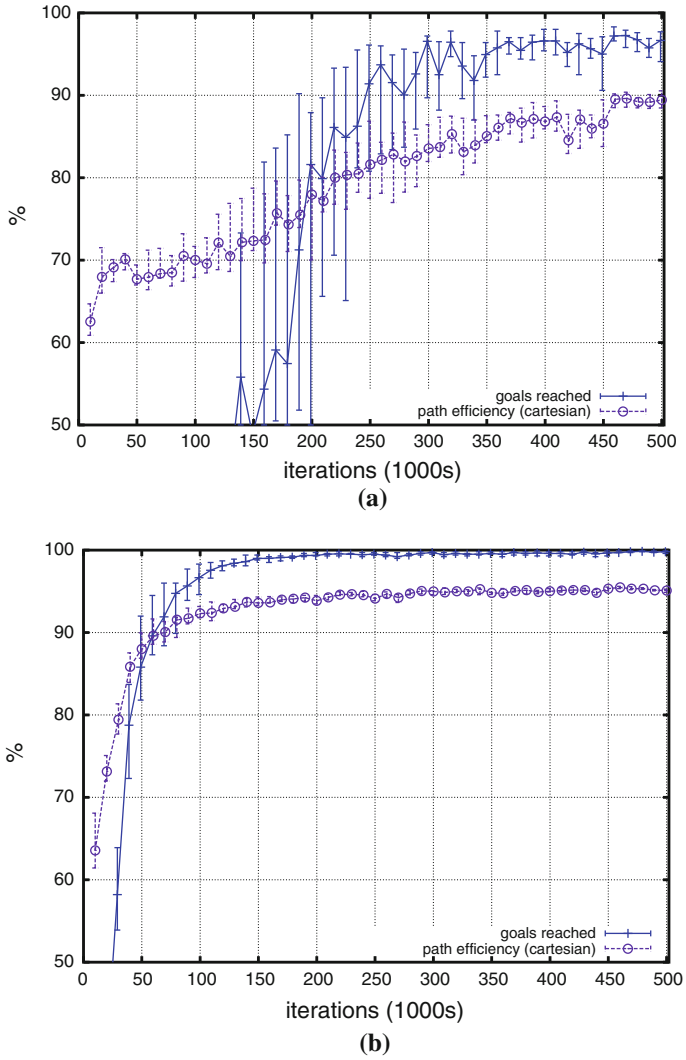


Fig. 5 XCSF's offline performance using a Cartesian task space representation for maximum population sizes **a** $N = 500$ RFs and **b** $N = 2{,}000$ RFs. Every 10,000 learning iterations an offline test is conducted. The percentage of *reached goals* specifies how many out of 1,000 randomly generated reaching tasks were successful, while the *path efficiency* describes how close XCSF's task space trajectory is to a *straight line*. The *graphs* show the median as well as first and third quartiles over 20 independent runs

seven dimensional population. Looking at the final iterations in Fig. 5, we see that this process does not affect the reaching performance.

4.2 Representational independence

Up to now, a Cartesian task space representation was used, that is, end effector coordinates are stored as three-dimensional vectors $\boldsymbol{\xi} = [x, y, z]^T$. Alternatively, an

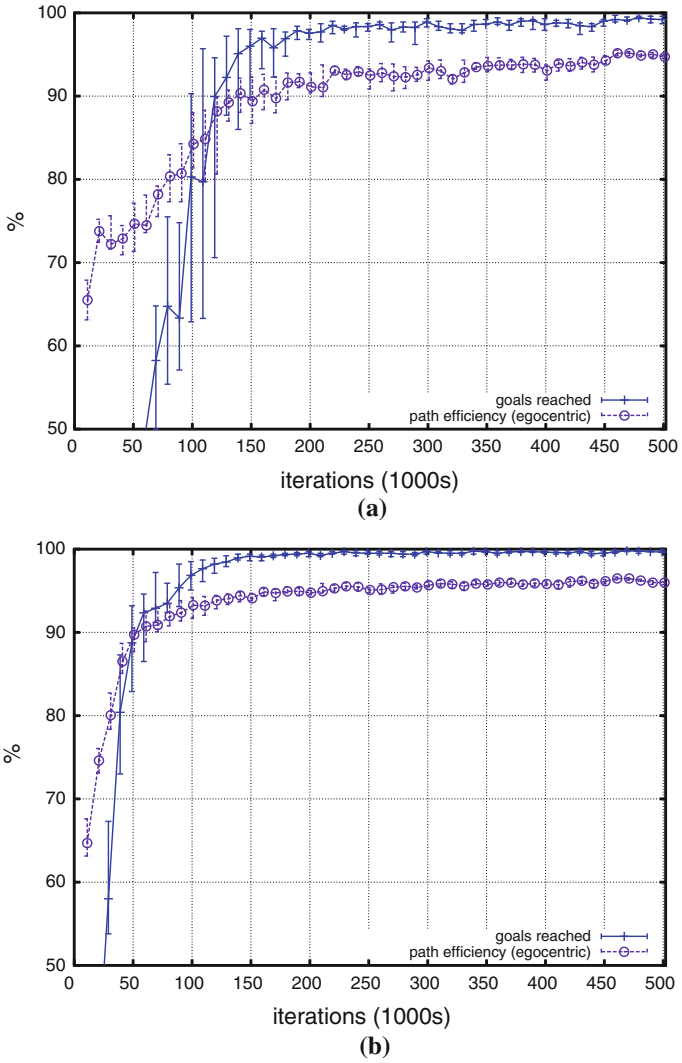


Fig. 6 Offline performance for the egocentric task space representation. Shown are median as well as first and third quartiles over 20 independent trials. **a** 500 RFs, **b** 2,000 RFs

egocentric view $\boldsymbol{\xi} = [d, \varrho, \psi]^T$ of the end effector consisting of distance d, horizontal angle ϱ, and vertical angle ψ can be applied. More precisely, the *center of view* is set to be the origin of the Cartesian coordinate system and the shoulder is located at $[15, -10, 0\ \text{cm}]^T$. Thus, the origin resembles the location of an imaginary vision system. The focus of this section is on the representational independence, not on the alternative coordinate system.

The egocentric representation is computed as[7]

$$\begin{aligned} d &= \sqrt{x^2 + y^2 + z^2}, \\ \varrho &= \text{atan2}(x, -z), \\ \psi &= \text{sign}(y) \arccos\left(\frac{x^2 + z^2}{d\sqrt{x^2 + z^2}}\right). \end{aligned} \tag{6}$$

If $x = z = 0$ and $y \neq 0$, ϱ is set to $\text{sign}(y)\pi/2$. If $x = y = z = 0$, both angles are undefined and set to zero, which creates a point gap in this representation. The horizontal angle $\varrho \in [-\pi, \pi]$ is the angle in x-z-plane between the end effector location and $[0, 0, -1]^T$, which is the vector that "points to the front". Thus, in this direction ϱ is zero. The vertical angle $\psi \in [-\pi/2, \pi/2]$ represents the signed angle between $[x, y, z]^T$ and $[x, 0, z]^T$.

Performance with this alternative representation (Fig. 6) is comparable to the Cartesian one (Fig. 5). With a population size of $N = 2{,}000$ RFs, the worst percentage of successful reaching tasks during offline testing in the final population was 98.9% for the egocentric representation. Comparing the learning speed with $N = 500$ (Figs. 5a, 6a), the egocentric representation seems to allow for faster learning. Intuitively this makes sense, as the angular representation yields a more linear function surface. On the other hand, the point gap complicates transitions close to the center of view. Thus, the final performance remains comparable.

4.3 Spatial structuring

Also within an egocentric task space XCSF's generalization capabilities allow for a suitable representation of the seven DoF velocity kinematics with intriguingly few RFs: 503.65 ± 31.93 RFs remain in the final population, while initially up to 2,000 RFs are evolved by the GA. Unfortunately, the spatial clustering is difficult to analyze for seven DoFs, since the space is covered by rotated, seven-dimensional ellipsoids. However, using a reduced number of controlled joints, we can visualize XCSF's population of RFs with respect to the contextual clustering. Figure 7 illustrates the evolved structure for two and three DoF arms. Only joints q_1, q_4 (two DoF), and q_2 (three DoF) are controlled, while the other joint angles remain fixed at a centered position. Again we can see that XCSF has a harder time with the Cartesian representation, because smaller RFs and a rather strict structure are required to maintain a suitable accuracy. The egocentric representation induces more linearities and almost spherical RFs suffice to learn a suitable prediction.

Interestingly, rather small RFs are evolved for the egocentric task space at the top right corner (Fig. 7b). This corresponds to maximum angle in q_1 and q_4. In other words, shoulder and elbow are flexed and the end effector almost reaches the center of view. Here, small joint angle changes yield large task space changes for the egocentric representation. Consequently, small RFs are required to maintain an accurate mapping. In contrast, large ellipsoids cover the bottom part that

[7] $\text{atan2}(x,-z)$ evaluates to $\arctan(-x/z)$ for $z < 0$, $\arctan(-x/z) + \pi$ for $z > 0 \wedge x \geq 0$, $\arctan(-x/z) - \pi$ for $z > 0 \wedge x < 0$, $\pi/2$ for $z = 0 \wedge x > 0$, $-\pi/2$ for $z = 0 \wedge x < 0$, and 0 for $z = x = 0$.

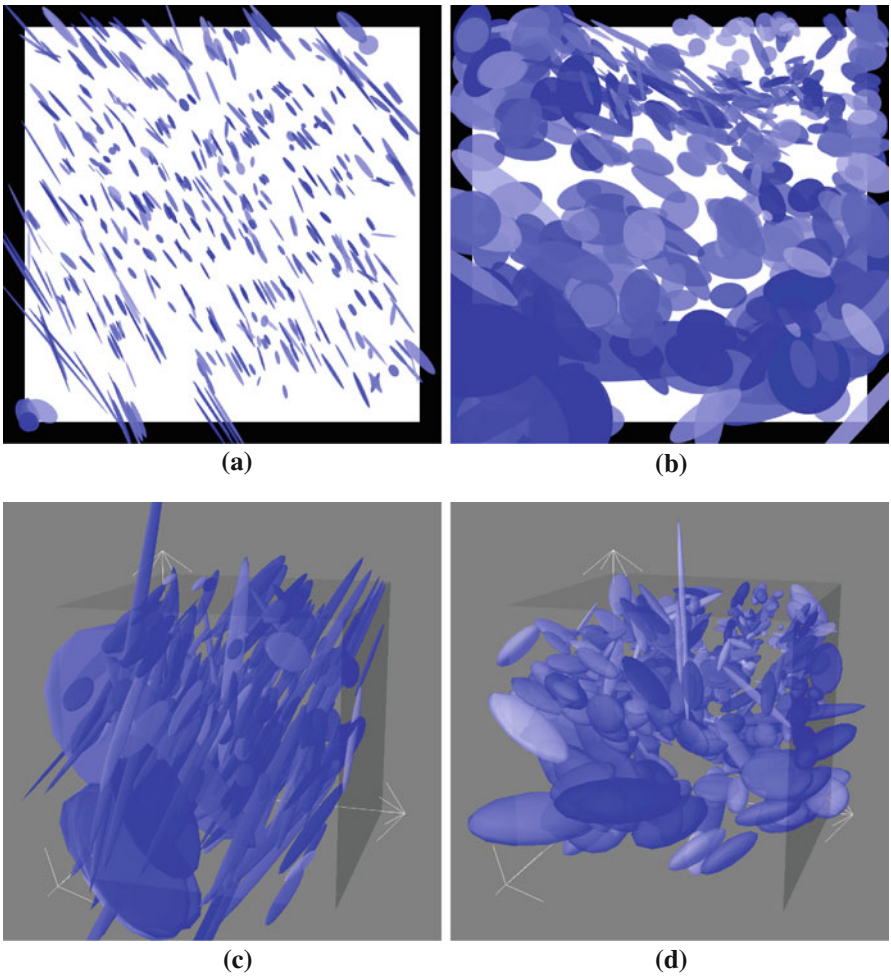


Fig. 7 Visualization of XCSF's final contextual clustering for a reduced number of DoFs. All RFs are depicted at 20% of their actual size. **a** 2 DoF, Cartesian representation, **b** 2 DoF, egocentric representation, **c** 3 DoF, Cartesian representation, **d** 3 DoF, egocentric representation

corresponds to a stretched elbow. Here, the distance from end effector to the center of view is large and configuration changes affect the task space location only slightly. For a Cartesian task space, the clustering is more uniform (Fig. 7a, c). This is consistent with the uniform sinusoidal structure of a kinematic forward mapping from joint angles to Cartesian coordinates.

With regard to the full seven joint arm, we can analyze the spatial structuring at least partially, although not as illustrative as above. Boxplots of the normalized extent per dimension of the RFs are depicted in Fig. 8. The larger the extent of a RF in a particular dimension, the more general it is in this dimension. Assuming an optimal clustering, this corresponds to less curvature in the underlying function. An extent of 1 implies that the RF covers the whole dimension. XCSF generalizes

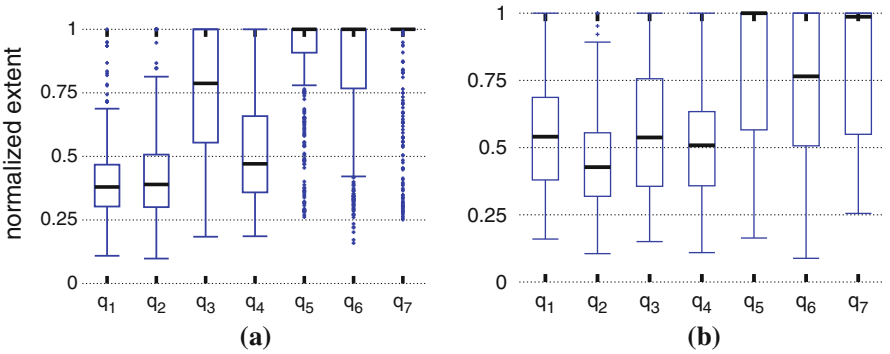


Fig. 8 *Boxplots* of the normalized RF extents. For each dimension, 50% of the data is contained within the *box*, which represents the range from first quartile Q_1 to third quartile Q_3. In each box the median is depicted as a *thick line*. Data points farther away than 1.5 $(Q_3 - Q_1)$ are outliers and are depicted as *crosses*. **a** Cartesian representation, **b** egocentric representation

crudely over all wrist related joint angles $(q_5, \ldots, q_7)$ with respect to both representations, especially within the Cartesian task space. The wrist related joint angles have less influence on the prediction error than the other joints and consequently XCSF focuses on joints $q_1, \ldots, q_4$ to increase its accuracy.

4.4 Analysis of an individual movement

The crude generalization raises the question if XCSF's model is accurate enough to control the wrist. In order to reach a particular location, it is often sufficient when shoulder and elbow are used to counter impreciseness in wrist control. Therefore, we measure individual joint angles for a single reaching task using a Cartesian task space and vary the angular target for the wrist. The initial arm configuration is set to $q = [0, 0, 0, 0, 0, 0, 0]^T$, which corresponds to a fully extended arm and a Cartesian task space location $t = [100\ ,\ 0\ ,\ 0\ \text{cm}]^T$. The task space goal is set to $[25\ ,\ 0\ ,\ -70\ \text{cm}]^T$. Initially, the default constraint is applied, that is, a centered configuration is preferred. Figure 9b shows that XCSF quickly reaches the desired target and maintains the task space location.

At time step $t = 500$, the constraint is slightly changed: Now wrist joint angle $q_6 = 1.4$, which is almost its maximum, is desired. Concerning the other joints, a centered position is preferred. Figure 9a illustrates the movement in configuration space. XCSF quickly navigates to the task space target that is reached after about 150 simulation steps. Upon change of the constraint, the redundancy is exploited to flex the wrist and the new constraint is fulfilled, while the task space accuracy is hardly affected.

To compare the results of the Cartesian task space with the egocentric representation, the Cartesian target is converted to egocentric coordinates, namely $[0.812\ ,\ -0.519,\ -0.123\ \text{cm}]^T$. Figure 10 validates the representational independence once again. With an egocentric task space, XCSF also reaches the target after about 150 simulation steps and the new constraint at $t = 500$ is flexibly realized using null-space movements as with the Cartesian representation. To sum up, the

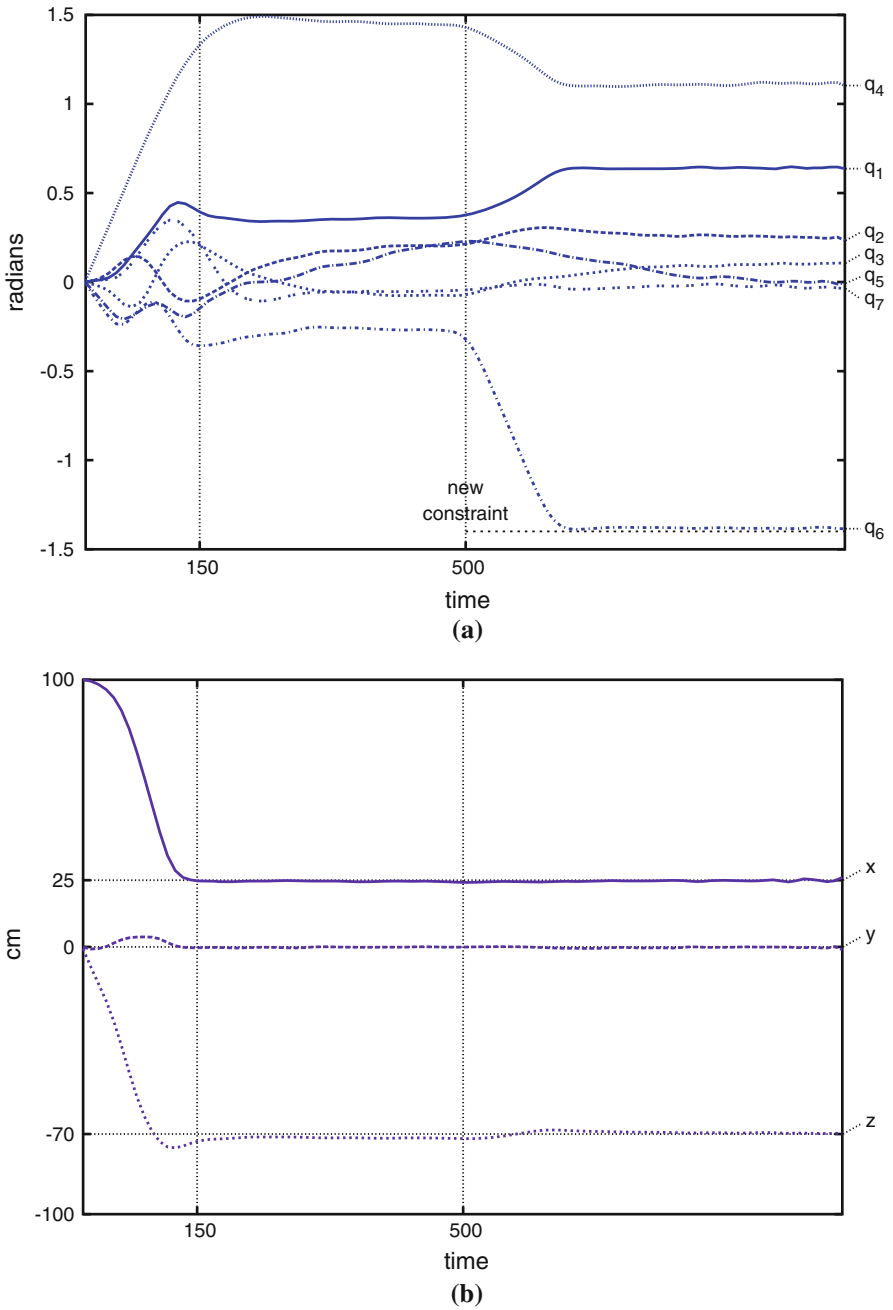
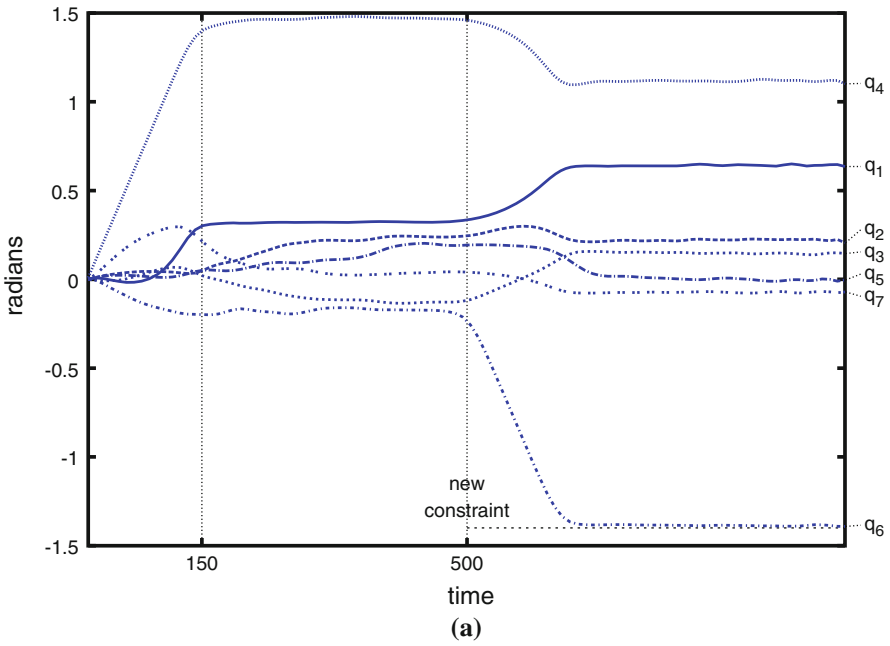


Fig. 9 After 100,000 learning steps with the Cartesian representation, a single movement is analyzed in depth. The task space goal is reached at $t \approx 150$, where the initial constraint aims at a centered position for each joint. Due to the range of the elbow joint (Fig. 3), the center for q_4 is at 1.4. At $t = 500$, the constraint is changed: Now, $q_6 = -1.4$ is desired. The task space location is hardly affected, while the null-space movement realizes $q_6 = -1.4$. **a** Joint angles, **b** Cartesian end effector location



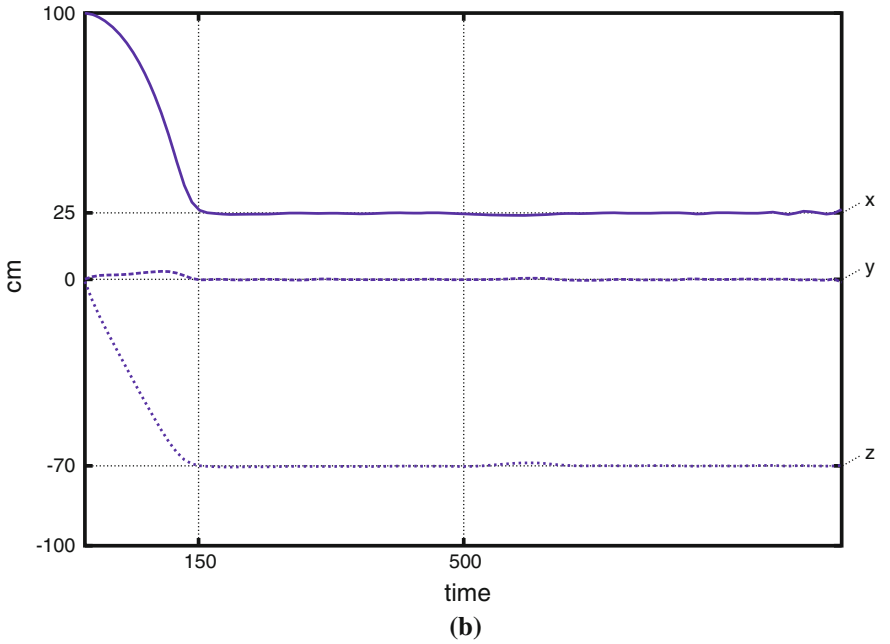


Fig. 10 Here the egocentric representation is used during learning and control. However, in (**b**) the *Cartesian* coordinates of the end effector are depicted to maintain comparability with Fig. 9b. Again, the target is reached at about $t \approx 150$, while the constraint is changed at $t = 500$. **a** Joint angles, **b** Cartesian end effector location

individual movements illustrate that the wrist is accurately controlled although the size of the RFs suggested impreciseness in those dimensions.

5 Summary and conclusions

We have presented a learning framework for autonomous learning and control at the kinematics level of a robot arm. In particular, the neuro-evolution algorithm XCSF was enhanced to learn a locally linear forward-inverse model of the velocity kinematics of robotic arms. XCSF learns a representation that preserves redundant control capabilities during learning. This knowledge can be easily exploited during goal-directed behavioral control due to its locally linear representation. Moreover, it was shown that the learner does not rely on any a priori controller but learns and controls the robot arm autonomously from "scratch". To achieve this, the system acts goal-directed from the beginning by means of goal babbling. Also, the model is adapted continuously online and can thus generally cope with changes in the controlled robotic device.

The framework was applied on an anthropomorphic, seven DoFs, three dimensional arm simulation. It was shown that XCSF quickly learned the full kinematic forward model and a simple control scheme based on XCSF's knowledge successfully reached arbitrarily generated task space goals. Due to its structuring and generalization capabilities, approximately 500 RFs in the final population, which covered the seven-dimensional joint angle space, sufficed to control the arm accurately, effectively, and flexibly: (a) The end-effector follows an approximately straight line to the target and reaches all targets. (b) The joint movements are minimized while approaching targets given no further constraints. (c) Additional constraints can be incorporated easily and on the fly—such as the avoidance of angular boundaries or the preference of a particular joint angle [10].

Moreover, no assumptions about the underlying coordinate system were made and consequently learning success was rather independent of the chosen configuration and task space representations. In particular, we compared a Cartesian task space representation with an egocentric, angular-distance representation and showed that the resulting behavioral capabilities were comparable. However, we also showed that XCSF learned a different spatial partitioning for the respective task space representations, where the partitioning was optimized for the development of an accurate forward model within the given representation. This insight strongly suggests that XCSF is a very general learner that can learn conditional forward models within various configuration and task spaces as well as for various types of robotic devices.

In conclusion, this paper has shown that the XCSF learning system can be applied to learn models for the control of robotic devices that preserve control alternatives during learning where available. The behavioral redundancy can be resolved on demand considering—possibly varying—additional task constraints. In comparison to the closely related LWPR system, XCSF has the advantage of being able to partition a contextual space (here the joint angle space) that does not need to be the same as the input space for the predictions (here joint angle velocities). As shown in various previous approaches, XCSF is furthermore not restricted to optimize Gaussian kernels, but can be generally applied for the optimization of any

kernel structure [8, 14, 36, 37]. Thus, XCSF is a more general learning system that, nonetheless, shows high learning reliability in various setups ranging from data mining and reinforcement learning problems to function approximation problems [2, 5, 8, 26, 31, 36].

Future research efforts should focus further on the applicability of the XCSF system to other robotic devices and setups, including real robots. For example, it is expected that also the dynamics of a device will be learnable by XCSF. Also, behavioral manifolds for imitation, such as the ones learned with LWPR [28], may be learned even more effectively and reliably with XCSF. Moreover, theoretical efforts should focus on the adaptability of the system as well as on the scalability in even higher dimensional problem spaces—within which possibly only a lower-dimensional manifold may be sampled. Most recent theoretical analyses suggest optimal system scalability at least in constrained problem settings [26]. Finally, the modularization of the XCSF system is expected to open up an even broader range of suitable application domains.

Acknowledgements The authors acknowledge funding from the Emmy Noether program (German Research Foundation, DFG, BU1335/3-1) and thank the COBOSLAB team as well as Olivier Sigaud and Vincent Padois from the ISIR lab at UPMC, Paris, for helpful comments and suggestions.

References

1. A. Ben-Israel, T.N. Greville, *Generalized Inverses: Theory and Applications*. (Springer, Berlin, 2003)
2. E. Bernadó-Mansilla, J.M. Garrell-Guiu, Accuracy-based learning classifier systems: models, analysis, and applications to classification tasks. Evol. Comput. **11**, 209–238 (2003)
3. N.E. Berthier, Learning to reach: a mathematical model. Dev. Psychol. **32**(5), 811–823 (1996)
4. N.E. Berthier, M.T. Rosenstein, A.G. Barto, Approximate optimal control as a model for motor learning. Psychol. Rev. **112**(2), 329–346 (2005)
5. M.V. Butz, *Rule-Based Evolutionary Online Learning Systems: A Principled Approach to LCS Analysis and Design*. (Springer, Berlin, 2006)
6. M.V. Butz, O. Herbort, *Context-dependent predictions and cognitive arm control with XCSF*. Genetic and Evolutionary Computation Conference, GECCO 2008 (2008), pp. 1357–1364
7. M.V. Butz, O. Herbort, J. Hoffmann, Exploiting redundancy for flexible behavior: unsupervised learning in a modular sensorimotor control architecture. Psychol. Rev. **114**, 1015–1046 (2007)
8. M.V. Butz, P.L. Lanzi, S.W. Wilson, Function approximation with XCS: hyperellipsoidal conditions, recursive least squares, and compaction. IEEE Trans. Evol. Comput. **12**, 355–376 (2008)
9. M.V. Butz, G.K. Pedersen, P.O. Stalph, in *GECCO '09: Proceedings of the 11th Annual Conference on Genetic and Evolutionary Computation*. Learning sensorimotor control structures with XCSF: redundancy exploitation and dynamic control (2009), pp. 1171–1178
10. M.V. Butz, G.K.M. Pedersen, P.O. Stalph, in *Learning sensorimotor control structures with XCSF: Redundancy exploitation and dynamic control*. Genetic and Evolutionary Computation Conference, GECCO 2009 (2009), pp. 1171–1178
11. A. D'Souza, S. Vijayakumar, S. Schaal, Learning inverse kinematics. IEEE Int. Conf. Intell. Robots Syst. **2001**(1), 298–303 (2001)
12. M. Haruno, D.M. Wolpert, M. Kawato, MOSAIC model for sensorimotor learning and control. Neural Comput. **13**(10), 2201–2220 (2001). doi:10.1162/089976601750541778
13. J.H. Holland, *Adaptation in Natural and Artificial Systems: An Introductory Analysis with Applications to Biology, Control, and Artificial Intelligence*. (The MIT Press, Cambridge, 1992)

14. J. Hurst, L. Bull, A neural learning classifier system with self-adaptive constructivism for mobile robot learning. Artif. Life **12**, 1–28 (2006)
15. M.I. Jordan, D.E. Rumelhart, Forward models: supervised learning with a distal teacher. Cogn. Sci. **16**(3), 307–354 (1992)
16. M. Kawato, K. Furukawa, R. Suzuki, A hierarchical neural-network model for control and learning of voluntary movement. Biol. Cybernet. **57**(3), 169–185 (1987)
17. P.L. Lanzi, D. Loiacono, S.W. Wilson, D.E. Goldberg, in *GECCO '06: Proceedings of the 8th Annual Conference on Genetic and Evolutionary Computation*. Prediction update algorithms for XCSF: RLS, Kalman filter, and gain adaptation (ACM, New York, 2006), pp. 1505–1512
18. A. Liegeois, Automatic supervisory control of the configuration and behavior of multibody mechnisms. IEEE Trans. Syst. Man Cybernet. **7**, 868–871 (1977)
19. A. Orriols-Puig, E. Bernadó-Mansilla, D.E. Goldberg, K. Sastry, P.L. Lanzi, Facetwise analysis of XCS for problems with class imbalances. IEEE Trans. Evol. Comput. **13**, 1093–1119 (2009). doi:10.1109/TEVC.2009.2019829
20. M. Rolf, J.J. Steil, M. Gienger, Goal babbling permits direct learning of inverse kinematics. Auton. Mental Dev. IEEE Trans. **2**(3), 216–229 (2010). doi:10.1109/TAMD.2010.2062511
21. D.A. Rosenbaum, S.E. Engelbrecht, M.M. Bushe, L.D. Loukopoulos, Knowledge model for selecting and producing reaching movements. J. Motor Behav. **25**(3), 217–227 (1993)
22. D.A. Rosenbaum, L.D. Loukopoulos, R.G.J. Meulenbroek, J. Vaughan, S.E. Engelbrecht, Planning reaches by evaluating stored postures. Psychol. Rev. **102**(1), 28–67 (1995)
23. D.A. Rosenbaum, R.G.J. Meulenbroek, J. Vaughan, C. Jansen, Posture-based motion planning: applications to grasping. Psychol. Rev. **108**(4), 709–734 (2001)
24. C. Salaün, V. Padois, O. Sigaud, in *Proceedings of the 2009 IEEE/RSJ International Conference on Intelligent Robots and Systems*. Control of redundant robots using learned models: an operational space control approach (2009), pp. 878–885
25. P.O. Stalph, M.V. Butz, G.K. Pedersen, in *Proceedings of the 32nd German Conference on Artificial Intelligence (KI-2009)*. Controlling a four degree of freedom arm in 3D using the XCSF learning classifier system (2009)
26. P.O Stalph, X. Llorà, D.E. Goldberg, M.V. Butz, Resource management and scalability of the XCSF learning classifier system. Theor. Comput. Sci. (in press). doi:10.1016/j.tcs.2010.07.007
27. P.O. Stalph, J. Rubinsztajn, O. Sigaud, M.V. Butz, in *GECCO '10: Proceedings of the 12th Annual Conference on Genetic and Evolutionary Computation*. A comparative study: function approximation with LWPR and XCSF (ACM, New York, 2010), pp. 1863–1870. doi:10.1145/1830761.1830818
28. S. Vijayakumar, A. D'Souza, S. Schaal, Incremental online learning in high dimensions. Neural Comput. **17**(12), 2602–2634 (2005)
29. S. Vijayakumar, S. Schaal, in *ICML '00: Proceedings of the Seventeenth International Conference on Machine Learning*. Locally weighted projection regression: an O(n) algorithm for incremental real time learning in high dimensional space (2000), pp. 1079–1086
30. D.E. Whitney, Resolved motion rate control of manipulators and human prostheses. IEEE Trans. Man Mach. Syst. **10**(2), 47–53 (1969)
31. S.W. Wilson, Classifier fitness based on accuracy. Evol. Comput. **3**(2), 149–175 (1995)
32. S.W. Wilson, in *Genetic Programming 1998: Proceedings of the Third Annual Conference*. Generalization in the XCS classifier system (1998), pp. 665–674
33. S.W. Wilson, in *Learning Classifier Systems, From Foundations to Applications*. Get real! XCS with continuous-valued inputs (Springer, Berlin, 2000), pp. 209–219
34. S.W. Wilson, in *Learning Classifier Systems, From Foundations to Applications*. State of XCS classifier system research (Springer, London, 2000), pp. 63–82
35. S.W. Wilson, in *GECCO '01: Proceedings of the 2001 Conference on Genetic and Evolutionary Computation*. Function approximation with a classifier system (Morgan Kaufmann, Los Altos, 2001), pp. 974–981
36. S.W. Wilson, Classifiers that approximate functions. Nat. Comput. **1**, 211–234 (2002)
37. S.W. Wilson, in *Learning Classifier Systems: 10th International Workshop, IWLCS 2006, Seattle, MA, USA, July 2006 and 11th International Workshop, IWLCS 2007, London, UK, July 2007, Revised Selected Papers, LNAI 4998* ed. by J. Bacardit, E. Bernadó-Mansilla, M.V. Butz, T. Kovacs, X. Llorà, K. Takadama. Classifier conditions using gene expression programming (Springer, Berlin, 2008), pp. 206–217
38. D.M. Wolpert, M. Kawato, Multiple paired forward and inverse models for motor control. Neural Netw. **11**, 1317–1329 (1998)