Project no.:	**IST-FP6- IP-027657**
Project full title:	**Perception, Action & Cognition through Learning of Object-Action Complexes**
Project Acronym:	**PACO-PLUS**

Deliverable no.: D 6.5

Title of the deliverable: Technical Report/Scientific publication on "Implementation of nSARSA with function approximation in its two stages as given by tasks 6.8.1 and 6.8.2 first on a simple platform and then on ARMAR".

Contractual Date of Delivery to the CEC:	Jan 31st, 2008
Actual Date of Delivery to the CEC:	Jan 31st, 2008
Organisation name of lead contractor for this deliverable:	BCCN
Author(s):	M. Tamosiunaite, F. Wörgötter, T. Asfour
Participant(s):	BCCN, UKA
Work package contributing to the deliverable:	WP6
Nature:	R
Version:	1
Total number of pages:	14
Start date of project:	1st Feb. 2006 **Duration:** 48 month

Project co-funded by the European Commission within the Sixth Framework Programme (2002-2006)		
Dissemination Level		
PU	Public	**X**
PP	Restricted to other programme participants (including the Commission Services)	
RE	Restricted to a group specified by the consortium (including the Commission Services)	
CO	Confidential, only for members of the consortium (including the Commission Services)	

Abstract: This report is compiled from a scientific publication submitted to Biological Cybernetics and a short preliminary report of an extension of the presented method. The first part ("paper") describes how to efficiently emulate reinforcement learning (RL) with function approximation in the continuous 4D action space of a robot arm for learning to reach and the second part ("report") shows how to augment the RL method by human coaching. This work is done in the context of PACO+ to improve motor control of the robot and can supplement methods developed in WPs3 and 4. The presented method deviates from the originally planned use of neural SARSA (nSARSA), as introduced in the last PACO+ review. However, reformulating this into nSARSA is a technical step and could, if desired, be done immediately. Instead much effort has been devoted to the much more pressing question of how to find an efficient RL method for this difficult 4D-problem. We find that the proposed RL method converges in about 20 trials. For coaching we only have preliminary results. It is based on the biological principle of efferent-copy error evaluation and uses exactly the same function approximation framework as the RL-method in the "paper". So far it has been tried only on a 2D problem. We report convergence with coaching often in only 2 trials. Note, as the obtained vector fields are dimension-independent we do not expect major differences in convergence speed for the 4D problem. Hence, it seems very much worthwhile to pursue the coaching method further. Implementation on ARMAR III is on the way.

Keyword list: Efficient RL methods for learning to reach, Coaching.

Learning to reach by reinforcement learning using a receptive field based function approximation approach with continuous actions

Minija Tamosiunaite[1,2], Tamim Asfour[3] and Florentin Wörgötter[1]
[1]Bernstein Center of Computational Neuroscience, University Göttingen, Germany, worgott@bccn-goettingen.de, tomas@bccn-goettingen.de
[2]Department of Informatics, Vytautas Magnus University, Vileikos 8, LT-44404 Kaunas, Lithuania, m.tamosiunaite@if.vdu.lt
[3]Institute for Computer Science and Engineering, Universität Karlsruhe (TH), Germany, asfour@ira.uka.de
m.tamosiunaite@if.vdu.lt;asfour@ira.uka.de;worgott@bccn-goettingen.de

Abstract

In this paper we present a novel, rather simple method which uses reinforcement learning with function approximation to learn approaching a target in 3D-space by a two-joint robot arm system with two DOF each. Function approximation is based on 4-dimensional, overlapping and probabilistically responding kernels (receptive fields) and the state-action space contains about 10000 of these. Different types of reward structures are being compared. For example, reward-on-touching-only against reward-on-approach. Furthermore, forbidden joint configurations are punished. A continuous action space is used. In spite of the large number of states and the continuous action space these reward/punishment strategies allow the system to find a good solution usually within about 20 trials.

Keywords:

1 Introduction

An efficient way to control a robot arm is by visual servoing, where the approach to an object is guided by the visual difference measure between the end-effector of the robot arm and the target object. For an excellent introduction to visual servoing the reader is referred to (Hutchinson et al., 1996). A visual servoing control schema uses the Jacobian matrix to gradually reduce the difference between the end-effector and the target object. However, the Jacobian matrix and its estimation becomes quickly hard as the number of joints and consequently the redundancy of the robot system increases. The high number of degrees of freedom, especially in humanoid robot arms and anthormopophic hands, together with the necessity to handle constraints in joint space (singular configurations, self-collision and mechanical joint limits avoidance) as well as in object space (obstacles) makes visual servoing in highly redundant robot systems a difficult problem.

Several methods exist, some are control-based, (Espiau et al., 1992; Hosoda and Asada, 1994; Hutchinson et al., 1996; Horaud et al., 1998), while others focus on the problem of learning visual servoing or using learning to improve visual servoing, (Shibata and Ito, 1999; Martinez-Marin and Duckett, 2004; Perez and Cook, 2004; Leonard and Jagersand, 2004).

Methods range from conventional neural networks, where some classical work had been performed early by the group of Schulten (Martinetz et al., 1990) and Torras (mainly in the domain of inverse kinematics learning, Ruis de Angulo and Torras (2005b,a)), to the now dominating methods of reinforcement learning (RL). Indeed a very large number of articles has appeared in the last years on RL for learning arm control as well as learning grasping. A large diversity of methods has been suggested exemplary represented by the papers cited here, Arm: Tham and Prager (1993); Kobayashi et al. (2005); Li et al. (2006); Wang et al. (2006); Peters and Schaal (2006); Grasp: Moussa and Kamel (1998); Moussa (2004); Rezzoug et al. (2006). This diversity arises due to the fact that conventional RL methods (Sutton,

1988; Watkins, 1989; Watkins and Dayan, 1992), for which rigorous convergence proofs exist, cannot be directly used as the state-action spaces in robot control are too large and convergence will take far too long, especially when using continuous actions, like here. As a consequence, the state-action value function of the used RL-method needs to be approximated by so-called function approximation methods. This is where the diversity arises as there is a terrifically high number of possible such methods existing (e.g.; Tesauro (1995); Horiuchi et al. (1997); Fukao et al. (1998); Gross et al. (1998); Enokida et al. (1999); Qiang et al. (2000); Takahashi et al. (1999); Takeda et al. (2001); Kabudian et al. (2004); Wiering (2004); van Hasselt and Wiering (2007); Sugiyama et al. (2007) for a textbook discussion see Sutton and Barto (1998)) and it lies at the discretion of the researcher to invent more. Convergence to the optimal solution can in general not be rigorously assured anymore. However, this aspect is of minor importance for a robot application as any trial will have to be interrupted if it is too long. Thus, as soon as these systems reach a "good" trajectory within "reasonable" time, the used method appears acceptable from an applied perspective. As a consequence the trade off between good-trajectory and reasonable-time is in the forefront of considerations here and our paper seeks to contribute a method that can satisfactorily handle this problem in a high-dimensional state-action space.

To this end we will adopt a strategy inspired by the place field system of rats, which is used for navigation learning as suggested by several models (Foster et al., 2000; Arleo and Gerstner, 2000; Strösslin et al., 2005), and use overlapping place fields to structure our state-action space (Tamosiunaite et al., 2008). Rats run on the ground and are, thus, faced with a 2D target problem with a 1DoF motion needing about 500 place fields in a 1x1m arena for good convergence (Tamosiunaite et al., 2008). On the other hand, the simulated arm, we use, operates in a 3D target domain with a 4DoF angle space for which we require 10000 four-dimensional place fields. As a consequence, efficient strategies for structuring the reward space are required, too, without which convergence to a good solution would take too long. Thus, this paper will compare different types of visual (distance dependent) and non-visual (touch, angle configuration) rewards showing that, with this combination of methods, it is generically possible to find a good solution within about 20 trials.

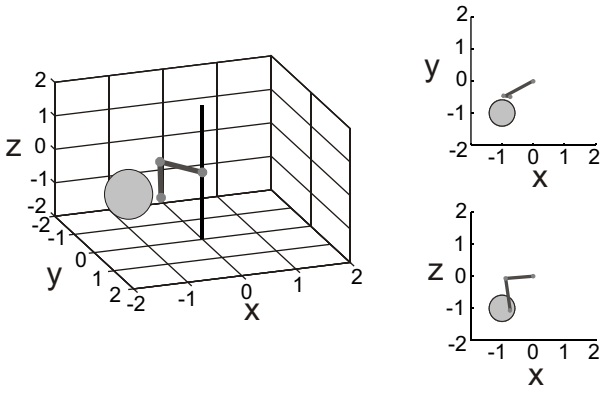

Figure 1: Two-joint arm and an object for reaching in 3D space as well as two projections (x,y) and (x,z) of the arm and the target object for reaching.

2 Methods

2.1 General Setup

Fig. 1 shows the setup of our system in 3D as well as two planar projections. The arm has two joints with two angles each (azimuth $\alpha_{1,2}$ and elevation $\phi_{1,2}$) covering a 3D reaching space

of 4.0^3 units^3 (each arm segment is 1.0 unit long). The target for reaching is a sphere with diameter 0.84 units. Joint angles are constraint between 0 and 360 degrees. Hence, when at a constraint the arm needs to go back. This simulates similar constraints in real (e.g. elbow constraint) or robot arms.

The control and learning circuit used for arm acting and learning is provided in Fig. 2. The four joint angles are considered to form a 4D-state space. In the state space spherical, binary 4D-kernels Φ^k with a diameter between 1382 and 3358 (uniform distribution) are formed in a 4D-space of 10000^4 units^4. Those kernels, of which we use 10000, have eight trainable connections to the motor units (total of 80000 connections). Kernels centers are distributed with a uniform distribution and a coverage of about $10 - 12$ kernels overlapping for any given 4D-location is obtained. A kernel is active, with an output value of 1, as soon as the 4D-joint angle vector falls into the topographical region covered by the kernel, otherwise kernel output is zero. As kernels are overlapping, total activity is then given by the average over active kernels (see definition of Q-values below).

According to the trajectory forming strategy (see below) the actual movement is then generated from the motor unit activity. The trajectory forming strategy includes, for example, the exploration-exploitation trade-off and trajectory smoothing (if applicable).

The employed learning scheme uses a certain type of Q learning with state-action value function approximation, where function approximation is introduced through the fact, that we are not using discrete states, but state information is, instead, provided by the activity of the kernels (commonly called 'feature vectors' in the reinforcement learning literature) as described above.

Lets use the proximal joint (subscript "1"), and its azimuth component α_1 to explain our methods. Explanation is the same for other angular components. We will be operating with Q values, and for the component α_1 two Q-values will be defined: $Q_{\alpha_1}(s, i)$ and $Q_{\alpha_1}(s, d)$, where i and d denote two possible actions: increase and decrease of the component, while s denotes a state. Q values are obtained through function approximation:

$$Q_{\alpha_1}(s, \{i, d\}) = \sum_{k=1}^{N} \theta^k_{\alpha_1}(\{i, d\})\Phi^k(s) / \sum_{k=1}^{N} \Phi^k(s) \tag{1}$$

where $\Phi^k(s)$ is the activation function of the k-th kernel (either 0 or 1 in our case) in state s, $\theta^k_{\alpha_1}(\{i, d\})$ are the weights from the k-th kernel to the motor unit either for action i or d, and N is the overall number or kernels in the system. Weights θ are adapted through learning as described in the subsection on Q learning below.

Next we calculate the difference $\Delta_{\alpha_1}(s)$ used for continuous action formation:

$$\Delta_{\alpha_1}(s) = Q_{\alpha_1}(s, i) - Q_{\alpha_1}(s, d) \tag{2}$$

Movement direction D is given by the sign function $D_{\alpha_1} = sign(\Delta_{\alpha_1})$ and movement amplitude A by the absolute value normalized against the other components, which are calculated in the same way (now omitting s for simplicity):

$$A_{\alpha_1} = \frac{\mid \Delta_{\alpha_1} \mid}{\sqrt{\Delta_{\alpha_1^2} + \Delta_{\alpha_2^2} + \Delta_{\phi_1^2} + \Delta_{\phi_2^2}}} \tag{3}$$

Normalization leads to a situation where the movement vector is formed proportional to the increase/decrease values, but the total length of the vector is kept fixed. This is done to preserve the topology of the space in which learning takes place.

To cover a full sphere, normally the polar representation for azimuth is defined over 360° and for elevation over 180°. With the methods used here both, azimuth and elevation, are

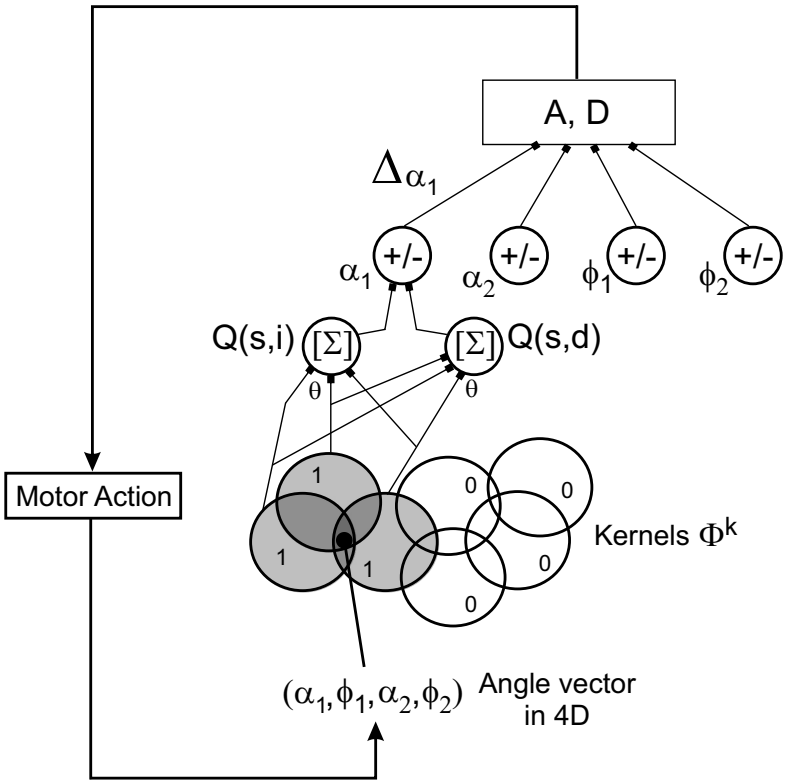

Figure 2: Schematic diagram of learning and action generation in our system. The angle vector (bottom) represents a location in 4D space and, thus, stimulates the gray kernels which fire while the white kernels are not stimulated and do not fire. The layers of units above calculate the finally required values A and D which control the motors and move the joints to a new position.

defined over 360° leading to duplicate angular space and, hence, an over-representation. Yet it is unnecessary to introduce any compensatory space transformation as the learning overcomes the over-representation anyways. For the same reason, the action space is duplicate, too, but this has the nice aspect of allowing us to treat actions at the angular constraints in a continuous way avoiding having to deal with wrap-around problems. Noise of around 7% of the movement amplitude was added to each step to imitate imperfections in the control of the angle in a real robot.

Movement strategies with smoothing were employed. All new steps in angle space were made as a combination of the currently calculated steps $\Delta_{current} = (\Delta\alpha_1, \Delta_{\alpha_2}, \Delta_{\phi_1}, \Delta_{\phi_2})_{current}$ combined with the previous steps $\Delta_{previous} = (\Delta_{\alpha_1}, \Delta_{\alpha_2}, \Delta_{\phi_1}, \Delta_{\phi_2})_{previous}$ using:

$$\Delta_{final} = c\Delta_{current} + (1-c)\Delta_{previous} \tag{4}$$

where $c = 0.6$ was used. Smoothing helps to avoid jerky movements in the beginning of learning and also influences the process of learning, as in Q learning with function approximation, which we are using here, the convergence pattern is dependent on the paths performed during learning.

Finally, if a newly calculate angle falls outwith the constraint boundaries, then random steps with uniform distribution in angle space are tried out until a point satisfying the constraints is obtained.

2.2 Learning method

From eight possible actions only four learn during a single step. These are the directions along which the actual movement has been taken. For example, if a particular angle has been increased, the unit, which has driven the increase, learns while the antagonistic 'decrease unit' will not be changed. Furthermore, learning will affect connections of all currently active kernels. Again we will demonstrate the learning rule for components representing angle α_1.

Weight update follows an algorithm similar to conventional Q-learning (Watkins, 1989; Watkins and Dayan, 1992). For each angle, two actions $a_{i,d}$ are possible: increase or decrease. Let us say the current state is described by angles $s = (\alpha_1, \phi_1, \alpha_2, \phi_2)_{current}$ and some action a (increase or decrease) is chosen that leads to a state $s' = (\alpha_1, \phi_1, \alpha_2, \phi_2)_{next}$. In our learning framework, the change in the value $\theta_{\alpha_1}(\{i, d\})$ of that state-action pair follows the mean across all activated kernels of s':

$$\theta^k_{\alpha_1}(\{i,d\}) = \theta^k_{\alpha_1}(\{i,d\}) + \mu[r + \gamma Q_{\alpha_1}(s', \{i,d\}) - \theta^k_{\alpha_1}(\{i,d\})]\Phi^k(s) \tag{5}$$

where k is the number of the kernel to which the weight is associated, r is the reward, $\mu < 1$ the learning rate, $\gamma < 1$ the discount factor, $\Phi^k(s)$ is the activity function for kernel k in state s, $Q_{\alpha_1}(s', \{i, d\})$ is the Q value of the system in the next state, either for action i or d defined as:

$$Q_{\alpha_1}(s', \{i,d\}) = \sum_{k=1}^{N} \theta^k_{\alpha_1}(\{i,d\})\Phi^k(s') / \sum_{k=1}^{N} \Phi^k(s') \tag{6}$$

Equivalent expressions are used for updating Q values for the angles ϕ_1, α_2 and ϕ_2.

Note, the algorithm is not so easy to attribute to standard $Q-$ or to SARSA-learning. It is neither SARSA, because it does not take into account where the system has gone in the next step; nor is it Q-learning, as it does not consider which action is best in the next step. The algorithm rather considers if continuing along the current direction is valuable according to the Q-values of the next state $Q_{\alpha_1}(s', \{i, d\})$.

Finally, to reduce computing time, we have introduced a variable path-length limit. If the arm has not reached the target within this limit, it is reset to the start and all changes in Q-values learned during this trial are undone. For a new experiment, path length limit starts with $l(1) = 1000$, and further develops according to the following recurrent equation:

$$l(n+1) = \begin{cases} 2.5 * k(n) & \text{if goal obtained in n}^{\text{th}} \text{ trial} \\ l(n) + 20 & \text{if goal not obtained in n}^{\text{th}} \text{ trial.} \end{cases} \tag{7}$$

where $k(n)$ is the number of steps to goal in trial n. That is, the limit comparable to the previous path to the goal is imposed once the goal has been obtained, but if the goal is not obtained in some trial, then the limit is relaxed by 20 units.

2.2.1 Reward Structure

Different rewards have been used, alone and in combination, in this study (Fig. 3) normalized to a maximal reward of 1. Highest reward is (usually) obtained on touching the target (Fig. 3 A), but we also define a vision-based reward structure (Fig. 3 B). For this, we assume that the distance between tip of the arm and target can be evaluated. Distance dependent reward is then defined linearly along this distance. Different combinations of distant dependent and touch rewards are shown in panels C1 and C2 of Fig. 3. Panel E shows an approach distance (differential) dependent reward structure were reward is only given if the arm has approached the target, while reward is generally zero if the arm has moved away. These reward structures are using absolute distance x to define the reward. By contrast, we have also used a differential, relative reward structure (Fig. 3 F). For this we first normalize the distance to target at the

start of an experiment to one. Then reward is given according to the *relative* approach d. Hence a larger step towards target was rewarded more than a smaller one, where absolute distance does not play any role. In the following we will always refer to the different rewards used in individual experiments using the panel labels from Fig. 3.

Furthermore, we also introduced punishments (negative rewards) for approaching the constraint boundaries. For this, we were increasing punishment linearly, starting with zero at a distance of 1/20th of the field width to the border up to a certain maximal value reached when touching the border. Maximal values were changed in different experiments and details will be given below.

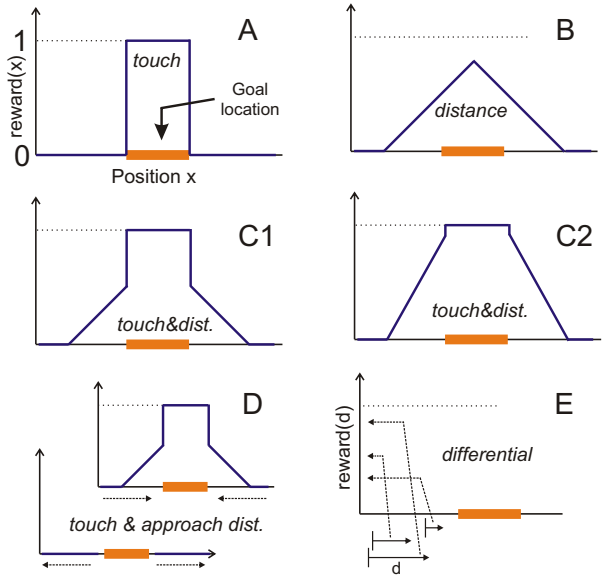

Figure 3: Schematic representation of different types of rewards used in this study. The bar in the middle of the x-axis represents the target location. Reward can be given only there (A) or on approach (B) or combining approach with touch (C1,C2). Panel D depicts a situation where reward is given on approach (top) but not on leaving (bottom, see direction of the arrows). Panel E shows a purely differential reward calculated by how far the arm has moved forward to the target.

3 Results

3.1 Individual convergence patterns

The main aim of the paper is to demonstrate condensed statistical results that help comparing convergence of learning processes using various reward structures and other learning parameters. However, before presenting those statistics, we will show examples of individual training sessions to provide a better intuition about the behavior of this system. In Fig. 4 five qualitatively different cases of system development are shown: (A) quick convergence, (B) delayed convergence, (C) intermittent episode of divergence, (D) bad convergence; and (E) the persistently falling back onto a wrong trajectory, when the optimal trajectory has already been learned (E). The bars shown in black demonstrate successful runs; the bars shown in gray are for unsuccessful runs, where the limit for the path length was exhausted before the goal was reached. For these experiments, a D type reward (Fig. 3) has been used. It is of interest to make a practical remark here: Cases shown in Fig. 4 panels C and E are largely irrelevant

from a practical point of view as any robot learning system of this kind would either have to be controlled by an operator or - at least - would require a stopping criterion, which tells the machine to stop as soon as it has found a reasonably good solution. In both cases the longer trajectories, which occur later in (D) or in an interspersed way in (E), would just be ignored (or not even be recorded). Note, Fig. 4 showed five archetypical examples. Many times transitions

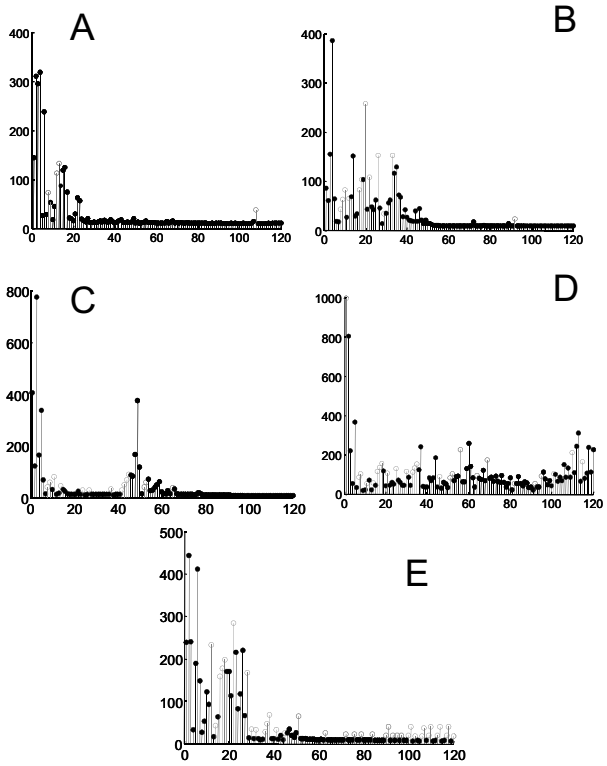

Figure 4: Different example runs. A) Quick convergence to a good trajectory within about 22 *trials. B) Delayed convergence (≈* 50 *trials). C) Quick convergence in less than* 20 *trials and thereafter some divergence and re-convergence to a different trajectory. Note from a practical perspective the robot would just stop learning after about trial* 20 *and the second phase is not of relevance. D) Badly convergent case. E) Good convergence but with interspersed other, longer trajectories (light shading). Also here the robot would normally stop learning after having found the better trajectory and ignore the others.*

between them exist, which cannot easily be distinguished. As a consequence, from now on we will not separate these cases anymore and perform statistical analyzes always across whole data sets. In the following for every experiment we train the system for 20 trials and then for another 100 trials we will determine the average trajectory length and the percentage how often the system has reached the goal. AS this is indicative of the average convergence property of a certain setup, it will allow us to compare strategies. In general we repeat every experiment 50 times.

Observations show that for best learning regimes divergent cases never occur, but various intermittent phenomena, or delayed convergence were present. For bad regimes about three to five divergence cases happen during 50 experiments, and quick and clearly convergent cases are relatively rare (e.g. 15 out of 50), the biggest proportion being taken by delayed convergence or by cases with intermittent behaviors.

3.2 Comparison of learning modes

In the simplest case reinforcement is only given when the target is actually hit (reward type A). Note, this would correspond to a task of finding an unknown object in the dark. As touching is a singular, rare event, this leads to a pronounced temporal credit assignment problem for large state-action spaces such as here. Thus, this case will only be considered for comparison reasons and not analyzed any further.

Of much more relevance for robotic applications is the case where visual information is used to position the arm (visual servoing) and this can be encoded by a distance dependent reward structure (reward type B). On touching additional reward may be given (reward type C). Furthermore, it makes sense to avoid giving a distance dependent reward when the arm has actually moved away from the target (reward type D). First, experiments to compare influence of these three different basic reward structures were performed. In Fig. 5 A the average number of steps to target is shown and in (B) the percentage of cases reaching the goal is presented. The bars show the standard deviation of the distributions. We observe that in the case where vision is excluded the trajectories of the arm are around 10 times longer as compared to cases with vision. Standard deviation is 218 steps, and is clipped to preserve the scale of the figure. Similarly, the percentage of attaining the goal without vision is only around 50%. This happens because in 3D space the target only takes a small part of the total space and it is not easy to hit it by chance. Furthermore, due to the singular reward structure, kernels with non-zero Q-values after around 100 runs are still too sparse to give clear indications about preferred trajectories in the 4D angle space. Runs for distance rewards (middle column) are substantially longer and the percentage of successful trials is smaller, as compared to the case that includes the approach distant dependent component (right column). Thus, in the following, we will focus our attention on cases with a approach distant dependent reward component, as these have noticeable better convergence properties compared to the others.

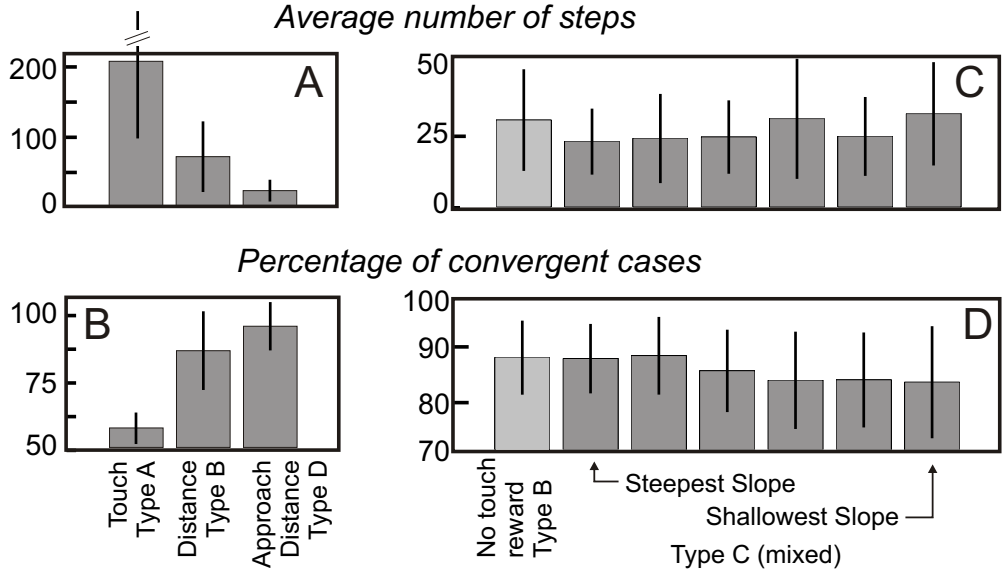

Figure 5: Statistics for different reward types. A,B) Two distance dependent reward types (right) compared to touch-only reward (left). Annotations at the bottom refer to the annotations introduced in Fig. 3. C,D) Comparison of different slopes for distance dependent rewards. For C,D slope values from left to right are: 1/3 (no touch reward case), 1/3, 1/4, 1/5, 1/10, 1/20, 1/50, reaching values at the goal boundary correspondingly: 0.86, 0.65, 0.52, 0.26, 0.13, 0.05. Touch reward is 1.

Next we analyze the steepness of the distance dependent reward curve in Fig. 5 C,D. Reward for reaching the target was kept at one and the slope of the distant dependent reward was varied from 0.86 (almost reaching one, the value of the touch reward) at the border of the target for the steepest slope to down to 0.05 for the shallowest slope. One can observe that for steep slopes (C,D left) values are quite similar and better than for shallow slopes. The first column shows the limiting case when no special reward for touching was applied. Here the entire process of approaching the target was regulated using exclusively distance-dependent rewards. This case in terms of convergence appears among the best cases, but step length was longer than for some of the mixed cases. Thus, for further experiments we were choosing slope steepness 0.25, which seems a good compromise. Using this steepness parameter, more reward structures were

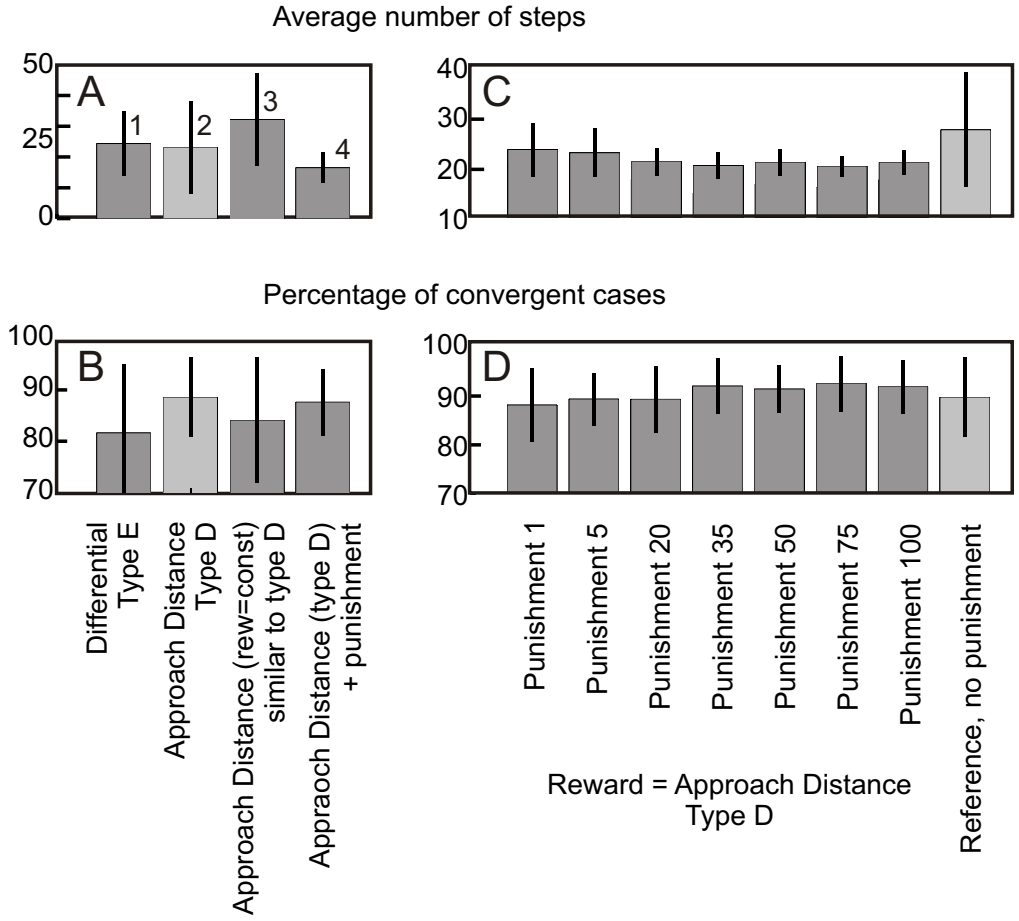

Figure 6: Statistics for other types of rewards. A,B) As per annotation at the bottom (see Fig. 3). C,D) Reward Type D is combined with punishment on approaching the maximally allowed joint angles. Note light gray shaded cases are identical across panels.

explored in Fig. 6 A,B. In the columns from left to right, we compare (1) differential reward (type E), (2) approach distance (type D), (3) a variant of type D, where we always give the same amount of reward $r = 0.3$, and (4) type D but applying punishment of the same size as reward on approach would be, if the action leads away from the goal by more than a certain distance. In general results are very similar where the rightmost column is slightly better than the others.

When applying distant dependent rewards, the problem arises that the joint might be drawn to go towards some direction, though this movement is not allowed due to physical limitations of

the joint construction. For example here the joints were not allowed to cross the border of $0-360$ degrees. The position and size of the reward was chosen such that again and again learning tried to violate these constraints. This problem can be mitigated best by applying punishment also at the constraints as described in the methods section. This additional mechanism leads to a noticeable improvement of convergence times (Fig. 6 C) and a much higher percentage of convergent cases (Fig. 6 D).

In the columns from left to right in Fig. 6 C,D maximal punishment is 1, 5, 20, 35, 50, 75 and 100. The rightmost column shows a control experiment without punishment. Reward structure was the same as in panels A,B column 2. Hence gray shaded columns in panels A,B are the same as those in C,D.

One can observe that punishment at constraints in general improves rate and speed of convergence. Performance of learning first increases with increasing punishment values and approaches an optimum around a punishment of 35 to 75 and then slightly drops again. Of importance, however, is that punishment at constraint works rather reliably over a large parameter range *and* that the variances have been drastically reduced. Hence quick convergence became rather reliable with this mechanism.

Combining constraint-based punishment, specifically employing the case with the maximal punishment value 35, with punishment for going away from the reward, as described before (column 4 in panels A,B), did not provide better results, remaining in the range of an average of $13-14$ steps to the goal with around 90% of successful trials.

The system was also tested with changing the position of the goal, and reverting positions of start and goal, to exclude possibilities of some special circumstances which were only providing good performance for one specific task. Trials with changed position were providing similar average trajectory length, and success percentage results.

4 Discussion

In this paper we have developed a novel type of reinforcement learning, similar but not identical to Q-learning and use it in a continuous action space. This method has been combined with a kernel based approach for value function approximation inspired by the place-field system in the hippocampus of rodents (Tamosiunaite et al., 2008). Our focus lay on a comparison of different reward structures and we could show that a combination of distance dependent rewards with constrain punishment and extra reward on touching will lead to a fast convergence to "a good" trajectory.

Of specific interest is that our formalism is very simple and can still handle large state-action spaces in a moderately high-dimensional problem. Currently this method is being implemented on ARMAR III, a humanoid robot built by the University of Karlsruhe (Asfour et al., 2006). Each arm has seven degrees of freedom: three DOF in the shoulder, two DOF in the elbow and two DOF in the wrist. For this purpose, a control schema of the arm is realized which makes use of hypothesis and results from neurophysiology on human movement control (Soechting and Flanders, 1989b,a). Soechting and Flanders have shown that arm movements are planned in shoulder-centered spherical coordinates and suggest a sensorimotor transformation model that maps the wrist position on a natural arm posture using a set of four parameters, which are the elevation and azimuth (yaw) of both the upperarm and forearm similar to the problem addressed in the current paper. Once these parameters are obtained for a given position of the wrist, they are mapped on the joint angles of the shoulder and elbow of the robot arm, which completely define the wrist position. For more details the reader is referred to (Asfour and Dillmann, 2003).

A central problem for the use of RL in large state-action spaces is that convergence of conventional methods will take far too long. Thus, value function approximation is required

for which a wide variety of methods exists. Alas, only few of them have proven convergence properties (Sutton and Barto, 1998; Gordon, 2001; Reynolds, 2002; Szepesvari and Smart, 2004) and those are in praxis many times not the fastest ones. Hence, the field of applied robotics is largely uninterested in the mathematical rigor of a given method as long as the trade-of is reasonable between goodness of solution and time to find it. From a practical perspective, on any humanoid robot learning will be stopped by force as soon as a predefined variable, which handles this trade-off, approaches a desired threshold.

The control of a multi-joint arm poses a terrific problem for RL methods as the state-action spaces are in general very high, needing to cover a space-time continuous system within which redundant solutions exist. While a very large number of papers exist on space-time continuous low-dimensional problems (e.g.; Qiang et al. (2000); Gross et al. (1998); Gaskett et al. (2000)). Only few articles try, to tackle this problem with a reasonable degree of complexity in a high dimensional system (Perez and Cook, 2004). Also many times action spaces of the problems solved are discrete or quite limited (Enokida et al., 1999; van Hasselt and Wiering, 2007; Martinez-Marin and Duckett, 2004) and, even if continuous state representations are used, actions are kept discrete in continuous task approximations (Enokida et al., 1999; Li et al., 2006). This is possible when action spaces are small, but with bigger action spaces (4D in our case) the usage of discrete actions becomes impractical. If continuous actions are to be used, often methods for the interpolation of discrete samples are employed relying on weighted sums or wire fitting (Gaskett et al., 2000; Takeda et al., 2001; van Hasselt and Wiering, 2007). Instead, here we are offering a generic approach to continuous action formation, where reinforcement learning is implemented on the features defining continuous actions.

In our models we deal with position based visual servoing (PBVS) (Hutchinson et al., 1996), however assuming that the inverse kinematics is not known and that the arm has to learn to target an object without this knowledge. The main theoretical development of the current study is the analysis of different reward structures for RL in visual servoing tasks. Robotic applications tend to be multidimensional, and if a reward is provided only at the goal, then additional means to guide a robot to the target are required, like 'learning from easy missions' (hence, bootstrapping the system with simple tasks) or using simpler controllers (like teachers, Martinez-Marin and Duckett (2004)), otherwise learning times would be impractically long. In image based visual servoing (IBVS) systems, richer reward structures were many times used, attempting to achieve convergence in bigger state spaces (Leonard and Jagersand, 2004; Perez and Cook, 2004). We analyzed and compared about half a dozen of vision-based reward structures and studies addressing applications of visual servoing could gain from that comparison using the structures that performed best according to our modeling. Although we are investigating these structures in the framework of PBVS, similar ones could be applied to IBVS, using visual error instead of distance. Possibly some adaptations of the method we used here would be required, as mapping from distance to visual features is rather complex.

Thus, in summary our method is meant to provide a solution with some practical relevance for robotics, but, given the kernel structure, it also possible to combine it with a newly developed biophysical framework for implementing Q- or SARSA learning by an LTP/LDT based differential Hebbian framework (Wörgötter and Porr, 2005; Kolodziejski et al., 2008). Hence, from a biophysical point of view receptive field based (i.e., kernel based) function approximation could indeed operate in a hebbian network emulating not just correlation based unsupervised, but also reward based reinforcement learning.

5 Acknowledgements

This work was supported by the European IP Grant PACO-PLUS.

References

Arleo, A. and Gerstner, W. (2000). Spatial cognition and neuro-mimetic navigation: a model of hippocampal place cell activity. *Biological Cybernetics*, 83(3):287–299.

Asfour, T. and Dillmann, R. (2003). Human-like motion of a humanoid robot arm based on a closed-form solution of the inverse kinematics problem. In *IEEE/RSJ International Conference on Intelligent Robots and Systems*.

Asfour, T., Regenstein, K., Azad, P., Schröder, J., Vahrenkamp, N., and Dillmann, R. (2006). ARMAR-III: An Integrated Humanoid Platform for Sensory-Motor Control. In *IEEE/RAS International Conference on Humanoid Robots*.

Enokida, S., Ohashi, T., Yoshida, T., and Ejima, T. (1999). Stochastic field model for autonomous robot learning. In *IEEE International Conference on Systems, Man, and Cybernetics*, volume 2, pages 752–757.

Espiau, B., Cahumette, F., and Rives, P. (1992). A new approach to visual servoing in robotics. *IEEE Transactions on Robotics and Automation*, 8(3):313–326.

Foster, D. J., Morris, R. G., and Dayan, P. (2000). A model of hippocampally dependent navigation, using the temporal difference learning rule. *Hippocampus*, 10(1):1–16.

Fukao, T., Sumitomo, T., Ineyama, N., and Adachi, N. (1998). Q-learning based on regularization theory to treat the continuous states and actions. In *IEEE International Joint Conference on Neural Networks*, pages 1057–1062.

Gaskett, C., Fletcher, L., and Zelinsky, A. (2000). Reinforcement learning for a vision based mobile robot. In *IEEE/RSJ International Conference on Intelligent Robots and Systems*, pages 403–409.

Gordon, G. J. (2001). Reinforcement learning with function approximation converges to a region. *Advances in Neural Information Processing Systems*, 13(6):1040–1046.

Gross, H., Stephan, V., and Krabbes, M. (1998). A neural field approach to topological reinforcement learning in continuous action spaces. In *IEEE World Congress on Computational Intelligence and International Joint Conference on Neural Networks, Anchorage, Alaska*, pages 1992–1997.

Horaud, R., Dornaika, F., and Espiau, B. (1998). Visually guided object grasping. *IEEE Transactions on Robotics and Automation*, 14(4):525–532.

Horiuchi, T., Fujino, A., Katai, O., and Sawaragi, T. (1997). Fuzzy interpolation-based Q-learning with profit sharing planscheme. In *Proceedings of the Sixth IEEE International Conference on Fuzzy Systems*, volume 3, pages 1707–1712.

Hosoda, K. and Asada, M. (1994). Versatile visual servoing without knowledge of true jacobian. In *IEEE/RSJ International Conference on Intelligent Robots and Systems*.

Hutchinson, S. A., Hager, G. D., and Corke, P. I. (1996). A tutorial on visual servo control. *IEEE Trans. Robotics and Automation*, 12(5):651–670.

Kabudian, J., Meybodi, M. R., and Homayounpour, M. M. (2004). Applying continuous action reinforcement learning automata(carla) to global training of hidden markov models. In *Proceedings of the International Conference on Information Technology: Coding and Computing*, volume 4, pages 638–642, Washington, DC, USA. IEEE Computer Society.

Kobayashi, Y., Fujii, H., and Hosoe, S. (2005). Reinforcement learning for manipulation using constraint between object and robot. In *IEEE International Conference on Systems, Man and Cybernetics*, volume 1, pages 871–876.

Kolodziejski, C., Porr, B., and Wörgötter, F. (2008). On the equivalence between differential hebbian and temporal difference learning. *Neural Networks (submitted)*.

Leonard, S. and Jagersand, M. (2004). Learning based visual servoing. In *International Conference on Intelligent Robots and Systems, Sendai, Japan*, pages 680–685.

Li, J., Lilienthal, A. J., Martnez-Marn, T., and Duckett, T. (2006). Q-ran: A constructive reinforcement learning approach for robot behavior learning. In *Proceedings of the IEEE/RSJ International Conference on Intelligent Robots and Systems*, pages 2656–2662.

Martinetz, T. M., Ritter, H. J., and Schulten, K. J. (1990). Three-dimensional neural net for learning visuomotor coordination of a robot arm. *IEEE Transactions on Neural Networks*, 1(1):131–136.

Martinez-Marin, T. and Duckett, T. (2004). Robot docking by reinforcement learning in a visual servoing framework. In *IEEE Conference on Robotics, Automation and Mechatronics*, volume 1, pages 159–164.

Moussa, M. (2004). Combining expert neural networks using reinforcement feedback for learning primitive grasping behavior. *IEEE Transactions on Neural Networks*, 15(3):629–38.

Moussa, M. and Kamel, M. (1998). An experimental approach to robotic grasping using a connectionist architecture and generic grasping functions. *IEEE Transactions on Systems, Man, and Cybernetics, Part C: Applications and Reviews*, 28:239–253.

Perez, M. A. and Cook, P. A. (2004). Actor-critic architecture to increase the performance of a 6-dof visual servoing task. In *IEEE 4th International Conference on Intelligent Systems Design & Application, Budapest*, pages 669–674.

Peters, J. and Schaal, S. (2006). Reinforcement learning for parameterized motor primitives. In *International Joint Conference on Neural Networks*, pages 73–80.

Qiang, L., Hai, Z. H., Ming, L. L., and Zheng, Y. G. (2000). Reinforcement learning with continuous vector output. In *IEEE International Conference on Systems, Man, and Cybernetics*, volume 1, pages 188–193.

Reynolds, S. I. (2002). The stability of general discounted reinforcement learning with linear function approximation. In *UK Workshop on Computational Intelligence (UKCI-02)*, pages 139–146.

Rezzoug, N., Gorce, P., Abellard, A., Khelifa, M. B., and Abellard, P. (2006). Learning to grasp in unknown environment by reinforcement learning and shaping. In *IEEE International Conference on Systems, Man and Cybernetics*, volume 6, pages 4487–4492.

Ruis de Angulo, V. and Torras, C. (2005a). Speeding up the learning of robot kinematics through function decomposition. *IEEE Transactions on Neural Networks*, 16(6):1504–1512.

Ruis de Angulo, V. and Torras, C. (2005b). Using psoms to learn inverse kinematics through virtual decomposition of the robot. In *International Work-Conference on Artificial and Natural Neural Networks (IWANN)*, pages 701–708.

Shibata, K. and Ito, K. (1999). Hand-eye coordination in robot arm reaching task by reinforcement learning using a neural network. In *IEEE International Conference on Systems, Man, and Cybernetics*, volume 5, pages 458–463.

Soechting, J. and Flanders, M. (1989a). Errors in pointing are due to approximations in targets in sensorimotor transformations. *Journal of Neurophysiology*, 62(2):595–608.

Soechting, J. and Flanders, M. (1989b). Sensorimotor representations for pointing to targets in three-dimensional space. *Journal of Neurophysiology*, 62(2):582–594.

Strösslin, T., Sheynikhovich, D., Chavarriaga, R., and Gerstner, W. (2005). Robust self-localisation and navigation based on hippocampal place cells. *Neural Networks*, 18(9):1125–1140.

Sugiyama, M., Hachiya, H., Towell, and Vijayakumar, S. (2007). Value function approximation on non-linear manifolds for robot motor control. In *IEEE International Conference on Robotics and Automation*, pages 1733–1740.

Sutton, R. and Barto, A. (1998). *Reinforcement Learning: An Introduction.* MIT Press, Cambridge, MA.

Sutton, R. S. (1988). Learning to predict by the methods of temporal differences. *Machine Learning*, 3:9–44.

Szepesvari, C. and Smart, W. D. (2004). Interpolation-based Q-learning. In *Twenty-First International Conference on Machine Learning (ICML04)*, volume 21, pages 791–798.

Takahashi, Y., Takeda, M., and M, A. (1999). Continuous valued q-learning for vision-guided behavior. In *Proceedings of the IEEE/SICE/RSJ International Conference on Multisensor Fusion and Integration for Intelligent Systems*, pages 255–260.

Takeda, M., Nakamura, T., and Ogasawara, T. (2001). Continuous valued Q-learning method able to incrementally refine state space. In *IEEE/RSJ International Conference on Intelligent Robots and Systems*, volume 1, pages 265–271.

Tamosiunaite, M., Ainge, J., Kulvicius, T., Porr, B., Dudchenko, P., and Wörgötter, F. (2008). Path-finding in real and simulated rats: On the usefulness of forgetting and frustration for navigation learning. *Journal of Computational Neuroscience*, in press.

Tesauro, G. (1995). Temporal difference learning and TD-gammon. *Communications of the ACM*, 38(3):58–67.

Tham, C. and Prager, R. (1993). Reinforcement learning methods for multi-linked manipulator obstacle avoidance and control. In *Proceedings of the IEEE Asia-Pacific Workshop on Advances in Motion Control, Singapore*, pages 140–145.

van Hasselt, H. and Wiering, M. (2007). Reinforcement learning in continuous action spaces. In *IEEE International Symposium on Approximate Dynamic Programming and Reinforcement Learning*, pages 272–279.

Wang, B., Li, J., and Liu, H. (2006). A heuristic reinforcement learning for robot approaching objects. In *IEEE Conference on Robotics, Automation and Mechatronics*, pages 1–5.

Watkins, C. J. (1989). *Learning from Delayed Rewards.* . PhD thesis, Cambridge University.

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

Wiering, M. (2004). Convergence and divergence in standard averaging reinforcement learning. In Boulicaut, J., Esposito, F., Giannotti, F., and Pedreschi, D., editors, *Proceedings of the 15th European Conference on Machine learning ECML'04*, pages 477–488.

Wörgötter, F. and Porr, B. (2005). Temporal sequence learning, prediction, and control: a review of different models and their relation to biological mechanisms. *Neural Computation*, 17(2):245–319.

Coaching based on teacher-learner signal difference: Preliminary Report

Minija Tamosiunaite[1,2], Tamim Asfour[3] and Florentin Wörgötter[1]
[1]Bernstein Center of Computational Neuroscience, University Göttingen, Germany,
worgott@bccn-goettingen.de, tomas@bccn-goettingen.de
[2]Department of Informatics, Vytautas Magnus University, Vileikos 8, LT-44404 Kaunas,
Lithuania, m.tamosiunaite@if.vdu.lt
[3]Institute for Computer Science and Engineering, Universität Karlsruhe (TH), Germany,
asfour@ira.uka.de
m.tamosiunaite@if.vdu.lt;asfour@ira.uka.de;worgott@bccn-goettingen.de

Abstract

This is an early report for the use in the PACO-PLUS consortium concerning how to augment the RL-based learning methods for trajectory learning by human coaching. This is a consequential continuation of the plain RL approach presented so far. We look at a 2D problem here and do not yet provide any statistics. Still, these first results show that this coaching based approach converges extremely quickly and seems worthwhile to pursue in the future.

1 Introduction

An agent was coached using the difference signal between expectation of the agent where it should end up making a movement and the actual position where the agent was lead by the coach. Hence, this relates to efferent copy learning comparing the expected motor outcome with an actual one and using this as the error signal for learning. The approach was tried out with navigation learning (hence not by using the arm similuation) as actions of the navigating agent as well as vector fields of 2D movement are easy to visualize and understand. The agent was performing a navigation task in a square arena discretized 10000 $\times$ 10000 units, where the start position was (1000,5000), and the goal was a square with a side of 1000 units, centered at the location $(7500, 7500)$, see Fig. 1. As a coach we were using another agent that had learned the task before through regular reinforcement learning. Note, this is just a convenient way to generate a teacher and is of no other practical relevance.

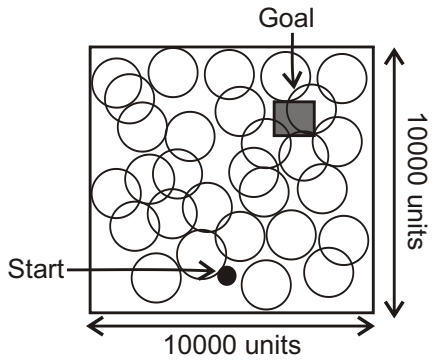

Figure 1: Schematic diagram of the arena with start and goal positions, as well as kernels used for function approximation.

2 Methods

2.1 General Setup

Two sets of Q-values were used: Q^t for a teacher, and Q^l for a learner. Two directions X and Y were considered to form the state space. In the state space 2D-kernels with scaled Gaussian activation probability were introduced:

$$p(\delta_k) = A \exp(-\delta_k^2 / 2\sigma^2) \tag{1}$$

where δ_k is a distance of an agent to the center of the kernel k, $k = 1, 2, ..., N$, where N is the overall number of kernels in the system, σ is a standard deviation of the distribution which was kept at 283 units in the current study, and A is a scaling factor which was 2.5. The probability was cut to one, if the scaled value was above one. Teacher and learner were having different kernel systems. $N = 500$ kernels were used both for a teacher and a learner. Those kernels have trainable connections to the motor units (Fig. 2). If a kernel is active at a given location (x, y) at a given moment, then it produces an output value of one, otherwise kernel output is zero. As kernels are overlapping, total activity is then given by the average over active kernels (see definition of Q-values below). Kernel centers are distributed with uniform distribution and a coverage of about 4.5 kernels overlapping for any given 2D-location is obtained.

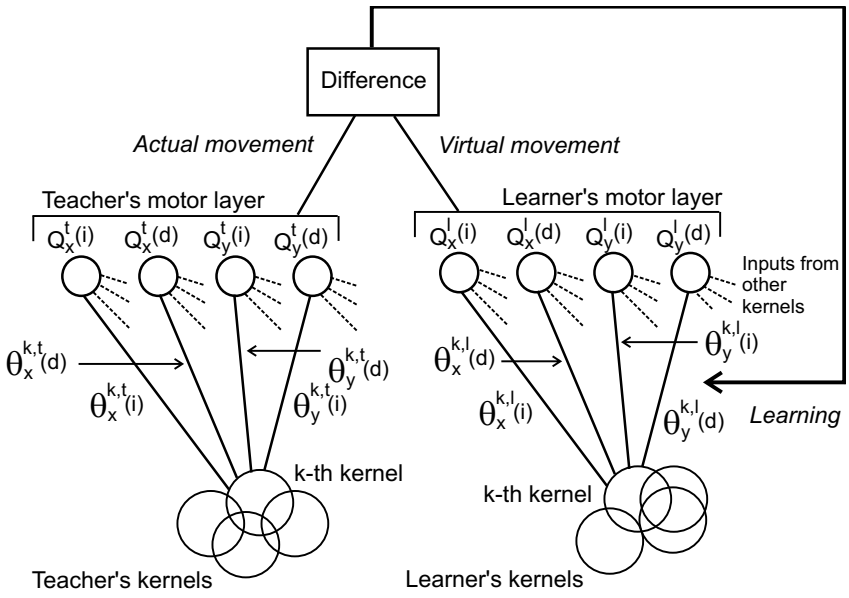

Figure 2: Scheme of the network. By θ are denoted the weights of a teacher and a learner, by Q are denoted the motor outputs of a teacher and a learner. The learners motor output is only used to produce a virtual movement and it is overruled by the teacher's output. According to the difference between actual and virtual movement the learning signal is derived.

According to the trajectory forming strategy (see below) the actual movement is then generated from the teacher motor unit activity Q^t (in the training phase), while a 'virtual' movement as expected by the learner is generated from the activity of Q^l. The trajectory forming strategy includes the exploration-exploitation trade-off and trajectory smoothing.

The employed learning scheme uses a certain type of Q learning with state-action value function approximation. Function approximation is introduced through the fact, that we are not using discrete states, but state information is, instead, provided by the activity of the kernels (commonly called 'feature vectors' in the reinforcement learning literature) as described

above. We will be using separate components along two directions: Q_x and Q_y. For the teacher those components will be labeled Q^t_x, Q^t_y, and for the learner Q^l_x, Q^l_y.

Let us use the teacher's values on direction X, Q^t_x to explain our methods. Explanation is the same for the other components involved in the algorithm (Q^t_y, Q^l_x, and Q^l_y). For each direction two Q-values will be defined: $Q^t_x(s,i)$ and $Q^t_x(s,d)$, where i and d denote two possible actions: increase and decrease of the component, while s denotes a state. Q values are obtained through function approximation:

$$Q^t_x(s,\{i,d\}) = \sum_{k=1}^{N} \theta^{t,k}_x(\{i,d\})\Phi^k(s) / \sum_{k=1}^{N} \Phi^k(s) \tag{2}$$

where $\Phi^k(s)$ is the activation function of the k-th kernel (either 0 or 1 in our case) in state s, $\theta^{t,k}_x(\{i,d\})$ are the weights from the k-th kernel to the motor unit either for action i or d, and N is the overall number or kernels in the system. Weights θ are adapted through learning as described in subsection on Q learning below.

We obtain a difference $\Delta^t_x(s)$ for $Q^t_x(s,i)$ and $Q^t_x(s,d)$ that is used for continuous action formation:

$$\Delta^t_x(s) = Q^t_x(s,i) - Q^t_x(s,d). \tag{3}$$

Note, these signals can strictly be understood as motion signals and not just inner variables. Thus, they are accessible as error signals in a general teacher-learner setup where a human teacher would guide a robot (arm) towards a goal. Movement direction D is given by the sign function $D^t_x = sign(\Delta^t_x)$ and movement amplitude A by the absolute value normalized against the y component, which is calculated in the same way:

$$A^t_x = \frac{\mid \Delta^t_x \mid}{\sqrt{{\Delta^t_x}^2 + {\Delta^t_y}^2}} \tag{4}$$

Normalization leads to a situation where the movement vector is formed proportional to the increase/decrease values, but the total length of the vector is kept fixed. This is done to preserve the topology of the space in which learning takes place.

Step length is 400 units. Noise of around 7% of the movement amplitude was added to each step to imitate imperfections in the control of a real robot.

Movement strategies with smoothing were employed. All new steps were made as a combination of the currently calculated steps $\Delta^t_{current} = (\Delta^t_{x,current}, \Delta^t_{y,current})$ combined with the previous steps $\Delta^t_{previous} = (\Delta^t_{x,previous}, \Delta^t_{y,previous})$, using:

$$\Delta^t_{final} = c\Delta^t_{current} + (1-c)\Delta^t_{previous} \tag{5}$$

where $c = 0.9$ was used. Smoothing helps to avoid jerky movements in the beginning of learning and also influences the process of learning, as in Q learning with function approximation, which we are using here, the convergence pattern is dependent on the paths performed during learning.

Finally, if a newly calculate position falls outwith the constraint boundaries, then random steps with uniform distribution are tried out until a point satisfying the constraints is obtained. A virtual step of the learner Δ^l_{final} is obtained in the same way.

2.2 Learning method

Learning is changing only the learner's Q values (more specifically, by changing the weight set $\theta^{k,l}_{\{x,y\}}$, $k = 1,...,N$). Two stages of learning are implemented: first coached learning (20 trials), then coach-free Q learning with function approximation. Three different varieties of coaching-based learning were implemented for the first stage, and some specific approach for Q

- learning with function approximation for the second stage. Let us introduce the Q-learning with function approximation first, and we will comment on how we amend it for including the coaching signal next.

In the second stage (regular Q learning), from four possible actions only two learn during a single step. These are the directions along which the actual movement has been taken. For example, if x has been increased, the unit which has driven the increase learns while the antagonistic 'decrease unit' will not be changed. Furthermore, learning will affect connections of all currently active kernels. Again we will demonstrate the learning rule for the component representing direction x.

Weight update follows an algorithm similar to conventional Q-learning. For each direction, two actions $a = i$ and $a = d$ are possible: increase or decrease. Let us say the current state is described by coordinates $s = (x, y)_{current}$ and some action a (increase or decrease) is chosen that leads to a state $s' = (x, y)_{next}$. In our learning framework, the change in the value $\theta_x^{k,l}(\{i, d\})$ of associated with that state-action pair follows the mean across all activated kernels of $(s', \{i, d\})$:

$$\theta_x^{k,l}(\{i, d\}) = \theta_x^{k,l}(\{i, d\}) + \mu[r + \gamma Q_x^l(s', \{i, d\}) - \theta_x^{k,l}(\{i, d\})]\Phi^k(s) \tag{6}$$

where k is the number of the kernel to which the weight is associated, r is the reward, $\mu < 1$ the learning rate, $\gamma < 1$ the discount factor, $\Phi^k(s)$ is the activity function for kernel k in state s, ${Q^l}_x(s', \{i, d\})$ is the Q value of the learner system in the next state and for action i or d is defined as:

$${Q^l}_x(s', \{i, d\}) = \sum_{k=1}^{N} \theta_x^{k,l}(\{i, d\})\Phi^k(s') / \sum_{k=1}^{N} \Phi^k(s') \tag{7}$$

Equivalent expressions are used for updating values Q_y^l. The teaches values Q_x^t and Q_y^t are kept constant during the whole experiment.

For the coaching phase equation (6) was modified to include punishment instead of the reward:

$$\theta_x^{k,l}(\{i, d\}) = \theta_x^{k,l}(\{i, d\}) + \mu[-p + \gamma Q_x^l(s', \{i, d\}) - \theta_x^{k,l}(\{i, d\})]\Phi^k(s) \tag{8}$$

Punishment p was calculated subtracting Δ_{final}^l from the Δ_{final}^t. Either the Euclidian norm of the expression was used as punishment p, or $p_{x,y}$ was introduced for each direction separately, applying modulus operation for the difference on each direction. The third variety was to use equation 8 without the recurrence included in Q-learning leading to:

$$\theta_x^{k,l}(\{i, d\}) = \theta_x^{k,l}(\{i, d\}) + \mu[-p - \theta_x^{k,l}(\{i, d\})]\Phi^k(s). \tag{9}$$

In this case the scheme was turning into traditional supervised learning, where an error signal was provided for a learner in each step, but the result was not propagating through states.

Throughout the coaching phase reward on reaching the goal was not provided, as it would have been given most often together with a wrong action of the learning agent, not actually leading to the reward (as the agent is lead to the reward by a teacher, but not reaching it through his own actions).

When switching to a second phase, where coaching was stopped, negative weights that have been developed in that phase were changed to positive through reverting the respective i and d weight components for each kernel, and changing the signs of the weights.

3 Results

In Fig. 3 the vector field of a teacher is shown. The vector field has a strong component from the start to the goal, and coaching most of the time happens straight to the goal, and only on rare occasions the trajectories are slightly curved due to differences produced because of

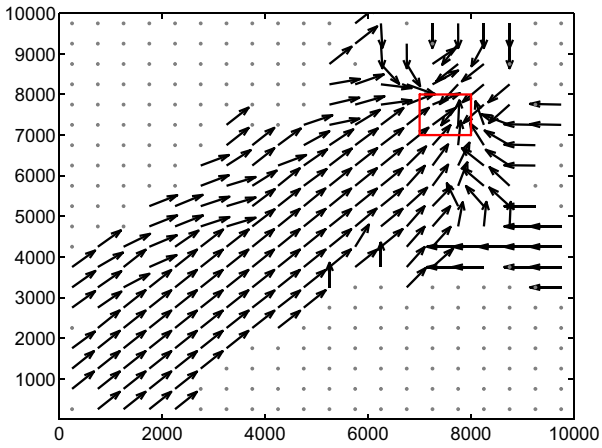

Figure 3: Vector field of a teacher.

random Gaussian activation of the kernels. The dots denote places where Q values have not developed (due to limited exploration in the learning phase) and information which direction to take is unavailable in those regions. Note, even with quite a strong exploration component with the given start-goal configuration, the agent would be getting to those regions with very small probability, and that should also be the situation with any real agent where full knowledge of environment would almost never be available.

In Fig. 4 examples of results for three coaching regimes are provided. In the top row (A, B, C) vector fields for the three regimes after 20 coaching trials are shown. In the second row (D, E, F) time to goal development from 20 trials after coaching has been switched off is shown. In the last row (G, H, I) vector field after those 20 coach-free trials is shown. In the first column results are shown for the case when the punishment is obtained as Euclidian distance between teacher's and learner's movement. In the second column results are shown for the case when the punishment is applied direction-wise. In the third column, recurrence which is a specific attribute of reinforcement-learning and allows propagation of information about rewards or punishment through states, is removed, leading rather to a more traditional supervised learning scheme.

One can be observe that after coaching with Euclidian distance based punishment the vector field from start to goal is vague. It has quite clear direction towards goal near the starting point. Later on the path, the field turns away from the goal, and even opposing vectors are observed (A). Irrespective of these bad properties, convergence after removal of the coach was quick, within 10 trails (D). The vector field obtained in direction-wise punishment regime was much smoother from start to goal (B), but some ripples at the sides of the path were observed. Those ripples were enough to push the learner of the path, after coaching regime has been removed, and multiple unsuccessful attempts were made by the learner in the beginning of the series of independent trials (E). (If number of steps was reaching 300, the trial was terminated and the agent was returned to the starting position, and nine out of ten runs are the ones where return to start was performed) Although, after several successful trials good trajectory was grasped very rapidly and then convergence was quick. An example shows that even with apparently successful coaching the learner may fall onto the bad path, but quick convergence afterwards shows positive influence of the weight values obtained through coaching. Note, however, as this is just a preliminary study so far, we cannot make a statement how representative this example is. Rather is shows that small ripples in the vector field may lead to initially divergent paths. Given the general smoothness of the coached field (B), one should hope that this case is indeed

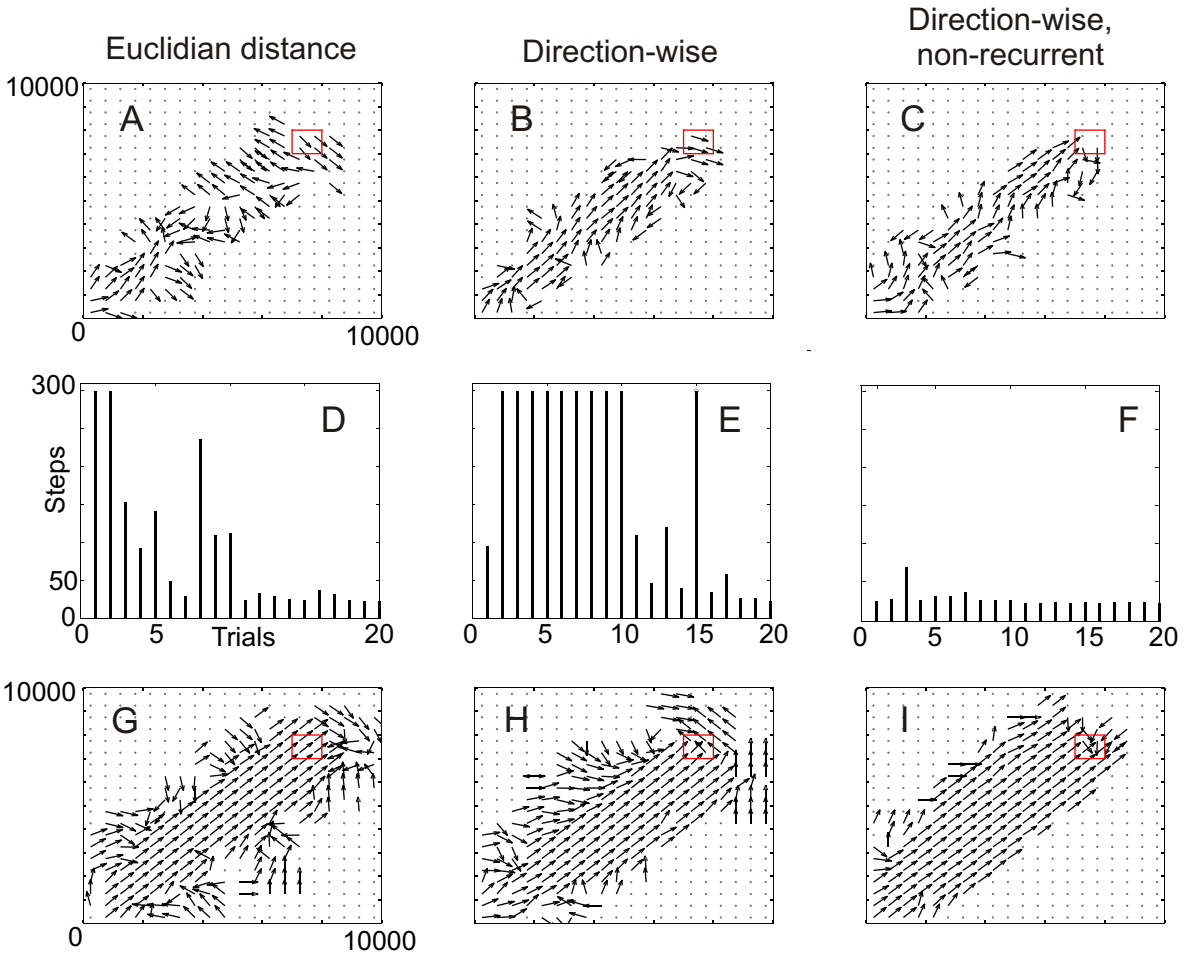

Figure 4: Coaching results. Shown are in the first column the case with Euclidian distance based punishment, in the second column the case with direction-wise punishment, and in the third column the case where recurrence was not applied for Q-value propagation. In the first row vector fields of three different cases after coaching session (20 trials) are shown, in the second row the number of steps to goal, after coaching has been released are shown, and in the last row vector fields after 20 coach-free trials are shown.

not representative.

The vector field obtained without recurrence in learning was the smoothest, although ripples on the sides were also present (C). This time trials immediately after coaching were very successful, and convergence happened within two trials (F).

In all cases after 20 runs without a coach (G, H, I) vector fields have improved, as compared to the ones obtained through coaching (compare to A, B, C, respectively). All resulting fields were of approximately equal quality, independent of the coaching method employed.

4 Final Remarks

All these results are very preliminary and are not yet supported by statistics one which we are currently working. Hence the following conclusion may be unwarranted. Still, it makes sense to wrap things up as they stand at the moment: As a conclusion is seems that recurrence in the coaching phase forces the vector field to ripple, sometimes strongly, like in (A). Although even a

field that does not cover the full path from start to goal may help providing quick convergence, like in (D).

Even quite consistent vector field from start to goal (like in B) may lead to a delay in convergence. This happens because of side ripples. Ripples at the side of the vector fields apparently come out because of worse sampling properties at the side of the path, as compared to the middle which is most often traveled through coaching.

As results are not equivocal (looking good at observation original fields may lead to delayed convergence, and bad initial fields may lead to quick convergence), one has to perform more trials and do statistical analysis of the obtained results, to make final conclusions. Although, initial observations would vote for not using recurrence through coaching.

On the side of theory, one needs now to compare to what degree the last (recurrence free method) might be related to δ-rule based learning.