Filtering Sensory Information with XCSF: Improving Learning Robustness and Robot Arm Control Performance

Jan Kneissler jan.kneissler@uni-tuebingen.de
Department of Computer Science, University of Tübingen,
Sand 14, 72076 Tübingen, Germany

Patrick O. Stalph patrick.stalph@uni-tuebingen.de
Department of Computer Science, University of Tübingen,
Sand 14, 72076 Tübingen, Germany

Jan Drugowitsch jan.drugowitsch@ens.fr
Départment d'Etudes Cognitives, Ecole Normale Supèrieure,
29 rue d'Ulm, 75005 Paris, France

Martin V. Butz martin.butz@uni-tuebingen.de
Department of Computer Science, University of Tübingen,
Sand 14, 72076 Tübingen, Germany

doi:10.1162/EVCO_a_00108

Abstract

It has been shown previously that the control of a robot arm can be efficiently learned using the XCSF learning classifier system, which is a nonlinear regression system based on evolutionary computation. So far, however, the predictive knowledge about how actual motor activity changes the state of the arm system has not been exploited. In this paper, we utilize the forward velocity kinematics knowledge of XCSF to alleviate the negative effect of noisy sensors for successful learning and control. We incorporate Kalman filtering for estimating successive arm positions, iteratively combining sensory readings with XCSF-based predictions of hand position changes over time. The filtered arm position is used to improve both trajectory planning and further learning of the forward velocity kinematics. We test the approach on a simulated kinematic robot arm model. The results show that the combination can improve learning and control performance significantly. However, it also shows that variance estimates of XCSF prediction may be underestimated, in which case self-delusional spiraling effects can hinder effective learning. Thus, we introduce a heuristic parameter, which can be motivated by theory, and which limits the influence of XCSF's predictions on its own further learning input. As a result, we obtain drastic improvements in noise tolerance, allowing the system to cope with more than 10 times higher noise levels.

Keywords

Noise, robotics, function approximation, learning classifier systems, XCSF.

1 Introduction

Controlling a robot arm such that goals can be reached efficiently by the robot's hand (end-effector) requires an understanding of how motor commands affect the end-effector position. Thus, the relationship between changes in limb angles to changes

Manuscript received: October 31, 2012; revised: March 25, 2013 and May 25, 2013; accepted: June 3, 2013.

in end-effector positions (the so-called velocity kinematics) has to be known. Assuming that the robot controller has no prior knowledge of the arm's geometry, it has to learn these mappings using simultaneous observations of limb angles and the end-effector position. Stereo-vision sensors provide a 3D measurement of the position of the robot's hand, but with a relatively high degree of uncertainty. As we will see, the noise introduced by end-effector position measurements may severely disrupt learning at moderate noise levels. Thus, any successful real-world learning approach needs to provide strategies to cope with this noise.

It is known that in a noise-free scenario, the velocity kinematics can be learned very successfully using XCSF, an evolutionary system of a population of linear regressors. The aim of this paper is to improve the noise robustness of an XCSF robot controller using Kalman filtering. The key idea is that the estimation of the state change, which is required in the Kalman prediction step, can be obtained using the velocity kinematics mappings stored in XCSF.

1.1 Robot Arm Control with XCSF

Stalph and Butz (2012) have developed a system that successfully learns to control robot arms with up to seven degrees of freedom, using an evolutionary function approximation system called XCSF. XCSF can be described as a derivative (Wilson, 2002) of the XCS learning classifier system (Wilson, 1995). It approximates nonlinear functions by a population of locally-weighted linear local approximators. Each approximator is usually referred to as a classifier whose condition determines its activity and thus its local influence given a current input. The classifier prediction is a linear approximation of the encountered input-output value combinations.

It was shown recently (Butz et al., 2008b; Stalph et al., 2010) that XCSF shares many similarities with the locally-weighted projection regression (LWPR) algorithm (Vijayakumar et al., 2005), which is well-known in the neuro-robotics literature (Peters and Schaal, 2008; Sigaud and Peters, 2010; Sigaud et al., 2011). With this interpretation in mind, XCSF classifiers may be considered as neural structures that specify local receptive fields.

1.2 Exploiting Model Knowledge to Improve Noise Robustness

XCSF learns the forward velocity kinematics (mapping changes in joint angles to changes in hand locations given the current arm configuration) from which the inverse velocity kinematics (mapping desired changes in hand location space to changes in joint angles) is constructed in order to generate targeted movement commands. The forward knowledge itself, however, has not been exploited in any way so far.

An obvious approach to increase noise robustness is to exploit temporal dependencies: since the observations are made along trajectories of the robot arm, subsequent samples are highly correlated. There is thus some potential for significant gains in the signal to noise ratio by exploiting these correlations by means of filtering. It is necessary to use adaptive filters, because fixed filters with a constant transfer function tend to introduce temporal delay, which introduces systematic biases in learning.

We propose to use a well-established approach, known as Kalman filtering, which assumes knowledge of the system's dynamics. A Kalman filter operates recursively on a stream of noisy input data to produce a statistically optimal estimate of the underlying system state. Each iteration can be split into a prediction step and an update step. In the prediction step, the system's next state is predicted from its presumed state in the previous time step and known control inputs. In the update step, the new system state estimate is obtained by fusing information obtained from sensors with the predicted state.

The crucial contribution of this paper is to use forward kinematic velocity estimates provided by XCSF as control input for the Kalman filter. This requires an estimate of the prediction variance, as investigated in Drugowitsch and Barry (2007, 2008) and Loiacono et al. (2008). When implementing the Kalman filter without any precautions, we show that the system may learn from over-filtered sensory information and thus may get stuck in a self-delusional loop, where XCSF over-believes its own predictions. By effectively limiting the estimated prediction confidence, however, we can show that the resulting XCSF system can deal with much larger measurement noise, still learning its bidirectional forward prediction and inverse control structures effectively.

1.3 Paper Organization

We first give an overview of XCSF and how it is applied to learn arm control. Next we introduce the basics of Kalman filtering. Then, we specify the modification of the interaction cycle necessary to exploit temporal dependencies of samples in the arm control scenario by means of Kalman filtering. We scrutinize the performance of XCSF on two simulated arm models with 2 DOF and 7 DOF, respectively.

A key finding, which was already reported at the GECCO 2012 conference (Kneissler et al., 2012) in a shorter version of this paper, is that XCSF with Kalman filtering can solve problems at a higher magnitude of sensory noise, but only after introduction of an additional threshold parameter. Here we additionally report the results of follow-up experiments that confirmed the self-delusional effect as the reason for failure without thresholding and find consistency with theoretical considerations, which provides a formula to calculate the (so-far manually optimized) threshold value as a function of the noise level. Finally, we investigate the possible sources for the failure of the baseline XCSF-based arm control under noisy conditions separately and find that the evolutionary algorithm of XCSF is the most vulnerable spot of the system. Summary and future work considerations conclude the paper.

2 Robot Arm Control Using XCSF

2.1 Prior Work

Learning classifier systems (LCSs) were originally introduced by John H. Holland (Holland and Reitman, 1978). The accuracy-based XCS learning classifier system was introduced by Stewart W. Wilson (Wilson, 1995). XCS was successfully applied in binary classification tasks (Wilson, 1995; Butz, 2006), data mining problems (Bernadó-Mansilla and Garrell-Guiu, 2003; Wilson, 2000), and function approximation problems (Wilson, 2001, 2002; Lanzi et al., 2007; Butz, Lanzi, et al., 2008), among others (Bull, 2004). In the real-valued function approximation case, XCS is termed XCSF, essentially constituting an iteratively learning regression system.

We focus on the XCSF system (Wilson, 2002; Butz, Lanzi, et al., 2008) and its potential for learning the forward velocity kinematics of a robot arm for inverse control (Butz and Herbort, 2008; Butz and Stalph, 2011; Stalph and Butz, 2012). The considered XCSF setup is based on the available implementation (Stalph and Butz, 2009) and the detailed descriptions available in the literature (Butz and Herbort, 2008; Butz, Lanzi, et al., 2008; Stalph and Butz, 2012).

2.2 Learning Forward Velocity Kinematics

In general, XCSF learns to approximate nonlinear multidimensional functions using piecewise linear models. It evolves a population of classifiers, where each classifier covers a particular subspace of the input space, which may be termed the receptive field of a classifier. Moreover, each classifier learns a linear model for approximating the function surface in its subspace. The linear model of a classifier is typically adapted by means of recursive least squares (Butz, Lanzi, et al., 2008; Lanzi et al., 2006). The receptive field, that is, the condition part of a classifier, is evolved over time by a steady-state genetic algorithm (GA). We use general Gaussian kernel-based receptive fields, yielding ellipsoidal regions of influence for a classifier. While each single classifier approximates a subset, the whole population approximates the complete function surface. Due to evolutionary optimization, XCSF tends to implicitly minimize the absolute difference between this surface and the encountered function (however, there is actually no such explicit goal encoded algorithmically in XCSF).

XCSF can be used to learn the velocity kinematics of a robot arm. A robot arm may be characterized by its configuration space $\mathcal{C} \subset \mathbb{R}^n$ and the task space $\mathcal{T} \subset \mathbb{R}^m$ of the end-effector. While the task space is usually encoded in a Cartesian coordinate system, the configuration space may be encoded by joint angles. Due to the arm kinematics, a configuration $\mathbf{q} \in \mathcal{C}$ uniquely determines the corresponding end-effector location in task space $\boldsymbol{x} \in \mathcal{T}$. This forward kinematics mapping can be expressed as a typically nonlinear function $\boldsymbol{x} = f(\mathbf{q})$. The forward velocity kinematics is obtained by taking the derivative $\dot{\boldsymbol{x}} = \mathbf{J}(\mathbf{q})\dot{\mathbf{q}}$, where $\mathbf{J}(\mathbf{q}) = \partial f / \partial \mathbf{q}$ is the $m \times n$ Jacobian matrix of f.

To control the robot arm, a goal direction $\dot{\boldsymbol{x}}$ needs to be translated into necessary control commands $\dot{\mathbf{q}}$ by applying the inverse velocity kinematics. Here lies the strength of XCSF: Due to the inherent ambiguity of the inverse kinematics, it is much easier to learn the forward kinematics instead. Since XCSF models the target function locally linear, the local Jacobian $\mathbf{J}(\mathbf{q})$, given the specified configuration $\mathbf{q}$, can be extracted from the population of XCSF classifiers at almost no computational cost.

It remains to invert $\mathbf{J}(\mathbf{q})$, resolving redundancy by introducing additional constraints. Here, we invert the Jacobian such that free DOFs are used to avoid extreme angular values. However, it was shown elsewhere (Butz et al., 2009; Stalph and Butz, 2012) that other redundancy resolution techniques can equally well be applied, such as applying additional movement constraints or computing the Moore-Penrose pseudo-inverse (Ben-Israel and Greville, 2003).

To sum up, XCSF learns the mapping from configuration space velocities $\dot{\mathbf{q}}$ to task space velocities $\dot{\boldsymbol{x}}$, depending on the current configuration $\mathbf{q}$. XCSF is well-suited for this 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 are the configuration space with hyper-ellipsoidal RFs (Butz, Lanzi, et al., 2008). In turn, each classifier approximates the $\dot{\boldsymbol{x}} \mapsto \dot{\boldsymbol{q}}$ mapping by estimating the local Jacobian matrix $J(\boldsymbol{q})$ using linear recursive least squares approximations. Given a particular configuration, the classifiers whose conditions overlap with the current configuration are considered; their linear Jacobians are combined in a weighted manner; finally, the resulting Jacobian matrix is inverted for generating a control signal.

3 Exploiting Temporal Dependencies

3.1 Kalman Filtering

Filtering can be seen as a procedure for transforming a noisy time series $(\tilde{x}_n)$ to a smoothed time series $(\bar{x}_n)$ closer to the sequence of "true" values (x_n), in our case the sequence of end-effector locations that are traversed in task space. Kalman filtering does that by elegantly combining the following two sources of information:

1. The filtered value $\bar{x}_{n\text{-}1}$ from the previous time step, corrected by the predicted change between steps with mean Δx_n^{p} and variance $\sigma_{n\text{-}1}^2$ (formed from the predictions of the individual classifiers, as we will describe in a later section), and
2. The sensor readings $\tilde{x}_n$ for the current time step.

At each time step n, the Kalman filter maintains an estimate of the end-effector location that is given by its mean $\bar{x}_n$ and its variance V_n. Assuming knowledge of this estimate at time $n-1$, it is updated in the light of the predicted change and the current sensory reading in two steps.

Prediction step:

$$\bar{x}_{n|n\text{-}1} = x_{n\text{-}1} + \Delta x_{n\text{-}1}^{\mathrm{p}} \tag{1}$$

$$V_{n|n\text{-}1} = V_{n\text{-}1} + \sigma_{n\text{-}1}^2 \tag{2}$$

This step predicts the end-effector state at time n without taking into account the sensory readings $\tilde{x}_n$.

Update step:

$$\bar{x}_n = \bar{x}_{n|n\text{-}1} + \beta\,(\tilde{x}_n - \bar{x}_{n|n\text{-}1}) \tag{3}$$

$$V_n = (1-\beta)\,V_{n|n\text{-}1} \tag{4}$$

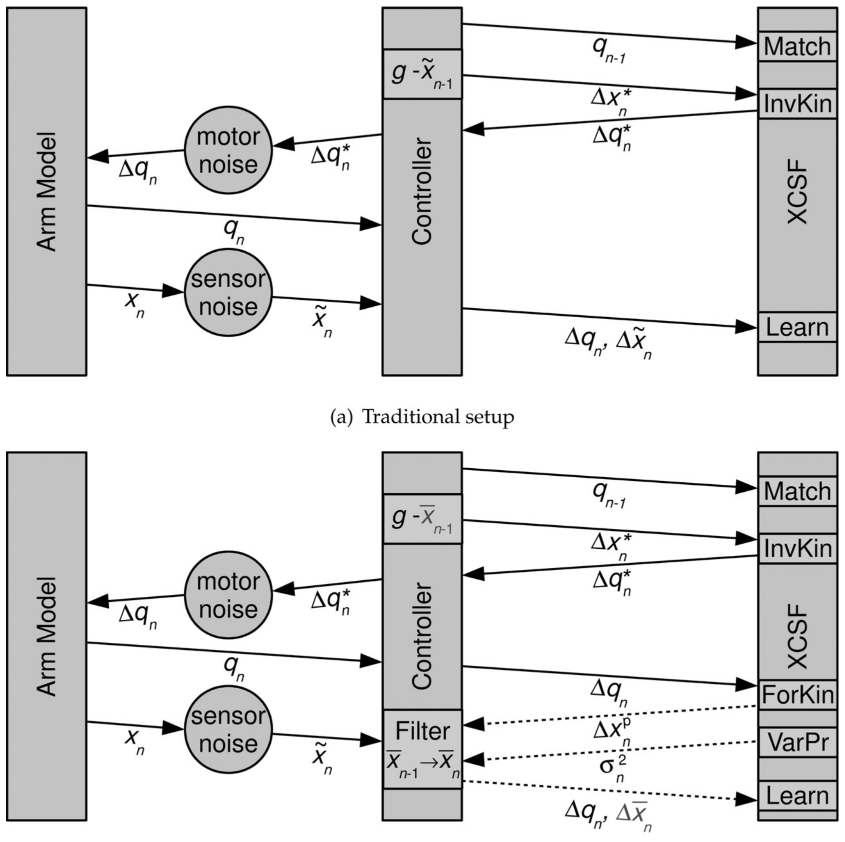
This step mixes sensory readings and the predicted internal state estimation, using the smoothing parameter β, which is specified by weighing the respective information contents:

$$\beta = \frac{V_{n|n\text{-}1}}{V_{n|n\text{-}1} + \sigma_{\mathrm{sensor}}^2}, \tag{5}$$

where σ_{sensor} denotes the standard deviation of the sensory noise. For $\beta = 1$, there is no filtering at all, while for $\beta = 0$, the new measurement is completely ignored. If the variance of several consecutive sensor variance estimates $\sigma_{n\text{-}k}^2, \ldots, \sigma_n^2$ are all very small, the variance estimate V_n converges quickly toward a value close to 0.

3.2 Kalman Filtering with XCSF

The overall flow of information in an XCSF-based arm control learner without filtering is shown in Figure 1(a). It is consistent with previous experiments, as conducted, for example, by Butz and Herbort (2008); Butz et al. (2009); and Stalph and Butz (2012). XCSF interacts with a controller module, which reads sensory information from and executes motor commands in the associated arm model. Motor noise and sensor noise may be added in this process. Given the current joint angle state of the arm $q_{n\text{-}1}$, XCSF generates a set of matching classifiers. Using this set and a given desired direction of the hand $\Delta x_n^* = g - \tilde{x}_{n\text{-}1}$ toward goal g, XCSF generates a control command Δq_n^* by means of its locally linear, inverse velocity kinematics model. In our kinematic arm simulations, motor noise is added to this motor command, which is then sent to the arm model. In return, the next angular state q_n is perceived. Additionally, the consequent location of the end-effector x_n is perceived and noise is added, yielding $\tilde{x}_n$. The information is thus complete to learn from the described interaction using $q_{n\text{-}1}$ as the context signal for matching, and $\Delta q_n = q_n - q_{n\text{-}1}$ as well as $\Delta\tilde{x}_n = \tilde{x}_n - \tilde{x}_{n\text{-}1}$ for the prediction and resulting error and fitness updates.



(a) Traditional setup

(b) With Kalman filtering

Figure 1: Setups for XCSF-based arm control and learning: traditional and modified with Kalman filtering to improve robustness of XCSF during learning and arm control against noise on the hand position sensors.

The described original XCSF setup for learning and goal-directed behavior control is modified as shown in Figure 1(b). The added Kalman filtering process essentially filters the successive perceptions of end-effector locations $\tilde{x}_n$ by means of XCSF's forward velocity kinematics model. Previously, the change in end-effector state was directly determined by the noisy location signal, that is, $\Delta\tilde{x}_n = \tilde{x}_n - \tilde{x}_{n\text{-}1}$. Now the change in the end-effector location is calculated by filtering the location feedback $\tilde{x}_n$ with the state prediction generated by the locally linear forward velocity kinematics model of XCSF, that is, $\bar{x}_{n\text{-}1} + \Delta x^P_n$, additionally considering the variance estimate σ^2_n of the local model. In consequence, XCSF can exploit its predictive knowledge controlling the arm based on the filtered internal hand location estimate $\bar{x}_n$ and learning from the filtered location changes $\Delta\bar{x}_n$ in each iteration n. Thus, we expect to reap the benefits of the filtering process for both learning and movement planning.

The dashed arrows in Figure 1(b) indicate a feedback loop that potentially leads to an increasing overconfidence of the system in its own predictions. To understand why, it has to be remembered that the prediction and prediction variance estimates of XCSF are always in flux during learning. Thus, these estimates need to be handled with care. In particular, it may well happen that XCSF temporarily overestimates its confidence in its own predictions. If this occurs, the system may enter a vicious cycle: due to the low variance estimates in the predictions, actual sensory feedback may be completely over-ruled. In effect, learning may stall and XCSF may solely rely on its own predictions—consequently preventing further learning. To avoid this effect, we introduce a threshold θ_β, which constitutes a lower bound on β, and then Equation (5) becomes

$$\beta = \max\left(\theta_\beta, \frac{V_{n|n\text{-}1}}{V_{n|n\text{-}1} + \sigma^2_{\text{sensor}}}\right). \tag{6}$$

Note that for $\theta_\beta = 0$, Equation (6) becomes equivalent to Equation (5).

3.3 Error and Confidence Estimation

XCSF traditionally estimates the mean absolute deviation of its prediction. However, a variance estimate is necessary to combine the prediction with other, for example, sensory information. Thus, XCSF is modified by estimating the standard deviation of a classifier prediction, rather than the mean absolute deviation. This was proposed and investigated elsewhere for the XCSF system in function approximation problems (Loiacono et al., 2008), where it was shown that the performance of XCSF generally does not suffer from this system change.

In general, given k samples $(\dot{\boldsymbol{q}}_i, \dot{\boldsymbol{x}}_i)_{i=1\ldots k}$ that a classifier has learned from, the variance of the error of a single classifier in XCSF may be estimated by using the sequence of values $\dot{\boldsymbol{x}}_i$ and respectively generated predictions $\dot{\boldsymbol{x}}_i^{\text{p}}$

$$\hat{\sigma}_k^2 = \frac{1}{k}\sum_{i=1}^{k}\left(\dot{\boldsymbol{x}}_i - \dot{\boldsymbol{x}}_i^{\text{p}}\right)^2. \tag{7}$$

Note that $\dot{x}_i^{\text{p}}$ specifies the prediction that was generated when the sample was encountered. It is thus based on the knowledge of the classifier available until that point in time. A disadvantage of this approach is that the estimate tends to be pessimistic, in that the true σ_k^2 is likely to be smaller because the improvement of the linear model due to later samples is not taken into account. The advantage, however, is that it is not necessary to calculate the prediction using all previous samples in every time step. In consequence, it is not required to keep a history of the previous samples. Rather, the variance estimate can be updated iteratively online:

$$\hat{\sigma}_{k+1}^2 = \frac{k\,\hat{\sigma}_k^2 + \left(\dot{\boldsymbol{x}}_{k+1} - \dot{\boldsymbol{x}}_{k+1}^{\text{p}}\right)^2}{k+1}. \tag{8}$$

3.4 Combining Stochastic Estimates across Classifiers

XCSF predicts function values as weighted linear combinations of predictions of the subset of classifiers that match the current context. In contrast to traditional XCSF, where these weights are proportional to the classifiers' fitness, we use the inverses of the variance estimates as weights, as proposed by Drugowitsch and Barry (2007). The joint error variance for its prediction, as used in the Kalman filter, is then computed as the weighted sum, such that highly accurate classifiers contribute more to the final prediction than inaccurate classifiers.

Application of the Kalman filter requires us to estimate the mean and variance of the change in task space, $\dot{\mathbf{x}}$, given a particular executed control $\dot{\mathbf{q}}$. While we have such an estimate for each classifier separately, we here describe how to combine these local estimates to provide a single, global estimate.

First, let us recapitulate a well-known methodology for extracting a higher fidelity signal by linear combination of several low-fidelity input signals. Let us therefore assume that we are given n random variables $(X_i)_{i=1\ldots n}$ (the estimate of each classifier i) that all share the same mean $\langle X_i \rangle = \bar{x}$ and have variances $V(X_i) = \sigma_i^2$. We further assume that the variances σ_i^2 are known to us, but the mean $\bar{x}$ is not, and shall therefore be estimated by a linear combination $\hat{x} = \sum w_i\, X_i / \sum w_i$, with some suitable weights w_i, with the hope that $\hat{x}$ gives us a better (i.e., lower variance) representation of the true value $\bar{x}$.

It is obvious that $\langle \hat{x} \rangle = \bar{x}$, but we can also calculate the variance of $\hat{x}$ in the extreme case of independent inputs

$$V_{\text{ind}}(\hat{x}) = \frac{\sum w_i^2\, \sigma_i^2}{\left(\sum w_i\right)^2}. \tag{9}$$

In cases where the independence requirement does not hold, we propose to use the weighted average of the variances instead (it can be shown that this always is an upper bound on the value of the true combined variance):

$$V_{\text{avg}}(\hat{x}) = \frac{\sum w_i\, \sigma_i^2}{\sum w_i}. \tag{10}$$

A natural choice is to make these weights proportional to those used to combine the classifier predictions. As described at the beginning of this section, the latter are chosen to be inversely proportional to the classifier's variance estimates, such that $w_i = \frac{1}{\sigma_i^2}$. As a result, the two formulas become identical up to an additional factor n:

$$V_{\text{ind}}(\hat{x}) = \frac{1}{\sum \frac{1}{\sigma_i^2}} \tag{11}$$

$$V_{\text{avg}}(\hat{x}) = \frac{n}{\sum \frac{1}{\sigma_i^2}} \tag{12}$$

We have chosen to use the more conservative approach of Equation (12) in our simulations.

4 Results

To evaluate the approach and scrutinize the dependency on the threshold θ_β, which controls the maximum influence of XCSF's predictive knowledge as detailed above, we tested performance on an exemplary 2 DOF robot arm system operating in a plane, as well as an anthropomorphic 7 DOF robot arm in three dimensions. All the reported results were carried out on a kinematic arm simulation.

4.1 XCSF Settings

In all experiments reported in this paper, we used the XCSF setup that was reported in Butz, Lanzi, et al. (2008). In particular, we use XCSF with hyper-ellipsoidal conditions, that is, general Gaussian kernels, which were first introduced in Butz et al. (2006). Moreover, each classifier learns a linear prediction by means of recursive least squares approximation. In particular, the parameters of XCSF were set as follows: $N = 2{,}000$, $\alpha = 1$, $\beta = 0.1$, $\delta = 0.1$, $\nu = 5$, $\chi = 1$, $\theta_{\text{del}} = 20$, $\theta_{\text{sub}} = 20$. The GA application threshold is set to $\theta_{\text{GA}} = 200$ and the target error is set to $\varepsilon_0 = 10^{-4}$. The mutation probability for each element of the condition is set to one over the number of alleles, for example

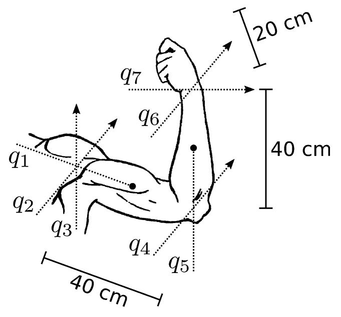


Figure 2: The anthropomorphic arm setup with seven DOFs.

$\mu = 1/5$ for the two-dimensional case and $\mu = 1/35$ for the 7 DOF arm. If mutated, center values are mutated within the receptive field bounds, the stretches are decreased or increased in size, at a maximum, doubling or halving their current size. The initial radius of receptive fields is taken uniformly random from $[0.01, 1]$. Uniform crossover, GA subsumption, and tournament selection with $\tau = 0.4$ are applied. Condensation (Wilson, 1998) commences after 80% of the learning iterations.

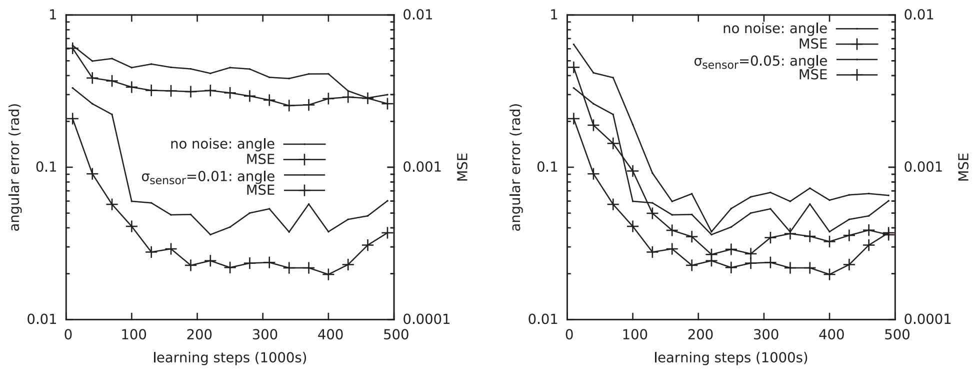
4.2 Experimental Setup

The planar 2 DOF arm with limb lengths $l_1 = 1.4$, $l_2 = 0.9$ and angular ranges $q_1, q_2 \in [-\pi, \pi]$ was used to explore the potential of the filtering approach. The maximum rotation velocity per joint is restricted to 0.01 radians per iteration. The kinematic specifications of the 7 DOF anthropomorphic arm are illustrated in Figure 2. 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 $q_1 \in [-1.0, 2.9]$, $q_2 \in [-1.5, 1.5]$, $q_3 \in [-1.0, 1.0]$, $q_4 \in [-0.0, 2.8]$, $q_5 \in [-0.7, 1.0]$, $q_6 \in [-1.5, 1.5]$, $q_7 \in [-0.5, 0.7]$. Similar to a human arm, the shoulder has 3 DOFs, the elbow allows for 2 DOFs, and the wrist offers another 2 DOFs.

During learning, the involved controller generates goal locations uniformly randomly within the workspace of the arm. From the beginning, XCSF utilizes its internal model to strive for the given goal. A goal is assumed to be reached if the arm end-effector reaches a distance smaller than 5% of its arm length to the goal. If the goal is reached or 300 iterations have passed without reaching the goal, a new goal location was generated. In this way, well-balanced exploration was achieved, as proposed elsewhere (Stalph and Butz, 2012). Moreover, 0.05 standard deviations in radians of angular motor noise were added to the motor commands, as also done elsewhere (Stalph and Butz, 2012). This is particularly necessary during the early stage to bootstrap XCSF's learning progress.

4.3 Quality Criteria

To track learning performance, offline testing phases were inserted. In an offline testing phase, XCSF-learning was switched off, but filtering was performed as usual. Each testing phase contained 100 episodes with randomly chosen start/goal-pairs (the same in all testing phases, disjoint to the start/goal-pairs used in learning). The Kalman filter was set to its initial state (infinite variance) at the beginning of each episode. When the distance of the hand position to the current goal was less than 0.05, the episode was considered successful. As during learning, if the goal is not reached within 300 time



(a) Baseline (b) Kalman Filtering

Figure 3: Learning progress (angular error and mean squared error) during the course of a full training run. (a) XCSF without filtering at noise levels $\sigma_{\text{sensor}} = 0$ and 0.01; noise deteriorates performance. (b) Kalman filtering with optimal β bounds at a noise level $\sigma_{\text{sensor}} = 0$ and 0.05; performance only is only mildly deteriorated due to noise.

steps, the episode was aborted. During this testing period, we evaluated the following two measures of the system's path planning capability:

1. **Quality.** The percentage of successful episodes was multiplied with the average path efficiency in order to obtain a performance measure in the range $0 \ldots 100$, where 0 indicates complete failure and 100 stands for perfect planning. The path efficiency of an episode is defined as the straight distance from start point to end point of the trajectory divided by its actual length.

2. **Angular Error.** The average angle (in radians) between the direction toward the target and the actually performed change in hand position (averaged over all time steps of all episodes). Since the measure includes the effect of motor noise, the theoretically optimal value of 0 is, in practice, never obtained.

4.4 Learning curves

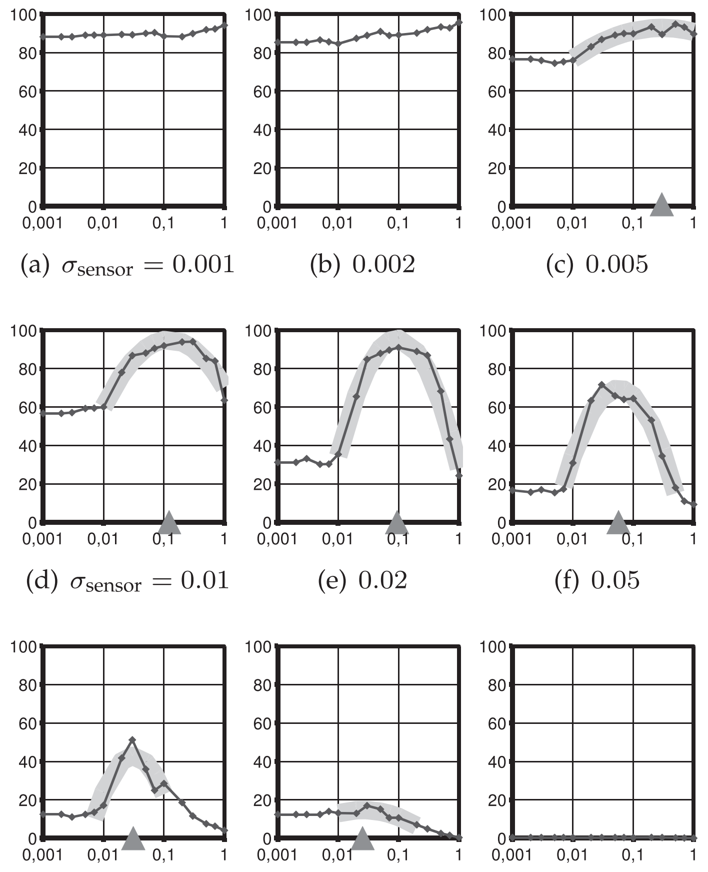
In Figure 3, we give the time course of XCSF performance for typical cases. Figure 3(a) is the baseline case without filtering. At a noise level of 0.01, learning of the baseline system already fails completely. With Kalman filtering (Figure 3(b)) and a suitably chosen θ_β (see Section 4.5), XCSF still learns well, even when the noise level is increased by a factor of 5.

Performances of different setups are subsequently compared by averaging the quality measures in the offline phases between 300k and 400k learning iterations (the final phase of an XCSF run is a condensation phase, starting at iteration 400k, during which redundant classifiers are eliminated).

4.5 Effect of Threshold Parameter $\boldsymbol{\theta_\beta}$

4.5.1 2-DOF Arm

We systematically measured the effect of the proposed threshold θ_β on performance for different levels of position sensor noise. In the plots of Figure 4, the quality measure (averaged over 20 runs) is shown as a function of θ_β for different levels of sensor noise σ_{sensor}. We find that if the noise is almost negligible, values of θ_β close to 1 are



(a) $\sigma_{\text{sensor}} = 0.001$ (b) 0.002 (c) 0.005

(d) $\sigma_{\text{sensor}} = 0.01$ (e) 0.02 (f) 0.05

(g) $\sigma_{\text{sensor}} = 0.1$ (h) 0.2 (i) 0.5

Figure 4: The quality measure (vertical axis) is plotted against parameter θ_β (horizontal axis) for different noise levels σ_{sensor}. (d)–(h) Quadratic fit to selected data points is shown as shaded area; resulting optimal θ_β values are marked with triangles on the x axis.

advantageous. For higher levels of sensor noise, however, there is a clearly distinguishable optimum (triangles on the x-axis, determined by a quadratic fit to the data points in the shaded area), which moves gradually to smaller values as σ_{sensor} increases. The height of the hill gradually decreases, and beyond a noise level of 0.2, the performance finally completely collapses for all choices of θ_β.

In effect, these results show that XCSF benefits from Kalman filtering with an appropriate θ_β threshold, but may also suffer from delusional cycles, in which it starts to believe too strongly in its own, inaccurate predictions (small θ_β values in Figure 4). The higher the actual noise of the sensors, the higher the need to prevent the system from believing too strongly in its own predictions.

The results show that the introduction of a threshold in Kalman filtering is helpful for a wide range of sensor noise levels. More than 20% SD of the actual arm length noise must be added to the location sensor to deduce a change in performance.

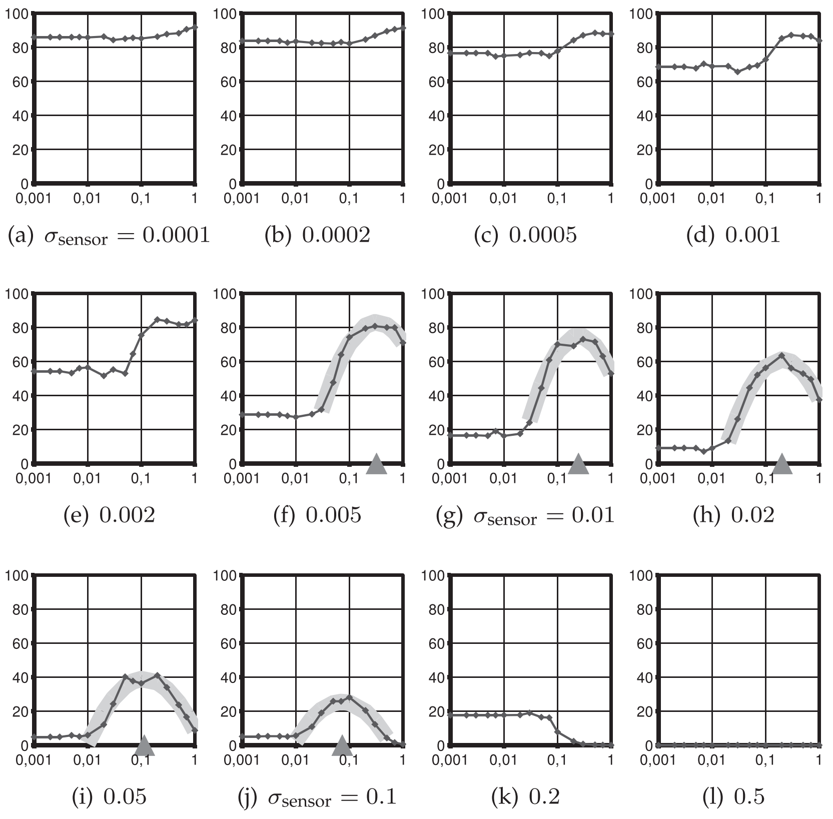


Figure 5: The quality measure on the 7-DOF arm is plotted against parameter θ_β for different noise levels σ_{sensor}. (f)–(j) Quadratic fit to selected data points is shown as shaded area; resulting optimal θ_β values are marked with triangles on the x axis.

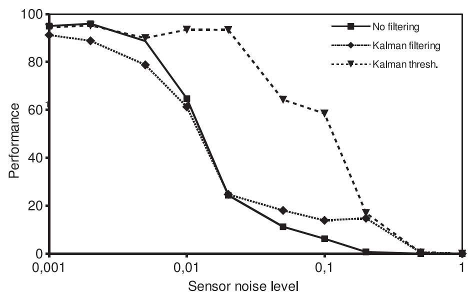
4.5.2 7-DOF Arm

While it is comparatively simple to reach high precision for a two-joint planar arm, the 7D task generally shows lower baseline performance, since the task generally scales exponentially in the dimension of the input space. Nonetheless, the XCSF learning classifier system is able to learn a suitable representation sufficient for accurate control.

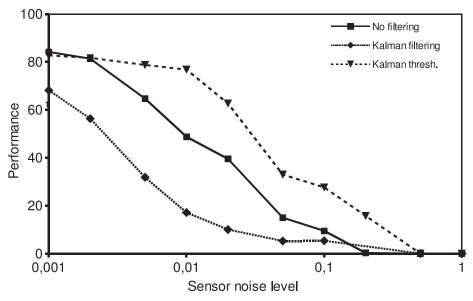
The result of an exhaustive search for optimal threshold parameters θ_β in Figure 5 is qualitatively very comparable to the 2-DOF case: For very small sensor noise $\sigma_{\text{sensor}} \leq 0.005$ in Figure 5(a) through Figure 5(c), the filtering approach cannot improve performance, which is confirmed by an optimal $\theta_\beta = 1$. With increasing sensor noise, the hill of optimal θ_β shifts to the left until performance collapses for noise levels beyond 0.1 in Figure 5(k) through Figure 5(l).

4.6 Noise Robustness

Finally, in Figure 6, we have assembled the summary of all experiments so far in a pair of graphs. They show the quality measure for different noise levels for the baseline system as well as the Kalman filtering systems with and without threshold θ_β (optimized according to Figures 4 and 5). It can be seen that pure Kalman filtering

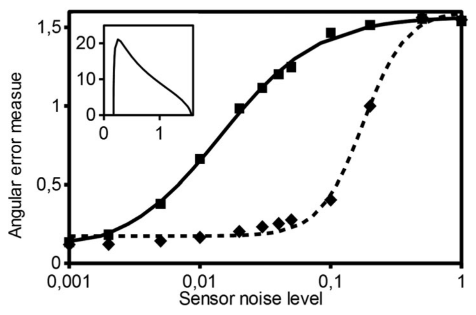


(a) 2-DoF

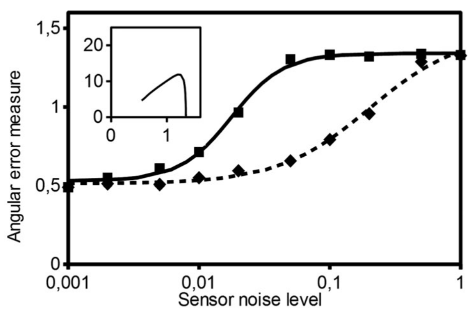


(b) 7-DoF

Figure 6: Performance is plotted against different levels of sensor noise for the three systems under consideration: baseline XCSF, XCSF with Kalman filtering, and XCSF with bounded Kalman filtering. With noisy sensors, the latter approach still allows for accurate control where the baseline system fails.



(a) 2-DoF



(b) 7-DoF

Figure 7: The angular error is plotted against the noise level σ_{sensor} with (diamonds) and without (squares) thresholded Kalman filtering. The data points have been fitted by tanh functions. The insets show the gain factor in noise tolerance as a function of admissible angular error.

reduces performance in the regime of moderate sensor noise levels. Only for higher noise levels does it show a small benefit for the 2-DOF arm, but only when its performance has already become unacceptably bad. The introduction of a filtering threshold θ_β completely changes the picture. For small σ_{sensor}, the baseline is reproduced and the performance stays almost unaffected for a much wider range (until 0.02 instead of 0.005 for 2-DOF, until 0.01 instead of 0.002 for 7-DOF). Taking 60% as the acceptance limit for the quality, we see that our proposed system can cope with higher noise levels compared to the baseline system (10 times higher for 2 DOF, four times higher for 7 DOF).

In Figure 7, we compare in similar fashion the angular error for baseline and thresholded Kalman filtering for 2-DOF and 7-DOF arms. Note that this data, which was already published in Kneissler et al. (2012), was obtained using slightly different XCSF settings (learning steps, population size, and other parameters) and using a faster offline measurement procedure (only the first 30 steps of a period where evaluated for averaging the angular error). The solid and dashed curves were obtained by fitting a tanh function to the angular error data points of the baseline and thresholded Kalman filtering system, respectively (in the log(σ_{sensor}) domain). The insets show the resulting gain factor in noise robustness plotted over the acceptance level of the angular error. It can be seen that noise tolerance is up to 20 times higher in case of the 2-DOF arm. For the 7-DOF arm, the maximal factor of approximately 10 is reached at the less challenging end (high error levels accepted). When requiring higher performance, the noise tolerance factor decreases to five.

4.7 Optimal Thresholds θ_β

So far, we have treated the threshold θ_β as a heuristically introduced, free parameter and have optimized it by exhaustive search. Now, we derive a formula to get an estimate for θ_β as a function of σ_{sensor} based on reasonable assumptions.

Let us start by assuming that irrespective of the noise level, XCSF eventually is able to establish a good distribution of classifiers, such that the estimation error becomes uniformly distributed over the input space. Furthermore, let us assume that in the limit of infinitely many samples observed, each of the estimators converges to its optimum (as if it would have seen noise-free samples). This implies that for each noise level, the root mean squared prediction error is limited uniformly by the same final error ϵ_∞.

The Kalman estimate for the variance V_n, and thus also the mixing constant β, will then converge as well, say against V_∞, β_∞. We can calculate these steady-state solutions by using Equation (2) and setting $V_{n\text{-}1} = V_n = V_\infty$ and $\sigma_{n\text{-}1} = \epsilon_\infty$ in Equations (4) and (5):

$$V_\infty = (1 - \beta_\infty)(V_\infty + \epsilon_\infty^2) \tag{13}$$

$$\beta_\infty = \frac{V_\infty + \epsilon_\infty^2}{V_\infty + \epsilon_\infty^2 + \sigma_{\text{sensor}}^2} \tag{14}$$

This system leads to a quadratic equation for β_∞ with a single positive solution:

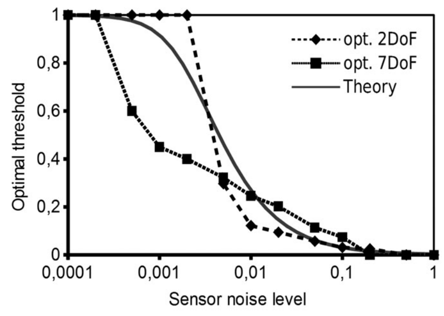
$$\beta_\infty = \frac{2}{1 + \sqrt{1 + \left(\frac{2\sigma_{\text{sensor}}}{\epsilon_\infty}\right)^2}} \tag{15}$$

Since the average error in the early phases can be expected to be $>\epsilon_\infty$, the corresponding Kalman mixing factor should always be larger than β_∞. It is thus reasonable to choose the threshold $\theta_\beta := \beta_\infty$ in order to avoid self-delusions without degradation of the system's learning capability.

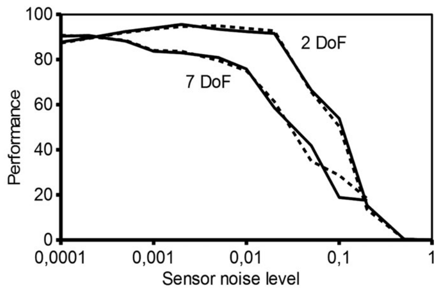
Figure 8(a) shows that the experimentally found optimal θ_β fits well with Equation (15) for $\epsilon_\infty^2 = 10^{-5}$. Figure 8(b) shows that for both arm setups, the performance of the system is not degraded when the theoretical threshold β_∞ is used instead of the optimized values.

4.8 Analysis of the Self-Delusion Effect

To investigate the effect of self-delusional loop that was introduced by feeding the XCSF predictions back to the Kalman filter, we allowed each individual of the XCSF population to maintain three independent predictors; one was trained using the clean position samples (before addition of sensor noise), the second was trained using noisy samples (before Kalman filtering) and the third was obtained from the output of the

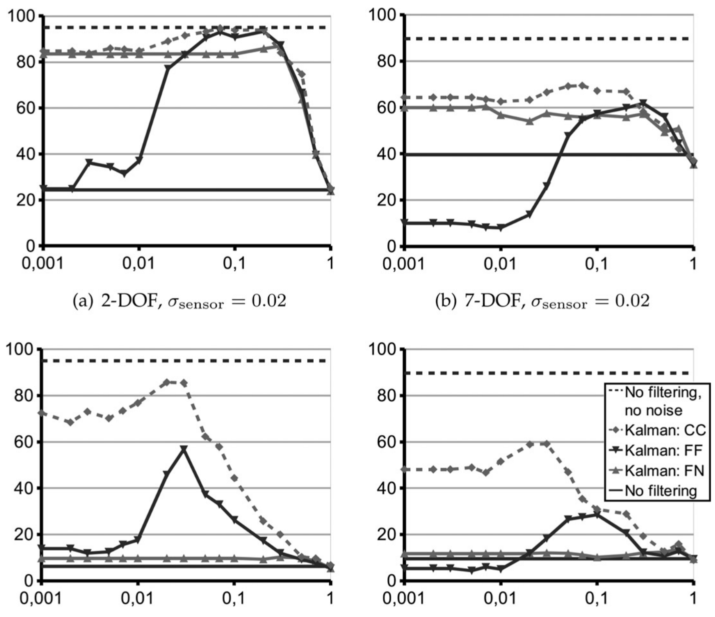


(a) Optimized/theoretical $\theta_\beta(\sigma_{\text{sensor}})$



(b) Performance loss with theoretical θ_β

Figure 8: (a) Theoretical β_∞ for $\epsilon_\infty^2 = 10^{-5}$ and optimized values of threshold parameters θ_β as function of noise level σ_{sensor}. (b) Performance comparison: using theoretical value $\theta_\beta = \beta_\infty$ (solid lines) instead of optimized θ_β (dashed lines).



(a) 2-DOF, $\sigma_{\text{sensor}} = 0.02$ (b) 7-DOF, $\sigma_{\text{sensor}} = 0.02$

(c) 2-DOF, $\sigma_{\text{sensor}} = 0.1$ (d) 7-DOF, $\sigma_{\text{sensor}} = 0.1$

Figure 9: Comparison of thresholded Kalman filtering (FF) and two modified systems: In CC, the input to the Kalman filter comes from predictors that have been trained on clean samples; in FN the error variance estimation is estimated on predictors trained on noisy (unfiltered) samples. We plot the quality measure as function of the threshold θ_β for the two different arms and for two selected levels of sensor noise.

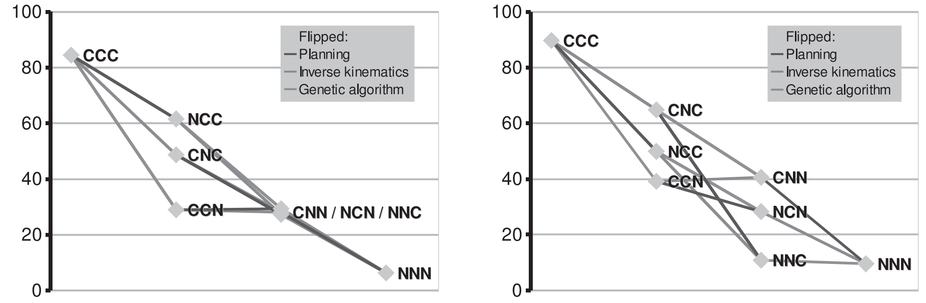
thresholded Kalman filtering. In Figure 9, we compare three sets of simulations that differ in the way that the Kalman filter was used: in setup CC, the Kalman filter was fed with output from the predictors trained on clean samples; setup FF used the filtered predictors; in setup FN the prediction of hand position change came from the filtered predictors while the error variance prediction came from the noisy predictors of the matched XCSF subpopulation. A number of observations can be made.

- **Effect of Self-Delusion Loop.** We suspected in Section 3.2 that the failure of a system with unbounded Kalman gain β is due to the feeding of information from predictors that have been trained on filtered samples back into the filter. This is confirmed by observing that the performance of system FF drops to the level of the baseline (or even below for the 7-DOF arm) when the threshold parameter θ_β approaches 0. In the setups CC, it stabilizes to a performance above the baseline for $\theta_\beta \to 0$.
- **Role of Error Variance Estimation in Self-Delusion Spiraling.** The curves FN, in which only the position prediction comes from predictors that have been trained on filtered samples, but not the variance estimates, agrees with those of CC and FF for high θ_β, and remains constant for small θ_β. For low levels of sensor noise, its performance is not too far from the maximum of curve FF. Thus, partially breaking the loop by avoiding variance feedback effectively prevents delusional-spiraling and leads to a Kalman filtering system that performs well without thresholding (and thus without the need for optimizing the ad hoc parameter θ_β). Unfortunately, the performance of the FN system quickly drops with increasing σ_{sensor} and is scarcely above baseline in Figure 9(c,d).
- **Influence of Thresholding.** On the right side of the optimum, from the curves for the setups FF are quite close to the curves for CC. Only toward lower θ_β hold the performance drop for FF, while it remains constant or further increases for CC. This can be taken as an indication that the thresholding does not interfere strongly with the Kalman filtering itself. Furthermore, there appears to exist a critical value for θ_β below which the negative effect of self-delusion starts getting noticeable and eats up the gains that could still be achieved by stronger filtering.

4.9 Investigation of Contributions of Different Parts of the System

The positive effect of filtering in the arm-controlling scenario is a combination of three separate sources:

- **Planning.** The desired movement direction can be determined more accurately using the filtered hand position instead of the noisy position signal from the sensors.
- **Inverse Kinematics.** The Jacobian used to translate desired movement into desired joint angle changes is more accurate when extracted from predictors trained on filtered samples.
- **Genetic Algorithm.** The error variance of predictors trained on noisy samples is contaminated with the sensor noise. In the filtering system, a predictor's error variance is thus closer to the noise-independent part of the prediction error (which measures to non-linearity of the target function for well-trained estimators). Therefore, its fitness estimate (derived from estimated error variance) is much better suited to guide the evolutionary algorithm.



(a) 2-DOF (b) 7-DOF

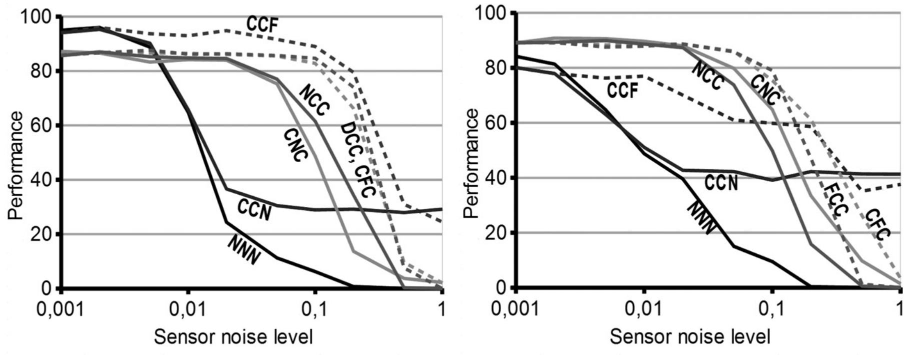
Figure 10: Contribution and interdependency of the three sources for performance degradation due to sensor noise. The performance for eight different setups is plotted along the y axis. The setup name is a combination of the letters C (for clean) and N (for noisy). The first letter stands for the hand position required for determining the direction toward the goal in planning. The second letter indicates if the predictors for estimating the inverse kinematics have been trained with clean or noisy samples. The third letter indicates which signal is used to estimate classifier fitness, used for guiding the evolutionary algorithm. Thus, in setup CCC, there is no sensor noise; the data point NNN illustrates the performance of the baseline system for noisy sensor signals. (In all experiments: $\sigma_{\text{sensor}} = 0.1$.)

In order to investigate the individual contributions of the three effects and their dependencies, we trained two predictors for each classifier: one trained on clean samples, the other trained on noisy samples (before/after addition of sensor noise). This was done in a modified baseline system (no filtering) in which we separately controlled which of the two sets of predictors was used at the three places described above. The result is given in Figure 10; it shows all eight possible combinations, labeled CCC...NNN (standing for planning, inverse kinematics, and genetic algorithm, as above). It shows, consistently for both arm setups, that the component that is most severely affected by noise is the genetic algorithm. The inverse kinematics is more critical than the planning in the 2-DOF case and the converse is true for the 7-DOF arm.

To verify in how far we can overcome the effect of noise by filtering, for the three described sources, we made similar changes to a system with thresholded Kalman filtering, allowing a third set of predictors that were trained on filtered samples. The input to the Kalman filter was always taken from the predictors that were fed with filtered samples. The threshold θ_β was set to optimal values according to Figure 8(a).

In Figure 11, we give the performance as a function of the sensor noise level σ_{sensor}. As can be seen, the genetic algorithm is affected the earliest of all three (i.e., note the steep drop of curve CCN in the range $0.002 \leq \sigma_{\text{sensor}} \leq 0.01$) while planning and inverse kinematics appear to be deteriorated only at much higher noise levels (around $\sigma_{\text{sensor}} \approx 0.1$). Compared to the baseline curve NNN, we see that the effect on the evolutionary algorithm is the only reason for failure under moderately noisy conditions. Only later does the CCN curve depart from NNN, which occurs when a negative effect on planning and inverse kinematics becomes noticeable (at $\sigma_{\text{sensor}} \approx 0.02$).

Curiously, the deterioration due to noise in the genetic algorithm appears to become stable at $\sigma_{\text{sensor}} \geq 0.05$, and even extreme noise levels (in the order of magnitude of the



(a) 2-DOF (b) 7-DOF

Figure 11: Role of Kalman filtering for the three sources for performance degradation due to sensor noise. The performance with threshold Kalman-filtering is plotted over noise level σ_{sensor} for seven different setups. The setup name is encoded as a combination of the letters C (for clean), N (for noisy), and F (for filtered). The position in the setup name is encoded as in Figure 10. Thus, NNN is the baseline system, the effect of filtering on the evolutionary algorithm alone can be seen by comparing *CCN* with *CCF*, etc.

total arm length) do not cause further performance loss. This is different for the effect of noise on planning and inverse kinematics: performance gradually drops to 0 when noise is applied to these components of the system. So while the evolutionary subsystem is already affected by low noise levels (compared to the rest of the system), it appears not to fail completely, irrespective of how much noise is present.

The dotted lines show that filtering drastically improves performance for all three sources. With the exception of curve CCF (genetic algorithm) in the 7-DOF case (Figure 10(b)), quality stays stable up to or even beyond a noise level of 0.1.

5 Summary and Conclusion

The goal of this work was to improve the performance of XCSF in noisy arm control scenarios. To do so, we have introduced information content based weighting of classifier predictions and an online estimation of the prediction error variance. To incorporate Kalman filtering, the evolving forward velocity kinematics model of XCSF was used to generate state predictions and thus filter the incoming sensory information. The performance results suggest that XCSF indeed runs into a self-delusional spiral, in which it may overfilter the incoming sensory information. With the setting of an appropriate threshold to prevent this effect, it was shown that XCSF can cope much better with sensor noise (up to 20th times higher noise levels, depending on circumstances and performance criterion).

In further investigations, the self-delusional feedback loop was confirmed as the source for the complete failure of Kalman filtering without thresholding. We identified the evolutionary algorithm as the limiting factor for performance degradation due to sensory noise. The other components of the system were found to be inherently much more noise-tolerant. By the means of thresholded Kalman-filtering, the noise-resistance of the evolutionary algorithm becomes comparable to the other components of the system. The dependency of the threshold parameter on the noise level could be derived theoretically, allowing this setting to be tuned automatically.

This finding encourages us to further explore the relation to Bayesian information processing with XCSF. We aim to derive a fitness criterion for the predictors that is inherently less deteriorated by sensor noise, promising an overall improvement of fitness guidance in XCSF.

We expect that the presented approach is also applicable to other prediction scenarios involving noisy sensors and possibly actuators. The setup, however, will generally be the same and the message of this paper appears to apply in a general sense: given that XCSF is predicting sensory information changes over time, its learning performance, its predictive capabilities, and possibly (dependent on the setup), its inverse control capabilities, can be improved by exploiting the predictive knowledge of XCSF, filtering the incoming sensory information online.

Acknowledgments

The authors acknowledge funding from the Emmy Noether program (German Research Foundation, DFG,BU1335/3-1) and thank the Cognitive Modeling team.

References

Ben-Israel, A., and Greville, T. N. (2003). *Generalized inverses: Theory and applications*. Berlin: Springer-Verlag.

Bernadó-Mansilla, E., and Garrell-Guiu, J. M. (2003). Accuracy-based learning classifier systems: Models, analysis, and applications to classification tasks. *Evolutionary Computation*, 11:209–238.

Bull, L. (Ed.). (2004). *Applications of learning classifier systems*. Berlin: Springer-Verlag.

Butz, M. V. (2006). *Rule-based evolutionary online learning systems: A principled approach to LCS analysis and design*. Berlin: Springer-Verlag.

Butz, M. V., and Herbort, O. (2008). Context-dependent predictions and cognitive arm control with XCSF. *Proceedings of the Genetic and Evolutionary Computation Conference, GECCO 2008*, pp. 1357–1364.

Butz, M. V., Lanzi, P. L., and Wilson, S. W. (2006). Hyper-ellipsoidal conditions in XCS: Rotation, linear approximation, and solution structure. *Proceedings of the Genetic and Evolutionary Computation Conference, GECCO 2006*, pp. 1457–1464.

Butz, M. V., Lanzi, P. L., and Wilson, S. W. (2008). Function approximation with XCS: Hyperellipsoidal conditions, recursive least squares, and compaction. *IEEE Transactions on Evolutionary Computation*, 12:355–376.

Butz, M. V., Pedersen, G. K. M., and Stalph, P. O. (2009). Learning sensorimotor control structures with XCSF: Redundancy exploitation and dynamic control. *Proceedings of the Genetic and Evolutionary Computation Conference, GECCO 2009*, pp. 1171–1178.

Butz, M. V., Reif, K. L., and Herbort, O. (2008). Bridging the gap: Learning sensorimotor-linked population codes for planning and motor control. *Proceedings of the International Conference on Cognitive Systems, CogSys 2008*.

Butz, M. V., and Stalph, P. O. (2011). Modularization of XCSF for multiple output dimensions. *Proceedings of the 13th Annual Genetic and Evolutionary Computation Conference, GECCO 2011*, 1243–1250.

Drugowitsch, J., and Barry, A. (2008). A formal framework and extensions for function approximation in learning classifier systems. *Machine Learning*, 70:45–88.

Drugowitsch, J., and Barry, A. M. (2007). Mixing independent classifiers. *Proceedings of the 9th Annual Conference on Genetic and Evolutionary Computation, GECCO 2007*, pp. 1596–1603.

Holland, J. H., and Reitman, J. S. (1978). Cognitive systems based on adaptive algorithms. In Waterman, D. A. and Hayes-Roth, F. (Eds.) *Pattern directed inference systems* (pp. 313–329). New York: Academic Press.

Kneissler, J., Stalph, P. O., Drugowitsch, J., and Butz, M. V. (2012). Filtering sensory information with XCSF: Improving learning robustness and control performance. *Proceedings of the Genetic and Evolutionary Computation Conference, GECCO 2012*, pp. 871–878.

Lanzi, P. L., Loiacono, D., Wilson, S. W., and Goldberg, D. E. (2006). Prediction update algorithms for XCSF: RLS, Kalman filter and gain adaptation. *Proceedings of the Genetic and Evolutionary Computation Conference, GECCO 2006*, pp. 1505–1512.

Lanzi, P. L., Loiacono, D., Wilson, S. W., and Goldberg, D. E. (2007). Generalization in the XCSF classifier system: Analysis, improvement, and extension. *Evolutionary Computation*, 15:133–168.

Loiacono, D., Drugowitsch, J., Barry, A., and Lanzi, P. L. (2008). Analysis and improvements of the classifier error estimate in XCSF. In J. Bacardit, E. Bernadó-Mansilla, M. V. Butz, T. Kovacs, X. Llorà, and K. Takadama (Eds.), *Learning classifier systems. Lecture notes in computer science*, vol. 4998 (pp. 117–135). Berlin: Springer-Verlag.

Peters, J., and Schaal, S. (2008). Reinforcement learning of motor skills with policy gradients. *Neural Networks*, 21:682–697.

Sigaud, O., and Peters, J. (Eds.). (2010). *From motor learning to interaction learning in robots*. Berlin: Springer.

Sigaud, O., Salaun, C., and Padois, V. (2011). On-line regression algorithms for learning mechanical models of robots: A survey. *Robotics and Autonomous Systems*, 59(12):1115–1129.

Stalph, P. O, and Butz, M. V. (2009). Documentation of JavaXCSF. Cognitive bodyspaces: Learning and behavior. University of Würzburg, Germany. (COBOSLAB Report Y2009N001).

Stalph, P. O., and Butz, M. V. (2012). Learning local linear Jacobians for flexible and adaptive robot arm control. *Genetic Programming and Evolvable Machines*, 13(2):137–157.

Stalph, P. O., Rubinsztajn, J., Sigaud, O., and Butz, M. V. (2010). A comparative study: Function approximation with LWPR and XCSF. In *Proceedings of the 12th Annual Conference on Genetic and Evolutionary Computation*, pp. 1863–1870.

Vijayakumar, S., D'Souza, A., and Schaal, S. (2005). Incremental online learning in high dimensions. *Neural Computation*, 17:2602–2634.

Wilson, S. W. (1995). Classifier fitness based on accuracy. *Evolutionary Computation*, 3(2):149–175.

Wilson, S. W. (1998). Generalization in the XCS classifier system. *Proceedings of the Third Annual Conference on Genetic Programming*, pp. 665–674.

Wilson, S. W. (2000). Get real! XCS with continuous-valued inputs. In P. L. Lanzi, W. Stolzmann, and S. W. Wilson (Eds.), *Learning classifier systems: From foundations to applications. Lecture notes in artificial intelligence*, vol. 1813 (pp. 209–219). Berlin: Springer-Verlag.

Wilson, S. W. (2001). Function approximation with a classifier system. *Genetic and Evolutionary Computation Conference, GECCO 2001*, pp. 974–981.

Wilson, S. W. (2002). Classifiers that approximate functions. *Natural Computing*, 1:211–234.