Self-learning Smart Cameras
Harnessing the Generalization Capability of XCS

Anthony Stein[1], Stefan Rudolph[1], Sven Tomforde[2] and Jörg Hähner[1]
[1]*Organic Computing Group, University of Augsburg, Eichleitnerstr 30, 86159 Augsburg, Germany*
[2]*Intelligent Embedded Systems, University of Kassel, Wilhelmshöher Allee 73, 34121 Kassel, Germany*

Keywords: Organic Computing, Self-x, Self-learning, Self-adaptive, Reinforcement Learning, Smart Camera, Smart Camera Network, Surveillance Camera, Learning Classifier System, Extended Classifier System, Q-learning.

Abstract: In this paper, we show how an evolutionary rule-based machine learning technique can be applied to tackle the task of self-configuration of smart camera networks. More precisely, the Extended Classifier System (XCS) is utilized to learn a configuration strategy for the pan, tilt, and zoom of smart cameras. Thereby, we extend our previous approach, which is based on Q-Learning, by harnessing the generalization capability of Learning Classifier Systems (LCS), i.e. avoiding to separately approximate the quality of each possible (re-)configuration (action) in reaction to a certain situation (state). Instead, situations in which the same reconfiguration is adequate are grouped to one single rule. We demonstrate that our XCS-based approach outperforms the Q-learning method on the basis of empirical evaluations on scenarios of different severity.

1 INTRODUCTION

The possible applications for networked cameras are manifold (Hoffmann et al., 2008). The most common use cases today are the detection of intruders in private areas (e.g. the home of a person or research facilities with classified prototypes) and the surveillance of high risk areas (e.g the track area in train stations or heavy machines in factories). But, there are also future applications such as huge parking places where the cameras could spot empty parking lots in order to guide the drivers to them, or the monitoring of persons that are potentially in need, such as nursing children or ill people.

Today's commercially available surveillance camera systems consist of cameras with *pan, tilt, zoom* (PTZ) capabilities. The captured video data is streamed to a central control room where security staff tries to monitor a large amount of this stream in parallel to identify critical situations that require countermeasures. In most cases, security staff is not able to fulfill this task properly due to natural limitations in terms of handling huge amounts of data simultaneously, e.g. multiple video streams at the same time, and the ability to concentrate over long time, e.g. an entire working day.

Previous work has addressed these issues by introducing smart cameras (Valera and Velastin, 2005). These cameras include a computational unit that allows to fulfill several tasks such as image processing, photogrammetry, or object tracking. Intervention of humans is optional, but not required for an efficient service of the system. Furthermore, most often smart cameras form a network, resulting in a so-called *Smart Camera Network* (SCN) (Rinner et al., 2008). That introduces the additional challenge of coordination between the cameras. Such systems are mostly designed to be self-organizing since they face the requirement of only local interaction and decision making since it is important to avoid problems that cannot be computed in the given time limits or create huge network loads. For example, in a huge smart camera network, it is not possible to find the globally best alignment of all cameras in a short time span.

To enable a local decision making, Rudolph et al. (2014) introduced the possibility of self-learning algorithms for the alignment problem, i.e., to determine the optimal PTZ configuration for a certain situation. Here, we pick this work up and improve their previous results by introducing the *Extended Classifier System* (XCS) as control algorithm.

Accordingly, the contribution of this paper is the application of a further, more sophisticated reinforcement learning technique – the XCS – in order to improve the self-learning capability of smart cameras. Based on the results of the conducted studies, we also

Stein A., Rudolph S., Tomforde S. and HÃd'hner J.
Self-learning Smart Cameras - Harnessing the Generalization Capability of XCS.
DOI: 10.5220/0006512101290140
In *Proceedings of the 9th International Joint Conference on Computational Intelligence (IJCCI 2017)*, pages 129-140
ISBN: 978-989-758-274-5

touch upon insights regarding the impacts of different learning parameter configurations in scenarios of varying complexity. Additionally, the resilience of the techniques employed to equip smart cameras with the intended self-learning capability is reflected in our experiments by considering camera failures and look at the time needed for the remaining camera to recover.
The remainder of this work is structured as follows: In Section 2, we give an overview of related work in the areas: (1) Coverage optimization in smart camera networks, (2) applications of reinforcement learning algorithms to smart cameras, and (3) the utilization of *Learning Classifier Systems* (LCS) (Holland et al., 2000) in real world contexts. Subsequently, in Section 3, we explain the general concept and learning interaction of XCS, the most prominent representative of LCS today. The formulation of our application scenario as *Markov Decision Process* (MDP) and necessary adaptions of the conventional XCS algorithm are the subject of Section 4. We demonstrate the benefit of using XCS based on the results of conducted experiments in Section 5. Eventually, Section 6 closes this work with a brief summary and an outlook to planned future work.

2 RELATED WORK

There are several works on the coverage optimization of cameras (Sec. 2.1) and *Reinforcement Learning* (RL) in smart camera networks (Sec. 2.2). The attempts include alignments through the PTZ capabilities and the placement of the cameras. There are also some works using RL techniques in the smart camera domain. Furthermore, there are several real-world applications on which XCS has been applied (Sec. 2.3).

2.1 Coverage Optimization in SCN

Recently, (Liu et al., 2016) reviewed approaches for the optimization of placement and alignment of cameras. The majority of approaches focuses on settings where a priori information about the environment is available, e.g. the importance of particular spots. Here, we follow a reinforcement learning approach, i.e. the reconfigurations are based only on feedbacks that we retrieve as a consequence of the realized actions (for a more detailed description of the reinforcement learning approach, see Sec. 4.1).
Murray et al. (2007) focus on an urban model with positions that allow to install cameras. Then an off-line optimization is used to determine the minimum number of cameras and the according alignment is calculated for the best coverage. This global optimization is computationally highly intensive and therefore not applicable for the on-line determination of the optimal alignment that is in the focus of this work.
Erdem and Sclaroff (2006) propose a possibility to optimize the placement and alignment of cameras according to a given floor plan and task-specific constraints. The method proposed is an off-line optimization algorithm and therefore can only be applied in advance to the camera installation. This paper focuses on the optimization of the camera alignments during runtime without previous knowledge of the environment.
Piciarelli et al. (2011) present an algorithm that utilizes Expectation Maximization to optimize the alignment of coverage in a three dimensional environment, but they focused on static and known environments.

2.2 Reinforcement Learning in SCN

Khan and Rinner (2012a) used Cooperative Q-Learning to schedule tasks in a Wireless Sensor Network and utilize it for an object tracking algorithm. In following work (Khan and Rinner, 2012b), the same authors employed artificial neural networks to tackle the dynamic power management of a traffic monitoring system involving smart cameras among other sensors.
Lewis et al. (2013) used bandit solvers as part of an algorithm for the allocation of tracking objects to cameras.
Even though RL techniques have been successfully applied in the domain of smart camera networks, to the best of our knowledge none of these contributions focuses on the problem of camera alignment, except for Rudolph et al. (2014). In this paper, we use one particular approach presented in their work, the *Q-learning* (Watkins and Dayan, 1992) algorithm, for comparison purposes.

2.3 Learning Classifier Systems in Real World Scenarios

Learning Classifier Systems (LCS) have gained plenty of research attention since their invention by John Holland (Holland, 1976). Initially designed to solve binary encoded reinforcement learning tasks, today many applications to real-world problems can be found in the literature. For instance, Goldberg applied LCS to simulated gas pipeline control (Goldberg, 1987). An application to robot arm control was reported by Stalph and Butz (2012).
Variants of LCS have been applied to system-on-chip architectures in (Bernauer et al., 2011). Bull et al. (2004) proposed the application of an LCS to traffic

management. In Prothmann et al. (2008), a substantially modified LCS that changes and constrains the generalizing nature of the conventional system is presented and applied to adapt traffic lights at urban intersections.
What such real-world scenarios typically have in common is the complexity of the underlying problem space. Not all possible states are known a priori what necessitates learning during the system's runtime. Large problem spaces with complex reward functions are the result. In consequence, the system has to cope with unforeseen or not anticipated situations at runtime. In order to cope with this challenging aspect, recently, a further extension was carried out to XCS in Stein et al. (2017b). The incorporation of a so-called interpolation component supports the system by increasing the learning speed and reducing the overall system error. This interpolation-based XCS variant was also applied to the aforementioned traffic light management scenario in Stein et al. (2016).
Another application domain XCS was recently applied to, is the task of ensemble time series forecasting. When signals are to be interpreted in technical systems, individual forecast methods have different strengths and weaknesses for different characteristics of time series. In Sommer et al. (2016a,b), an XCS derivative for function approximation (XCSF) was applied in order to learn the weights for a linear combination of the outputs (forecast values) of a heterogeneous ensemble of forecasting techniques.

3 EXTENDED CLASSIFIER SYSTEM

The Extended Classifier System (XCS) is a flexible, evolutionary rule-based online machine learning system. It was introduced in Wilson (1995) and can be seen as one of the biggest milestones in LCS research. XCS evolves its knowledge base as a population $[P]$ of condition-action rules, also called *classifiers*. This set $[P]$ is iteratively filled with a maximum number N of classifiers of the following structure:

$$cl := (C, a, p, \varepsilon, F, exp, num, ts, as)$$

The first attribute C determines a certain subspace of the defined n-dimensional state (or input) space S that this classifier covers and is called the *condition*. By this means, XCS partitions the state space S into smaller subspaces. In contrast to tabular Q-Learning (Watkins and Dayan, 1992) where for each state-action pair a single Q-value is approximated (more precisely $|S \times A|$ Q-values), XCS can generalize over a finite or even infinite set of situations/states.

A classifier cl is 'activated' and put to a so-called *match set* $[M] \subseteq [P]$ whenever a situation $\sigma(t) := \vec{x} \in S$ arrives that is encompassed by the condition C of cl (in the following we use the dot-notation to refer to a specific attribute of a certain classifier cl). $cl.a \in A$ represents one possible *action* of the action space A that a specific classifier cl advocates.

Each classifier also maintains an incrementally calculated estimate of the average reward it has received so far for advocating action a in one of the situations $\sigma \in cl.C$ which is usually called *predicted payoff* $cl.p$. The *mean absolute error* of $cl.p$ and the actually received reward r after realizing action $cl.a$ is stored in the attribute $cl.\varepsilon$. The accuracy of predicting the correct payoff relative to the environmental niche is calculated as a sort of inverse of $cl.\varepsilon$ and is called the classifier's fitness $cl.F$.

An *experience* statistic, $cl.exp$, gives indication of how often this classifier was selected for action execution and subsequently updated by the reinforcement mechanisms of XCS.

The numerosity $cl.num$ of a classifier determines the number of classifiers that this classifier could subsume so far. Subsumption happens when a classifier cl_1 is found to be more or at least equally general as another classifier cl_2 which is to be subsumed. Additionally, cl_1 has to be more accurate than the predefined error tolerance ε_0 and must advocate the identical action as cl_2. A classifier with a $cl.num > 1$ is called a *macroclassifier*, whereas a newly generated classifier with $cl.num == 1$ is termed a *microclassifier*. The main advantage of introducing subsumption to XCS is the reduced computational effort during the matching procedure, as well as the more compact representation of the entire population $[P]$.

The so-called *timestamp* attribute $cl.ts$ stores the last time when the involved *Genetic Algorithm* (GA) was invoked on a set of selected classifiers where this classifier also belonged to. It is used to control the application frequency of the GA.

XCS attempts to create a maximally accurate, maximally general, complete and compact mapping $S \times A \rightarrow R$, where R determines the set of possible reward values. To create a complete mapping, it is necessary to guarantee that each environmental niche (subspace of the entire state space S) is covered and gets assigned an equal amount of resources in terms of microclassifiers. The *action-set size* attribute $cl.as$ is used to determine a deletion candidate classifier when the maximum number of microclassifier N is exceeded and thus the following condition holds: $\sum_{cl \in [P]} cl.num \geq N$. In consequence, each environmental niche should have been assigned an equal number of classifiers.

XCS is flexible since it can be applied to a variety of machine learning problems, e.g. Reinforcement Learning problems (Wilson, 1995; Butz et al., 2005) or rather Supervised Learning tasks, such as pure classification (Wilson, 2001) or also regression (Stalph and Butz, 2012). There have also been investigations on using XCS as an unsupervised learning mechanism (Tamee et al., 2007).
XCS is called an evolutionary learning system since it relies on a steady-state niche GA to find a globally optimal state space coverage. The system thereby pursues two main goals: (1) evolve classifies that are maximally general, and (2) retain maximal prediction accuracy. This relationship in combination with population-wide deletion became known as Wilson's *generalization hypothesis* (Wilson, 1995) and was extended by Kovacs to the *optimality hypothesis* (Kovacs, 1998).
In essence, XCS learns in an online fashion by partitioning the state space S into smaller problems and by estimating reward predictions locally by means of (stochastic) gradient-descent techniques (Lanzi et al., 2007). Additional to the local approximation, the incorporated GA is responsible for optimizing the state space coverage globally. From an architectural point of view, XCS can be decomposed into three main components: (1) The *performance component* that accomplishes the tasks of finding matching classifiers and form $[M]$, building the prediction array PA that is further used to determine the action to be executed a_{exec}, and finally realizes the chosen action. (2) The *reinforcement component* takes care of updating all classifiers advocating the same action as the one (a_{exec}) to be executed (these are grouped in a so-called action set $[A] \subseteq [M]$) on the basis of the received reward $r \in R$. (3) The *discovery component* comprises the *covering* mechanism that is responsible for generating classifiers on demand, as well as the steady-state niche GA for refining the conditions of already existing classifiers to reach an optimal state space coverage.

4 APPROACH

This section describes our assumptions and defines the problem space where the two algorithms under consideration - XCS and Q-learning - are applied to. Afterwards, we describe how XCS was adapted to fit the requirements of the outlined setting.

4.1 The Learning Task

We describe our Reinforcement Learning setting as Markov decision process (MDP), see Sutton and Barto (1998), and define the necessary components, i.e. the state space S, the action space A, the reward function $r\colon S \times A \to R$, as well as the state-transition function $\delta\colon S \times A \to S$.

4.1.1 State Space

The state of each smart camera controller is defined by the current values of its pan, tilt and zoom attributes. Thus, the state can be described as a 3-dimensional vector $\vec{s} = (c_p, c_t, c_z)^T$. We restricted the set of possible values for c_p, c_t, c_z according to reasonable ranges and technical specifications. Furthermore, we discretized the ranges as follows:
$c_p \in \{30, 60, \dots, 360\}$, $c_t \in \{120, 150, 180\}$ and $c_z \in \{12, 18\}$.
According to the above definitions, we get an state space S with a magnitude of $12 \cdot 3 \cdot 2 = 72$.

4.1.2 Action Space

For each of the three control variables that constitute the state $\vec{s}$, a possible action determines whether each of these values is increased ($\uparrow$), decreased ($\downarrow$) or remains unchanged ($=$). This leads to an action space A of a magnitude of $3 \cdot 3 \cdot 3 = 27$. We predefined the increase as well as the decrease amount for each of the three control variables. An action vector $\vec{a}$ can thus be represented by numeric values, e.g. $\vec{a} = (+20, -30, 0)^T$. This allows a simple vector addition to yield the new system state $\vec{s'}$ (cf. Sec. 4.1.4). Combined with the state space, the overall problem space consists of $|S| \times |A| = 1944$ state-action combinations. Q-learning builds-up a table for each of these possibilities. By using XCS, we strive to harness its generalization capability to summarize similar state-action pairs into a single classifier.
We want to note that the action space could be restricted in a way so that there is no entry within the Q-table for actions that would have no effect on the succeeding state. For instance, when the zoom value is at its maximum value and the smart camera controller suggests to further increase the zoom, the algorithm would not need to approximate these values since they were already sorted out a priori by a human expert. However, such a restriction was not applied in this work.
A corresponding restriction of the action space has also not been applied to XCS for two reasons: (1) We intended to reduce the design-time effort and, thus, the system's engineer. (2) Such a restriction is not that

simple to implement, since XCS – roughly speaking – creates a sort of Q-table in an online manner and is designed to attempt the construction of a complete solution map $S \times A \rightarrow R$. Nonetheless, future research efforts are planned to address the introduction of exactly such exploration constraints to XCS.

4.1.3 Reward Function

After an action has been realized and, subsequently, the camera was reconfigured, a reward is calculated by determining the sum of newly detected objects within the vision area of all smart cameras. Hence, the reward is highly stochastic, since it depends not only on the configuration of a single camera, but also on the current configuration of all other cameras in the scenario under consideration. Naturally, it also depends on the previous configuration of the camera and the number of undetected objects that were exactly in that vision field, i.e. objects that have already been detected by any smart camera do not increase the reward of the current step.
The stochastic nature of our reward function necessitates an adaptive and incremental update procedure to converge to nearly optimal (Q-)values describing the utility of realizing action $\vec{a}$ in state $\vec{s}$. As we describe in the next section, we utilized the conventional method to cope with stochastic reward functions – a learning rate β which has been determined via various parameter studies.

4.1.4 State Transition Function

The state transition function δ is given by a simple vector addition of the current state vector $\vec{s} \in S$ and the selected action $\vec{a} \in A$. Thus, the next state is determined as $\vec{s'} = (c_p, c_t, c_z)^T + (a_p, a_t, a_z)^T$. In this work, we do not consider uncertainty as a consequence of noisy reconfigurations. This aspect constitutes a further topic on our research agenda.

4.2 Adaptations to XCS

Since we are confronted with a discretized state space, it seemed plausible to rely on a derivative of the conventional XCS system that is capable of dealing with numeric input values. In Wilson (2001), Wilson introduced an extension to XCS for nominal inputs. The condition is represented by so-called interval predicates that determine a lower and an upper bound for values accepted to satisfy the condition. In Wilson (2000), XCS was further extended toward XCSR – an XCS variant for real-valued input values.
For our problem scenario, we adopted Wilson's techniques by the following means: The condition is encoded by three interval predicates (l, u), one for each control variable (c_p, c_t, c_z). Since the state space is discrete, we modified the covering and mutation operators of XCS to reduce the search space complexity to a reasonable level. Whenever XCS needs to cover a situation not encountered so far, in a first step the condition of the new classifier is set exactly to the current state $\vec{s}$. Subsequently, the lower and upper bounds for each of the $i = 1 \ldots 3$ control variables are adapted as follows:

$$l_i = l_i - r_0 \cdot U(M_{cov}) \text{ and } u_i = u_i + r_0 \cdot U(M_{cov}).$$

Thereby r_0 is a predefined range that determines a discrete step along each state space dimension. M_{cov} is a multiplier and $U(M_{cov})$ delivers a uniformly distributed random integer value between 1 and M_{cov}.
A similar procedure is applied to the mutation operator of the GA utilized by XCS to find a globally optimal coverage of the problem space. The only difference is that the step range for mutation m_0 and the multiplier M_{mut} differ from r_0 and M_{cov}, respectively. Furthermore, for mutation it is also permitted that the lower bound is increased and the upper bound is decreased, i.e. $l_i = l_i \pm m_0 \cdot U(M_{mut})$ and $u_i = u_i \pm m_0 \cdot U(M_{mut})$. The sign is also selected at random.

So far, our applied interval representation has one considerable drawback: Yet, the step ranges r_0 and m_0 are globally defined for any dimension of the state space S. This may lead to an over-sized classifier condition that does not exactly meet the possible discrete values of the control variables as defined in Section 4.1.1. The elimination of this drawback is left for future work.

5 EVALUATION

In this section, we elaborate on the results of our empirical investigations that were carried out in the context of our described smart camera application. We chose this application domain for two reasons: (1) We can compare to previous work in this application context that adopts reinforcement learning techniques (Rudolph et al., 2014). (2) SCNs provide a realistic (real world) problem which is a field of active research (Piciarelli et al., 2016).
We conduct the experiments in a 3D simulation with PTZ-capable cameras. The objective of the cameras is to detect as many new objects as possible, i.e. already detected objects are not of interest. We explicitly note that the system goal is not congruent to the coverage maximization, since depending on the setup it is not necessary to observe the whole space but only the regions that are populated with objects. Such an

objective is suitable, e.g. for a people counter in a shopping mall, or for security purposes at the airport (detecting wanted criminals).

5.1 Experimental Setup

We compare the performance of XCS and Q-learning in three different scenarios which are depicted in Figures 1, 2, and 3. The schematics show a top view. Each scenario consists of one or multiple cameras which are represented by a black dot. Around the cameras, the colored sphere indicates the potential observable space, i.e. the area that can be observed when an appropriate PTZ-configuration is assumed. The small subregion – depicted by a red box – represents an exemplary field of view. For clarification, the entire colored sphere cannot be observed at once, but only partially as indicated by the red boxed area. Furthermore, the yellow arrows represent entry points and the movement direction of the objects to be detected. The quantity of entering objects is determined by a probabilistic process that can be described as follows: Within a maximal interval of t_{max} ticks an object enters the scene every $U(1, t_{max})$ ticks, where again $U(\min, \max)$ delivers a uniformly distributed random integer between min and max.
Please note that the figures only provide a schematic view on the different scenarios, but the actual experiments have been conducted in a simulated 3-dimensional environment.

Scenario 1. In this setting, one single camera in the middle of the space is asked to observe as many objects as possible. The objects enter the scene from the south. An adequate strategy for the camera would be to alternate between several pan configurations, since with a static configuration it is not possible to reach a high reward.

Scenario 2. In the second scenario, the number of objects is significantly decreased to one single stream. An exemplary strategy that allows to detect all objects would be to employ a static configuration that covers the stream.

Scenario 3. The third scenario is inspired by an airport where two man conveyors or escalators are situated beside each other but with different movement directions. Thus, a representative adequate strategy would be that each camera focuses on one stream of objects at a time using a static configuration as for scenario 2. A further challenge is introduced by a camera failure at a certain point in time. Accordingly, the remaining camera suddenly has to observe both streams

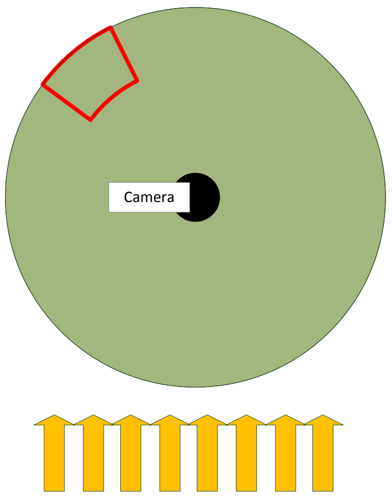


Figure 1: Schematic representation of scenario 1. One camera observes multiple streams of objects.

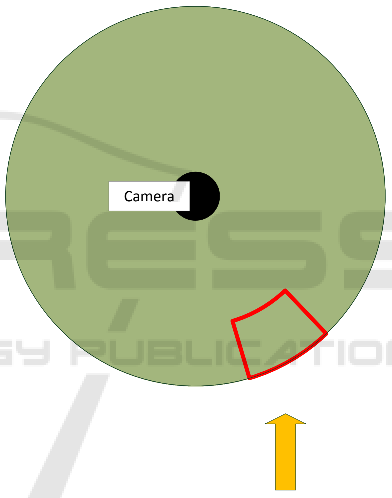


Figure 2: Schematic representation of scenario 2. One camera observes a single stream of objects.

and is thus forced to reconfigure to an alternating configuration.

5.2 Parameter Study

For our parameter studies, we repeated the experiments for 10 independent runs. For Q-learning, we conducted a comprehensive parameter study comprising parameters learning rate β (For the sake of consistency, we denote the learning rate with β as it is typically the case for XCS. In the standard literature for Q-learning, the learning rate is usually denoted by α.) and the discount factor γ for scenario 1. We tested the ranges of $\beta \in \{0.1, 0.2, \ldots, 1.0\}, \gamma \in \{0.1, 0.2, \ldots, 0.9\}$. Regarding the exploration/exploitation trade-off, we utilized

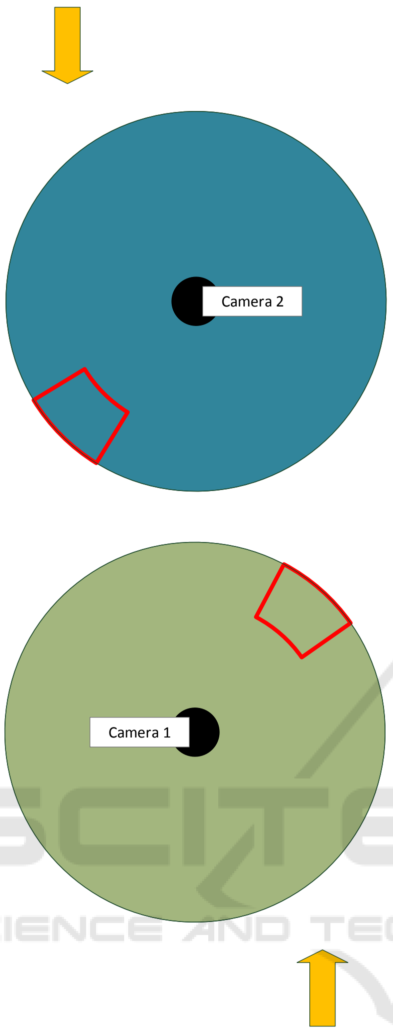
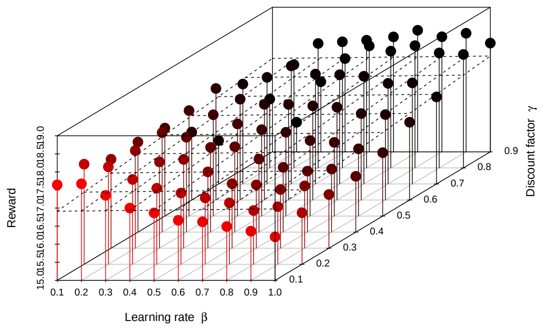


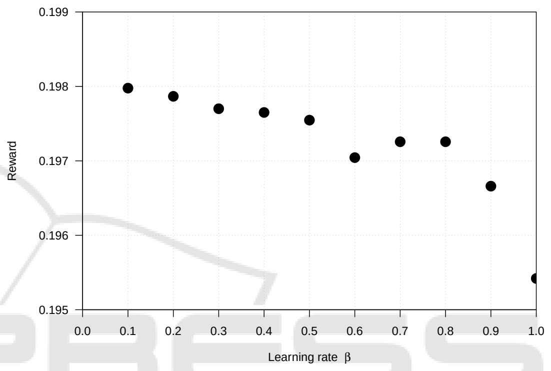
Figure 3: Schematic representation of scenarios 3. Two cameras observe two single streams of objects.

the ε-greedy action-selection strategy, where ε determines the probability of choosing a random action instead of the action with the maximum Q-value. Further results, not included in this work, showed that an exploration rate $\varepsilon = 0.05$ is appropriate for this learning task, therefore we limited the presented results to this value. The Q-values have been initialized with 0.005. Each configuration was executed for 30.000 steps. Each point on the plots shows the average over the entire 30.000 steps. To also capture the learning speed, we furthermore analyzed the learning progress to support our decisions (for the sake of brevity, these plots are not included in this paper.).

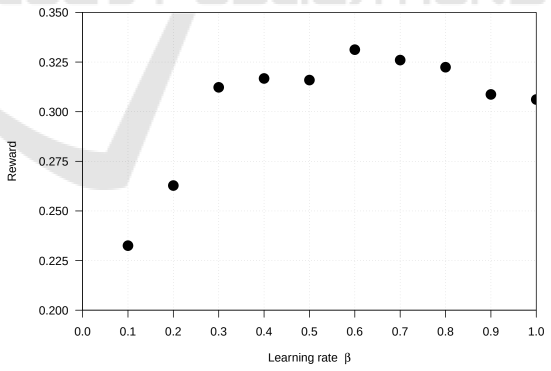
Figure 4(a) depicts the results as a 3-dimensional scatter plot. As Figure 4(a) shows, for scenario 1, the best results have been achieved with a configuration of $\beta = 0.7, \gamma = 0.9$.



(a) Scenario 1: Full parameter study of the learning rate β and the discount factor γ



(b) Scenario 2: Parameter study of the learning rate β



(c) Scenario 3: Parameter study of the learning rate β

Figure 4: Results of the conducted parameter studies. Points in the plot correspond to the average reward values of 30.000 learning steps.

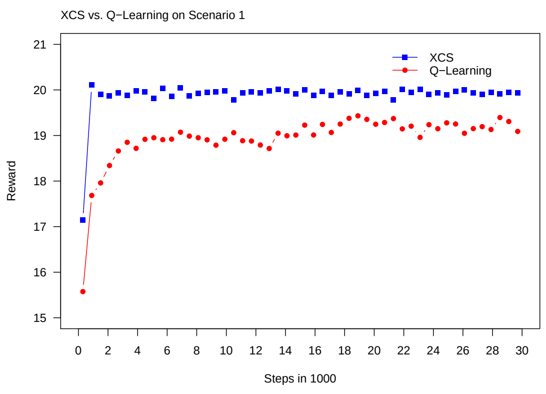
For the second and third scenario, we restricted our study to the learning rate β, since a comprehensive parameter study for each scenario is not viable due to the enormous computational effort. Thus, we fixed the value for γ to 0.9, since we yielded this value in our first parameter study for scenario 1. Furthermore, higher values for γ in general seemed to be legit, since we are confronted with a multi-step problem. The results of the partial parameter study for the scenarios 2 and 3 can be found in Figures 4(b) and 4(c), respectively. As can be seen, for the second scenario $\beta = 0.1$ and for scenario 3 $\beta = 0.6$ yield the best configurations. These observations could be supported by our analysis of the learning progress of all investigated configurations of β.

5.3 Comparison of XCS and Q-learning

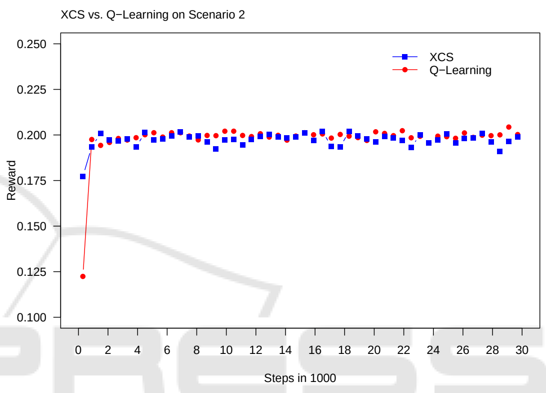
In the following, we compare Q-learning with XCS. For the comparison, we use the best configuration found for Q-learning based on the results of the parameter study described above. Each comparison is based on 30 independent runs with 30.000 steps for each algorithm. A statistical analysis was carried out considering a significance level of 5% ($\alpha = 0.05$). We performed a Shapiro-Wilk-test combined with a visual inspection of the Quantile-Quantile-plots to figure out whether the differences of the two samples stem from a normally distributed population. To test on variance homogeneity, an F-test was conducted. Depending on the data properties of variance homogeneity and normal distribution, we decided for an adequate test.

Since a comprehensive parameter study is not feasible due to the large number of dependent parameters, we handcrafted our configuration based on domain knowledge. For our experiments, XCS was configured as follows: $N = 800$, $\alpha = 0.1$, $\gamma = 0.9$, $\varepsilon = 0.05$, $\delta = 0.1$, $\nu = 5$, $\theta_{GA} = 12$, $\varepsilon_0 = 0.05$, $\theta_{mna} = |A| = 27$, $\theta_{del} = 50$, $\theta_{sub} = 50$, $\chi = 0.8$, $\mu = 0.04$, $p_{ini} = 0.005$, $\varepsilon_{ini} = 0.0$, $F_{ini} = 0.01$, $r_0 = 0.1$, $m_0 = 0.05$, $M_{mut} = M_{cov} = 3$. For each scenario, the learning rate β has been set to the values we yielded from the conducted parameter studies described above. Thus, to provide a comparability, we adopted the learning rate values used for Q-learning for our XCS configuration. For more details regarding the meaning of the standard XCS parameters, we refer to (Butz and Wilson, 2002; Wilson, 2000).

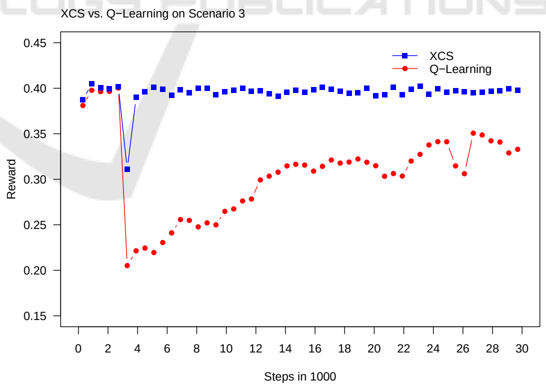
The results for scenario 1 can be found in Figure 5(a). We see that XCS clearly outperforms Q-learning in both learning speed and the quality of the learned strategy. To compare the results of the two algorithms in the overall performance, we relied on a unpaired two sample t-test with significance level $\alpha = 0.05$. XCS reached an overall reward of around 19.88 ± 0.05, whereas Q-learning showed an inferior average performance of around 18.94 ± 0.27. The p-value of the conducted two sample test is less than



(a) Scenario 1, $\beta = 0.7$ for XCS and Q-learning



(b) Scenario 2, $\beta = 0.1$ for XCS and Q-learning



(c) Scenario 3, $\beta = 0.6$ for XCS and Q-learning

Figure 5: Graphical comparison of XCS and Q-learning on three scenarios. Each point depicts the aggregated system reward, averaged over 600 measurements.

2.2×10^{-16}, which indicates that the average performance of XCS is significantly higher.

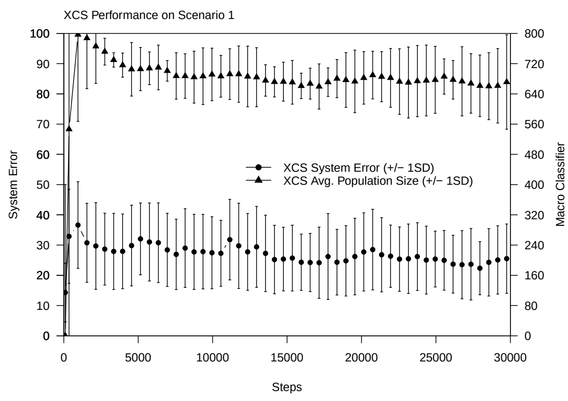
In scenario 2, we face a quite different situation. In Figure 5(b), we see that the learned strategy is of similar quality. However, during the initial learning phase,

XCS shows a faster learning behavior. We conducted the same significance test as for scenario 1. The average performance of XCS is around 0.197 ± 0.003. Q-learning reaches a performance value of about 0.198 ± 0.001. This difference is marginal, and with a p-value of 0.2827 not significant.
The results for scenario 3 are shown in Figure 5(c). There, during the initial learning phase, we see a similar behavior of XCS and Q-learning. At step 3000 the performance drops noticeably. This can be attributed to the simulated failure of one camera. Afterwards, we see a very fast recovery of the XCS controlled camera, but, the Q-learning controlled camera on the other hand does only show a rather slow upwards trend. The overall performance was about 0.40 ± 0.002 for XCS and 0.31 ± 0.037 for Q-learning, respectively. As indicated by the conducted two sample t-test, the difference of the performances are statistically significant having a p-value of 5.891×10^{-14}.
Concluding the results, we observe that overall XCS clearly outperforms Q-learning. This becomes apparent in scenarios 1 and 3. Especially, we figured out that XCS shows its strengths in scenario 3, where the self-learning cameras are asked to adapt to abruptly changing circumstances, i.e. the failure of a camera. Also in the second scenario, we observed a higher learning speed during the initial learning phase in comparison to Q-learning, but the eventually learned strategies are on the same level.
The observed performance benefits are attributed to the generalizing nature of XCS. For the defined MDP (cf. Sec. 4.1), Q-learning has to individually approximate $|S \times A| = 1944$ state-action pairs within its Q-table. XCS is able to yield improved results with only ≤ 800 $(= N)$ macroclassifiers, each representing a certain subset of the entire situation space S by means of its interval encoding discussed in Section 4.2. This leads to a simultaneous approximation of distinct but adjacent state-action-pairs which in turn results in a faster learning. The involved GA additionally exerts evolutionary pressure toward an optimal coverage of relevant subsets of the situation space S by favoring the most accurate classifiers in the most frequently occurring situations. A more proactive way of exploring the situation space as proposed by Stein et al. (2017a) could further improve the learning speed but is not subject of this paper. Although the experimental results already show benefits when using a generalizing XCS, we assume that the performance difference can still be increased when XCS is configured with nearly optimal parameters. However, this necessitates an exhaustive parameter study which demands for a high computational effort.

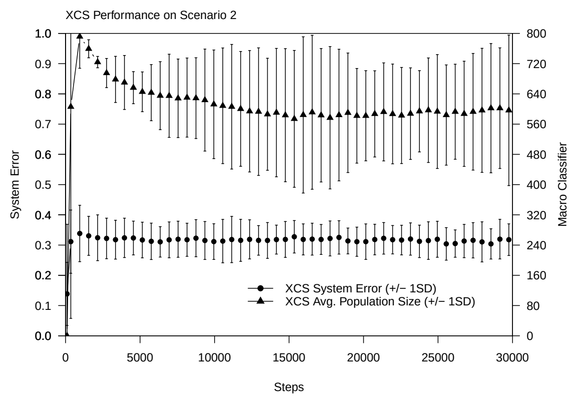
5.4 XCS Analysis

Figure 6 shows the learning progress of XCS. The plots show two important metrics: (1) The mean absolute system error, which averages the deviation from the actual state-action value to the system prediction over the last 100 steps. The system error relates to the state-action value which comprises the reward for realizing the chosen action of the previous step plus the discounted maximum prediction array value of the current time step. (2) The number of (macro-)classifiers in the population $[P]$, again averaged over the last 100 steps.

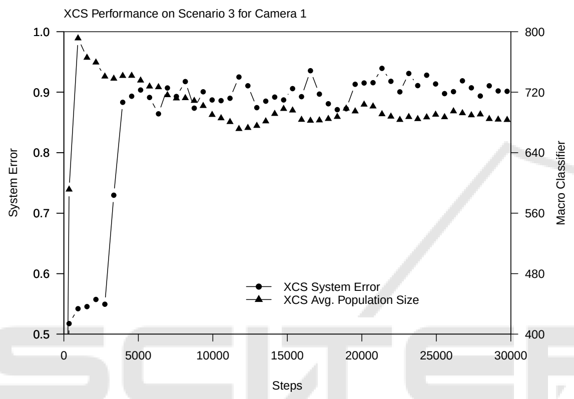
As Figure 6(a) suggests, the system error increases during the initial learning phase and subsequently decreases gradually. Considering the number of classifiers needed to approximate the problem space, we can observe that after an initial peak near to the allowed maximum number N, the level continuously drops to less than 700 classifiers. Remembering the superior performance of XCS in scenario 1, this is a clear indicator that the utilization of XCS' generalization mechanism yields beneficial effects over the tabular representation of Q-learning.
For the second scenario, on average we can observe a more significant decrease regarding the population size $|[P]|$, however with a larger standard deviation in comparison to scenario 1. This observation is attributed to the fact that an optimal strategy for this scenario appears to be much simpler than for the first scenario. Considering the system error, there is no clear trend observable, i.e. the system error stagnates on a similar level. Possible reasons for this behavior are: (1) the stochastic nature of the reward function, (2) the applied non-decaying probability of selecting a random action $\varepsilon = 0.05$, as well as, (3) the restricted expressiveness of the utilized interval condition representation, as discussed in Section 4.2. Under the presumption of the aforementioned error sources, we expect that XCS has already converged to the minimal achievable error (expected value).
The third scenario introduces a further challenge to the system. After 3000 steps, we simulated a camera failure. Both cameras deploy one XCS instance each. During the initial phase before the camera failure, both cameras show a similar behavior regarding the system error and the number of macroclassifiers in the population (see Fig. 6(d)). Looking at Figure 6(c), a sharp increase of the system error is observable subsequent to the failure of the second camera. Subsequently, the system error oscillates and remains on that increased level. As we outlined in Section 4.1, the reward function of the present scenario is stochastic. After the camera failure, the remaining camera



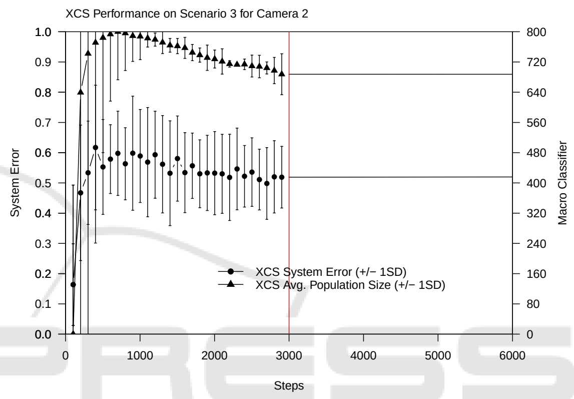
(a) Scenario 1



(b) Scenario 2



(c) Scenario 3 for camera 1



(d) Scenario 3 for broken camera 2

Figure 6: Learning progress of the XCS learning algorithm in terms of the number of evolved macroclassifiers and the system error for the three considered scenarios. For the sake of readability, we omitted the error bars depicting the standard deviation in subfigure (c).

is asked to observe both streams of objects to be detected. Thus, more reward can be gained but the degree of non-determinism also increases. As for scenario 2, we assume that the error level has already converged and can hardly decrease further due to the aforementioned three reasons. As for the first and the second scenario, also in scenario 3 the average population size gradually reduces to a level of less than 700 classifiers.

6 CONCLUSION

Surveillance networks face certain challenges such as finding the the optimal camera alignment or automated detection of suspicious events. In this paper, we addressed the issue of automated camera alignment by means of reinforcement learning approaches that aim to maximize the number of detected objects in the observable range. We adapted the well-known extended classifier system technique and applied it to the camera control problem. Within our experimental evaluation, we compared the approach to an alternative technique, the Q-learning algorithm, that has been previously proposed for the same problem setting in literature. We demonstrated that the XCS-based approach is able to significantly increase the utility. This observation was attributed to the generalization approach of this particular learning system.

Future Work. As already indicated within the paper, our current and future work focuses on two main challenges resulting from the presented approach: (1) an introduction of exploration constraints in XCS may be beneficial to guide the exploration behavior and improve the learning speed, and (2) the incorporation of expert knowledge (i.e. a priori knowledge of humans) may also be used to steer the desired learning behavior. In addition, we will apply the developed technique to large-scale simulation with heterogeneous constellations of cameras, e.g. in terms of varying capabilities.

ACKNOWLEDGEMENTS

This research is partially funded by the DFG within the project CYPHOC (HA 5480/3-2).

REFERENCES

Bernauer, A., Zeppenfeld, J., Bringmann, O., Herkersdorf, A., and Rosenstiel, W. (2011). Combining Software and Hardware LCS for Lightweight On-chip Learning. In *Organic Computing*, pages 253–265. Birkhäuser Verlag, Basel, CH.

Bull, L., Sha'Aban, J., Tomlinson, A., Addison, J., and Heydecker, B. (2004). Towards Distributed Adaptive Control for Road Traffic Junction Signals using Learning Classifier Systems. In Bull, L., editor, *Applications of Learning Classifier Systems*, volume 150 of *Studies in Fuzziness and Soft Computing*, pages 276–299. Springer Berlin Heidelberg.

Butz, M., Goldberg, D., and Lanzi, P. (2005). Gradient descent methods in learning classifier systems: improving XCS performance in multistep problems. *Evolutionary Computation, IEEE Transactions on*, 9(5):452–473.

Butz, M. and Wilson, S. W. (2002). An Algorithmic Description of XCS. *Soft Comput.*, 6(3-4):144–153.

Erdem, U. M. and Sclaroff, S. (2006). Automated camera layout to satisfy task-specific and floor plan-specific coverage requirements. *Computer Vision and Image Understanding*, 103(3):156–169.

Goldberg, D. E. (1987). Computer-aided pipeline operation using genetic algorithms and rule learning. part ii: Rule learning control of a pipeline under normal and abnormal conditions. *Engineering with Computers*, 3(1):47–58.

Hoffmann, M., Hähner, J., and Müller-Schloer, C. (2008). *Towards Self-organising Smart Camera Systems*, pages 220–231. Springer Berlin Heidelberg, Berlin, Heidelberg.

Holland, J. H. (1976). Adaptation. In Rosen, R. and Snell, F., editors, *Progress in Theoretical Biology*, volume 4, pages 263–293. Academic Press, New York.

Holland, J. H., Booker, L. B., Colombetti, M., Dorigo, M., Goldberg, D. E., Forrest, S., Riolo, R. L., Smith, R. E., Lanzi, P. L., Stolzmann, W., and Wilson, S. W. (2000). *What Is a Learning Classifier System?*, pages 3–32. Springer Berlin Heidelberg, Berlin, Heidelberg.

Khan, M. I. and Rinner, B. (2012a). Resource coordination in wireless sensor networks by cooperative reinforcement learning. In *PerCom Workshops*, pages 895–900. IEEE.

Khan, U. A. and Rinner, B. (2012b). A reinforcement learning framework for dynamic power management of a portable, multi-camera traffic monitoring system. In *GreenCom*, pages 557–564.

Kovacs, T. (1998). XCS Classifier System Reliably Evolves Accurate, Complete, and Minimal Representations for Boolean Functions. In Chawdhry, P., Roy, R., and Pant, R., editors, *Soft Computing in Engineering Design and Manufacturing*, pages 59–68. Springer London.

Lanzi, P. L., Loiacono, D., Wilson, S. W., and Goldberg, D. E. (2007). Generalization in the XCSF Classifier System: Analysis, Improvement, and Extension. *Evol. Comput.*, 15(2):133–168.

Lewis, P. R., Esterle, L., Chandra, A., Rinner, B., and Yao, X. (2013). Learning to be different: Heterogeneity and efficiency in distributed smart camera networks. In *Proceedings of the 7th IEEE Conference on Self-Adaptive and Self-Organizing Systems (SASO)*, pages 209–218. IEEE Press.

Liu, J., Sridharan, S., and Fookes, C. (2016). Recent advances in camera planning for large area surveillance: A comprehensive review. *ACM Comput. Surv.*, 49(1):6:1–6:37.

Murray, A. T., Kim, K., Davis, J. W., Machiraju, R., and Parent, R. E. (2007). Coverage optimization to support security monitoring. *Computers, Environment and Urban Systems*, 31(2):133–147.

Piciarelli, C., Esterle, L., Khan, A., Rinner, B., and Foresti, G. L. (2016). Dynamic reconfiguration in camera networks: A short survey. *IEEE Transactions on Circuits and Systems for Video Technology*, 26(5):965–977.

Piciarelli, C., Micheloni, C., and Foresti, G. L. (2011). Automatic reconfiguration of video sensor networks for optimal 3d coverage. In *2011 Fifth ACM/IEEE International Conference on Distributed Smart Cameras*, pages 1–6.

Prothmann, H., Rochner, F., Tomforde, S., Branke, J., Müller-Schloer, C., and Schmeck, H. (2008). *Organic Control of Traffic Lights*, pages 219–233. Springer Berlin Heidelberg, Berlin, Heidelberg.

Rinner, B., Winkler, T., Schriebl, W., Quaritsch, M., and Wolf, W. (2008). The evolution from single to pervasive smart cameras. In *Distributed Smart Cameras, 2008. ICDSC 2008. Second ACM/IEEE International Conference on*, pages 1–10.

Rudolph, S., Edenhofer, S., Tomforde, S., and Hähner, J. (2014). Reinforcement learning for coverage optimization through PTZ camera alignment in highly dynamic environments. In *Proceedings of the International Conference on Distributed Smart Cameras, ICDSC '14, Venezia Mestre, Italy, November 4-7, 2014*, pages 19:1–19:6.

Sommer, M., Stein, A., and Hähner, J. (2016a). Ensemble Time Series Forecasting with XCSF. In *2016 IEEE 10th International Conference on Self-Adaptive and Self-Organizing Systems (SASO)*, pages 150–151.

Sommer, M., Stein, A., and Hähner, J. (2016b). Local ensemble weighting in the context of time series forecasting using XCSF. In *2016 IEEE Symposium Series on Computational Intelligence (SSCI)*, pages 1–8.

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.

Stein, A., Maier, R., and Hähner, J. (2017a). Toward Curious Learning Classifier Systems: Combining XCS with Active Learning Concepts. In *Proceedings of the Genetic and Evolutionary Computation Conference Companion*, GECCO '17, pages 1349–1356, New York, NY, USA. ACM.

Stein, A., Rauh, D., Tomforde, S., and Hähner, J. (2017b). Interpolation in the eXtended Classifier System: An architectural perspective. *Journal of Systems Architecture*, 75:79 – 94.

Stein, A., Tomforde, S., Rauh, D., and Hähner, J. (2016). Dealing with Unforeseen Situations in the Context of Self-Adaptive Urban Traffic Control: How to Bridge the Gap? In *2016 IEEE International Conference on Autonomic Computing (ICAC)*, pages 167–172.

Sutton, R. and Barto, A. (1998). *Reinforcement learning: An introduction*, volume 116. Cambridge University Press.

Tamee, K., Bull, L., and Pinngern, O. (2007). Towards Clustering with XCS. In *Proceedings of the 9th Annual Conference on Genetic and Evolutionary Computation*, GECCO '07, pages 1854–1860, New York, NY, USA. ACM.

Valera, M. and Velastin, S. (2005). Intelligent distributed surveillance systems: a review. *Vision, Image and Signal Processing, IEE Proceedings* -, 152(2):192–204.

Watkins, C. J. and Dayan, P. (1992). Q-learning. *Machine learning*, 8(3):279–292.

Wilson, S. W. (1995). Classifier Fitness Based on Accuracy. *Evolutionary Computation*, 3(2):149–175.

Wilson, S. W. (2000). Get Real! XCS with Continuous-Valued Inputss. In Lanzi, P. L., Stolzmann, W., and Wilson, S. W., editors, *Learning Classifier Systems*, volume 1813 of *Lecture Notes in Computer Science*, pages 209–219. Springer Berlin Heidelberg.

Wilson, S. W. (2001). Mining Oblique Data with XCS. In Luca Lanzi, P., Stolzmann, W., and Wilson, S. W., editors, *Advances in Learning Classifier Systems*, volume 1996 of *Lecture Notes in Computer Science*, pages 158–174. Springer Berlin Heidelberg.