AUV Pipeline Following using Reinforcement Learning

Sigurd A. Fjerdingen[1], Erik Kyrkjebø and Aksel A. Transeth
SINTEF ICT, N-7465 Trondheim, Norway
[1]sigurd.fjerdingen@sintef.no

Abstract

This paper analyzes the application of several reinforcement learning techniques for continuous state and action spaces to pipeline following for an autonomous underwater vehicle (AUV). Continuous space SARSA is compared to the actor-critic CACLA algorithm [19], and is also extended into a supervised reinforcement learning architecture. A novel exploration method using the skew-normal stochastic distribution is proposed, and evidence towards advantages in the case of tabula rasa exploration is presented. Results are validated on a realistic simulator of the AUV, and confirm the applicability of reinforcement learning to optimize pipeline following behavior.

1 Introduction

Subsea oil&gas pipeline inspection is a costly and time-consuming operation traditionally carried out by trained operators as a manually controlled operation using remotely operated underwater vehicles (ROVs). The ROV is tethered to a surface vessel, which makes the operation highly dependent on surface weather conditions. An untethered autonomous underwater vehicle (AUV) only requires a short launch window, which may greatly reduce the costs and man-hours required for inspection since it may operate without the constant presence of a costly surface vessel. However, autonomous pipeline inspection using a robotic vehicle requires algorithms able to follow the pipeline efficiently and robustly in the presence of disturbances and changing or unknown conditions.

The problem of pipeline localization by sensors such as a camera system, sonar, and echo sounder have been adressed separately for the AUV in question, and references and results may be found in a paper by Breivik et al [5]. AUV control has also been studied extensively previously, examples include using model-based [9, 23] and model-free (learning) [6] techniques. In this paper we concentrate on obtaining a mapping from the detected pipeline by the sensors to an efficient set of waypoints for the AUV – low level control of the AUV is handled using traditional controllers as described in section 3.

Reinforcement learning (RL) is a very active research field, and has been successfully applied to a number of robotic applications. It can be used to make a robot learn how to accomplish or optimize a task *while* interacting with its environment. The robot will receive rewards or punishments based on its choice of actions, and thus over time learns to optimize its actions in relation to the received rewards.

This paper focuses on how to program a robust high level controller for maneuvering an AUV efficiently in relation to a pipeline. The correct and complete set of control parameters for an AUV controller, and their corresponding values, may be hard to determine. Reinforcement learning strategies promise to alleviate such difficulties by exchanging pre-programming by a robot programmer with on-line experimentation by the AUV itself. Furthermore, RL algorithms are able to account for situations unforseen at the time of programming. Instead of spending time on pre-programming the perfect controller able to cope with any forseen or unforseen situations, an alluring alternative is to let RL algorithms figure the difficulties out for themselves by experimenting with pipeline following in the real environment.

The paper is organized as follows. In section 2 a short introduction to reinforcement learning is given. Moreover, a particular type of RL and an extension using the CACLA algorithm is discussed in section 2.2, together with a novel exploration strategy in section 2.3. An overview of the used dynamic simulation environment is given in section 3. Section 4 describes the setup of the simulations conducted, and the results are presented in section 5. Section 6 and 7 gives a discussion of the results and a conclusion, respectively.

2 Reinforcement Learning

Reinforcement learning (RL) deals with the problem of learning when to do what, i.e. how to map situations to actions, in order to maximise a reward [14]. An agent (e.g. an AUV) interacts with a stochastic process modelled as a Markov decision process (MDP), and can observe the current state and immediate reward. The objective is to discover which actions yield the most reward in each situation by experimenting, and may be viewed as a form of associative learning.

An MDP can be defined as a tuple $(\mathcal{S}, \mathcal{A}, R, T)$, where $\mathcal{S}$ is the set of all states, $\mathcal{A}$ the set of all actions, R the re-

ward function, and $T(s, a, s') \in [0, 1]$ the transition function ($s \in \mathcal{S}$ defines the current state, $a \in \mathcal{A}$ the current action and $s' \in \mathcal{S}$ the resulting state). In reinforcement learning problems, the reward function and the transition function are unknown to the agent, and thus ordinary dynamic programming approaches do not apply (see [14] for details).

A value function in reinforcement learning may be defined as

$$V(s) = \mathrm{E}\left\{\sum_{i=0}^{\infty} \gamma_d^i r_{t+i+1} | \pi, s_t = s\right\}. \quad (1)$$

This function describes the cumulative future discounted reward an agent expects to receive using its current policy π from state $s = s_t$, where r is the received reward and $\gamma_d \in (0, 1]$ the discount factor. A corresponding action-value function describes discounted reward when performing action a in state s as

$$Q(s, a) = \mathrm{E}\left\{\sum_{i=0}^{\infty} \gamma_d^i r_{t+i+1} | \pi, s_t = s, a_t = a\right\}. \quad (2)$$

This formulation has been the basis for many RL-algorithms focused on control (e.g. Q-learning and SARSA).

SARSA is a well-known on-policy temporal difference-based reinforcement learning algorithm for control problems – the action-value $Q(s, a)$ is estimated for the current policy π. The algorithm is detailed by Sutton and Barto [14], and is given by the equation

$$Q(s, a) = Q(s, a) + \alpha \left(r + \gamma_d Q(s', a') - Q(s, a)\right), \quad (3)$$

where (s, a) is the current state-action pair, (s', a') the next, r the numerical reward signal received when going from s to s' using action a, and $\alpha \in (0, 1]$ is the learning rate. The temporal difference (TD) error for value functions is given as

$$\delta = r + \gamma_d V(s') - V(s), \quad (4)$$

and corresponding (s, a) for action-value functions. SARSA and other TD-methods converge to an optimal policy for discrete and finite states and actions under the assumptions that all state-action pairs are visited an infinite number of times and that the policy converges to a greedy policy (a policy that always chooses the highest valued action of $Q(s, a)$ for a given s).

2.1 Continuous States and Actions

The case of continuous state spaces in reinforcement learning has been extensively studied [3, 13, 14, 15, 16, 18]. The use of function approximators in some form has emerged as the method of choice for representing the state space. This also allows for generalizing experience. Commonly used function approximators include connectionist structures such as artificial neural networks (ANNs), radial basis function networks (RBFNs) [20], and cerebellar model arithmetic computers or tile coding (CMACs) [13, 14].

Two differing approaches of handling continuous actions are common in reinforcement learning literature. The most intuitive approach in relation to the previous description of reinforcement learning may be to use a numerical optimiziation method on the estimated Q-value (e.g. Newton-Raphson or wire-fitting). Santamaría et al [13] use what they call a one-step search to find $\max_a Q$. The approach consists of a discretization $(a_1, \ldots, a_n)$ over Q and selection of the maximum a_i. Numerical optimization, however, may require a lot of evaluations of the objective function, so if computing the objective function has a high cost this procedure quickly becomes unmanageable for on-line applications.

The other mentioned approach involves actor-critic methods, which separate the estimation of the value function (critic) from the estimation of the policy (actor). The approach has been around for quite some time (see e.g. the 1977 article by Witten [22]), but has not gathered a lot of interest until later years when problems using the approach of direct determination of the policy from the action-value estimate became apparent [15].

CACLA. Van Hasselt and Wiering [19] present an actor-critic based reinforcement learning algorithm named CACLA (Continuous Actor-Critic Learning Automaton) for learning in continuous action spaces. The value function (critic) is updated using the TD-error from (4) as

$$V(s) = V(s) + \alpha_V \delta, \quad (5)$$

where $\alpha_V \in (0, 1]$ is the learning rate for the value function. For continuous state spaces, the value function may be represented by a function approximator $\hat{V}_\theta(s)$ parameterized by a vector θ. Using gradient-descent on the mean squared error between the experienced and currently estimated value function gives an update rule for the parameters as (see Sutton and Barto [14] for details)

$$\theta = \theta + \alpha_V \delta \nabla_\theta \hat{V}_\theta(s). \quad (6)$$

In actor-critic algorithms, a stochastic policy $\Pr(a|s) = \pi(a|s; \phi)$, parameterized by a function approximator with parameter vector ϕ, is usually employed. The policy parameters for CACLA are updated by

$$\phi = \phi + \alpha_\pi \max(\mathrm{sgn}(\delta), 0)(a - A_\phi^c(s)) \nabla_\phi A_\phi^c(s), \quad (7)$$

where $\alpha_\pi \in (0, 1]$ is the actor learning rate and

$$\pi(a, s; \mu, \sigma) = \frac{1}{\sqrt{2\pi}\sigma} e^{-\frac{(a - A_\phi^c(s))^2}{2\sigma^2}} \quad (8)$$

is a stochastic actor policy employing the Gaussian distribution with mean $\mu(s) = A_\phi^c(s)$ approximated by a function approximator. The parameter σ is used to control the amount of exploration for the policy. For a general actor-critic algorithm, the policy parameters are updates as (e.g. [4])

$$\phi = \phi + \alpha_\pi \delta \nabla_\phi \ln \pi(a, s; \phi). \quad (9)$$

The modifications of (7) in relation to (9) is based on the following intuitions (see [19] for details):

1. The $\max$-term insures that the estimate of the best actor value is not updated when the TD-error is negative. This is reasonable since we do not want to adjust the policy in the opposite direction of some perceived negative action as this does not necessarily equal better solutions.

2. The signum-term (sgn) makes updates of the actor invariant to scaling issues when relating the TD-error to the actor policy. Van Hasselt and Wiering note CACLA as superior to some comparable actor-critic algorithms when experimenting with varying the scaling of the reward function.

The strong theoretical underpinnings presented by several authors [4, 7, 15] for actor-critic algorithms, together with the intuitions given by Van Hasselt and Wiering in addition to the good performance of the algorithm when tested on the cart pole-problem, constitute the reasoning behind considering this particular method for the application domain of autonomous underwater vehicles.

2.2 Supervised Reinforcement Learning

The main difference between supervised learning and reinforcement learning is the availability of a trainer with knowledge of correct input/output sequences for a supervised learning problem. In reinforcement learning, the learner has to discover these by trial and error via the external reinforcement signal.
Rosenstein and Barto [11] combine a form of supervised learning with an actor-critic reinforcement learning architecture in order to implement previous knowledge in a reinforcement learner. The algorithm uses a supervisor in the form of a previously known controller. Actions from this controller are combined with the actions from the RL controller through a weighted sum

$$a = ka^{\text{RL}} + (1-k)a^{\text{SUP}}, \tag{10}$$

where the parameter $k \in [0,1]$ weights the influence of the supervisor action a^{SUP} versus the RL controller action a^{RL}. The actor approximator weights are now updated as

$$\begin{aligned} \phi &= \phi + \alpha_\pi \Big[k\delta(a - A^c_\phi(s)) \\ &+ (1-k)(a^{\text{SUP}} - A^c_\phi(s)) \Big] \nabla_\phi A^c_\phi(s). \end{aligned} \tag{11}$$

The first part of (11) is identical to (9), while the second part is the gradient from a quadratic supervisory error. When $k \to 1$ the update rule behaves like a standard actor-critic algorithm, while $k \to 0$ turns the update rule into adapting the weights to fit the supervisory controller.
In this paper we propose to extend the CACLA algorithm into the supervised reinforcement learning architecture of Rosenstein and Barto by modifying (11) to suit the update rule of (7) in the following manner

$$\begin{aligned} \phi &= \phi + \alpha_\pi \Big[k \max(\text{sgn}(\delta), 0)(a - A^c_\phi(s)) \\ &+ (1-k)(a^{\text{SUP}} - A^c_\phi(s)) \Big] \nabla_\phi A^c_\phi(s) \end{aligned} \tag{12}$$

The parameter k weighting the influence of the supervisor versus the learner may vary with state; $k(s)$. Rosenstein and Barto use a function approximator to keep track of the state-dependent k. The underlying intuition is that k may be used as a measure of confidence for each state in the state-space: States that have been visited more often can yield more trust to the learner since the value function and policy should already have adopted the supervisory controller's action as a base estimate. $k(s)$ thus starts at 0 for all $s \in \mathcal{S}$, and is updated using

$$k(s) = k(s) + \Gamma_k(\Delta_k) \tag{13}$$

in our algorithms, where Γ_k caps updates when k reaches $k_{\text{max}} = 1$. Rosenstein and Barto also implement a forgetting mechanism into the k updates, such that states which have not been visited in a long time will be less trusted.

2.3 Exploration

A reinforcement learning agent needs to explore its environment in order to discover more optimal solutions. Methods for exploring range from simple to the more elaborate. Gibbs softmax, ϵ-greedy and Gaussian exploration are commonly used exploration strategies [11, 13, 14, 19]. Other methods attempt to build a model of the environment (directed exploration) [17, 21] or augment the reward function [8]. Rückstieß et al [12] use a hybrid method of random and directed exploration where actions are offseted the same state-dependent amount in an episode, and the amount is randomized using a Gaussian distribution between each episode. Supervised reinforcement learning may also be viewed as a way to guide exploration, either in the form of gradual guidance from a teacher [11] or e.g. in the context of apprenticeship learning [1].

This paper proposes to direct the exploration resources in an action direction that looks more promising with regards to receiving positive temporal difference-errors in a state-dependent manner. Looking at one of the fundamental underpinnings of reinforcement learning theory, Thorndike's Law of Effects states (amongst other things) that *responses accompanied by discomfort to the animal will* [...] *have their connections with that situation weakened* (see e.g. Sutton and Barto [14] for an introduction). The CACLA approach breaks from this behavior when electing to avoid updating the actor policy for negative TD-errors (see (7)). At the same time, the intuition of not updating the estimated optimal action towards some value with unknown utility holds some merit. This proposition is an attempt at using the information inherent in negative TD-errors while keeping the current estimate of the optimal action constant. This is achieved by using the skew normal distribution instead of the Gaussian distribution (see (8)) as basis for the stochastic policy. The skew normal distribution was introduced by Azzalini [2], and its probability density function

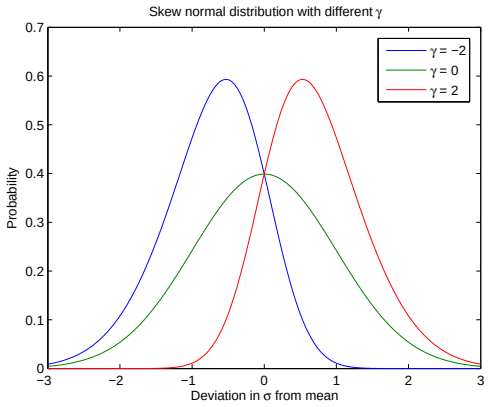

Figure 1: Influence on the skew normal distribution for different values of γ.

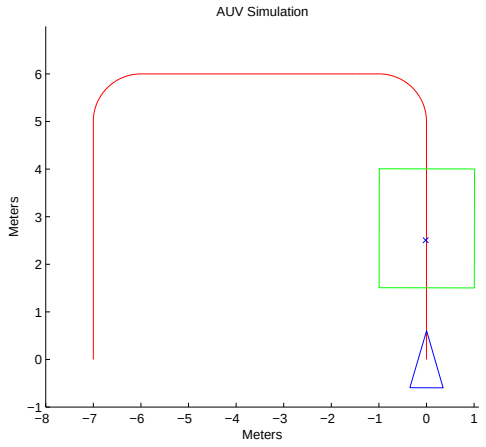

Figure 2: Red line illustrates the pipe centerline, the blue triangle is the AUV, and the green rectangle is the visible area for the camera system.

is given by

$$f(x) = \frac{1}{\sigma\pi} e^{-\frac{(x-\mu)^2}{2\sigma^2}} \int_{-\infty}^{\gamma\left(\frac{x-\mu}{\sigma}\right)} e^{-\frac{t^2}{2}}\, \mathrm{d}t, \qquad (14)$$

where the mean μ and standard deviation σ equal their Gaussian distribution counterparts, while the parameter γ directs the *skewness* of the distribution. $\gamma = 0$ gives the Gaussian distribution. Skewness refers to an asymmetric perturbation of the density center and tail of the Gaussian distribution in such a way as to give higher probabilities for drawing either from the right or left side of μ. The effects of varying γ are illustrated in Fig. 1.

The γ parameter of the skew normal distribution may in other words be used to guide exploration away from action values that have received negative TD-errors while keeping the current estimate μ of the optimal action fixed. In addition to updating the weight parameters for the mean, an update has to be done for the γ parameter in order to direct the exploration. In order to handle this, we propose the update equation

$$\phi_\gamma = \phi_\gamma + \Gamma\left(\mathrm{sgn}(\delta)\mathrm{sgn}(a - A_\phi^c(s))\Delta_\gamma\right), \qquad (15)$$

where Γ is a function which caps updates when $|\gamma_{\max}|$ is reached, and Δ_γ is the γ increment (or decrement). Note here the underlying assumption that also the γ parameter is estimated using a function approximator similar to the estimation of μ.

3 AUV Model and Implementation

The AUV used for pipeline following is based on a small, low cost vessel developed for experimental validation of underwater vehicle control systems at NTNU/SINTEF. The AUV is equipped with two vertical and two horizontal tunnel thrusters, two aft propellers and two diving rudders. The 3 degree-of-freedom simulation model of the AUV in the body frame is given as

$$\mathbf{M}\dot{\nu} + \mathbf{C}(\nu)\nu + \mathbf{D}(\nu)\nu = \tau_\nu, \qquad (16)$$

and is a function of the body fixed velocities $\nu = [u, v, r]^T$. The inertia matrix $\mathbf{M}$, Coriolis and centrifugal matrix $\mathbf{C}(\nu)$, and the nonlinear damping matrix $\mathbf{D}(\nu) = \mathbf{D} + \mathbf{D}_n(\nu)$ are defined as

$$\mathbf{M} = \begin{bmatrix} 80 + 0.026\rho & 0 & 0 \\ 0 & 80 + 0.04\rho & 0.0135\rho \\ 0 & 0.0135\rho & 10 + 0.0107\rho \end{bmatrix}$$

$$\mathbf{C}(\nu) = \begin{bmatrix} 0 & 0 & (0.04\rho - 80)\,v + 0.0135\rho r \\ 0 & 0 & (80 - 0.026\rho)u \\ (80 - 0.04\rho)\,v - 0.0135\rho r & (0.026\rho - 80)u & 0 \end{bmatrix}$$

$$\mathbf{D} = \begin{bmatrix} 0.72 & 0 & 0 \\ 0 & 0.8896 & 7.25 \\ 0 & 0.0313 & 1.90 \end{bmatrix}$$

$$\mathbf{D}_n(\nu) = \begin{bmatrix} 1.33|u| + 5.87u^2 & 0 & 0 \\ 0 & 36.5|v| + 0.805|r| & 0.845|v| + 3.45|r| \\ 0 & 3.96|v| - 0.130|r| & -0.080|v| + 0.75|r| \end{bmatrix}$$

$\rho = 1.025$ is the density of seawater. The AUV is controlled by a PD waypoint controller in an earth-fixed reference frame with gains $\mathbf{K}_p = [1.5, 0.3, 1]$, $\mathbf{K}_d = [6, 4, 2]$ where waypoints are limited to be placed on a circle within the angular range of $[-90°, 90°]$. The AUV is equipped with an Inertial Measurement Unit, an echo sounder and a stereo camera system detecting the centerline of the pipeline in view. If the pipeline is lost from view, a heuristic circular search algorithm is invoked.

4 Simulation Setup

This section describes the simulation setup used to train and validate the previously described reinforcement learning algorithms for the AUV and pipeline environment. Two main experiments have been conducted. The first is carried out in order to analyse the performance of previously discussed RL algorithms for our application of interest, and is described in section 4.1. The second experiment, described in section 4.2, validates if the learned policy is able to generalize to other pipeline geometries.

Function Approximation. In the experiments presented in this paper, RBFNs are used as function approximators both for critic and actor approximation. RBF networks typically have three layers; an input layer, a hidden layer with a Gaussian activation function, and a linear output layer. The output of the network is

$$f(\mathbf{x}) = \sum_{i=1}^{N} \phi_i \rho(||\mathbf{x} - \mathbf{c}_i||), \tag{17}$$

where

$$\rho(||\mathbf{x} - \mathbf{c}_i||) = e^{-\frac{1}{\sigma}||\mathbf{x}-\mathbf{c}_i||^2} = \prod_j e^{-\frac{1}{\sigma_j}(x_j - c_{ij})^2} \tag{18}$$

is the Gaussian activation function with center vector $\mathbf{c}$ and width σ, ϕ the associated weight parameter, and N the number of activation functions.
The state space is constructed of four dimensions; x and y position of one endpoint of the current pipeline in view, x position of the other end of the detected pipeline, and the longitudinal speed v_x of the AUV. Fig. 3 illustrates the points of the pipeline available as state space dimensions. The reasoning behind the state space division has been to minimize the number of state space dimensions in order to keep calculations at a minimum, while keeping all coordinates local with respect to the AUV in the sense that all information is available without external reference systems. Both critic and actor use the exact same state space dimensions, and the basis function parameters are also equal. The basis function parameters are the center vector $\mathbf{c}$ and the width vector σ. The center vectors for the state space dimensions are as follows

$$\begin{aligned} x_1 &\rightarrow [-1, -0.8, -0.4, 0, 0.4, 0.8, 1][\text{m}] \\ y_1 &\rightarrow [0, 0.375, 1.125, 1.875, 2.625, 3][\text{m}] \\ x_2 &\rightarrow [-1, -0.8, -0.4, 0, 0.4, 0.8, 1][\text{m}] \\ v_x &\rightarrow [-0.25, -0.156, 0.031, 0.219, 0.406, 0.5][\text{m/s}], \end{aligned}$$

and the width is calculated as half the largest distance between centers in order to cover the entire state space completely in each dimension. See e.g. Park and Sandberg [10] for more details on RBFNs. The total number of basis functions thus sums to 1'764. Reducing or optimizing the choice of basis functions has not been tried.

The learning rate has been set to $\alpha_V = 0.4$ for the value function updates and $\alpha_\pi = 0.2$ for the actor update, and are held constant throughout the learning period. The discount factor is set to $\gamma_d = 0.8$. The exploration factor starts at $\sigma = 1$ and decreases linearly per episode towards $\sigma = 0.01$ for the final episode.
The reward function is calculated by the environment (unknown to the learner) as

$$R = v_x - e^2_{\text{dist}} - e^2_{\text{angl}} - (a - a_{\text{prev}})^2. \tag{19}$$

That is, the AUV receives reward for higher speed v_x and is punished if the pipeline either deviates from the middle of the camera frame (e_{dist}), the angle of the pipeline deviates from a vertical line in the camera fram (e_{angl}), or if chosen actions vary greatly between consecutive decisions $(a - a_{\text{prev}})$. Qualitatively, this should provide the AUV with a goal of keeping a smooth trajectory with the pipeline as center and level as possible in view. If the AUV ever looses track of the pipeline, a pre-coded safety behavior overrides the learning behavior, and a punishment of -10 per control step – a larger punishment than possible if the pipeline is in view – is incurred. The safety behavior is a rotating motion around the AUV center axis, making the AUV camera system sweep the entire surrounding of the AUV in hope of returning track of the pipeline.

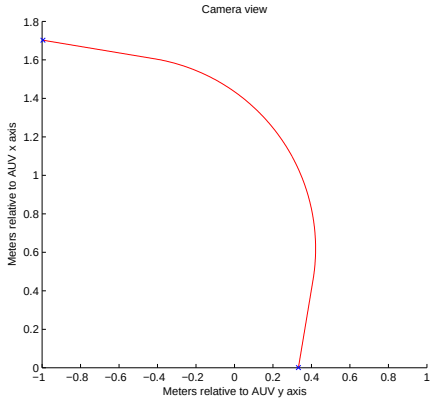

Figure 3: Red line is visible pipeline segment. Blue crosses are data points extracted by the camera system available to the learner.

4.1 RL analysis

In this experiment, the pipeline is laid out as three straight segments with $90°$ left turns between. The simulation environment may be visualized as in Fig. 2. The goal of the experiment is to analyse the application of (a) the standard SARSA algorithm, (b) the CACLA algorithm, (c) CACLA with skew normal exploration, and (d) supervised CACLA. Each episode lasts for 20 seconds, and each trial repeats 500 episodes. The learning controller runs once each second, giving a total of 20 decisions per episode. Each of the experiments were repeated 5 times.

SARSA. For the SARSA algorithm, the central equation is (3). An RBFN is used to estimate Q, employing the state space previously described. The action dimension has been added to the state space, with centers $[-90°, -81°, -63°, -45°, \ldots, 63°, 81°, 90°]$. For action selection, the Q-function has been discretized with step size $9°$ in the range $[-90°, 90°]$. The state space explosion of Q-function based methods is easily visible here, as the total number of basis functions now sums to 22'932.

CACLA. For skew normal exploration, $\Delta_\gamma = (11.25°, 22.5°, 45°)$ has been tested and Γ caps updates of ϕ_α at $\alpha_{\max} = 1$ in all cases. In the case of supervision, the heuristic controller providing a^{SUP} is a controller that sets the waypoint in the exact direction of the pipe endpoint furthest away from the AUV. This simple control scheme generates a path that is able to track the $90°$ turns, but overshoots somewhat when tracking. The α and k parameters are updated in a state-dependent manner by using an identical RBFN as for the state space. The k parameter is initialized at 0 for all states, meaning that the supervisory controller has full control over the action. Updates are given with $\Delta_k = 0.05$. Our algorithms do not implement a forgetting factor.

4.2 Validation

This experiment aims to validate if the AUV is able to track a different, unknown pipe geometry. A suitable algorithm was selected based on the analysis experiment – the supervised CACLA algorithm with $\gamma = 22.5°$ skew (equals absolute value of 0.5 in relation to Fig. 1), see section 5 for results. Since the AUV learning algorithm had only been trained in left turns, the pipeline was first mirrored along the y-axis in Fig. 2 to get right turns. A learning phase with identical parameters as in section 4.1 was conducted. The AUV learning algorithm was validated on the pipe geometry shown in Fig. 4. During this experiment, the optimal action was always chosen deterministically. For comparison, the unknown pipe geometry was also tested using the heuristic controller described in section 4.1.

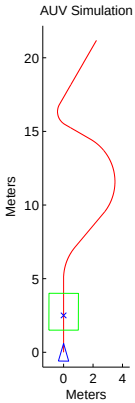

Figure 4: Validation environment for learner.

5 Results

5.1 RL analysis

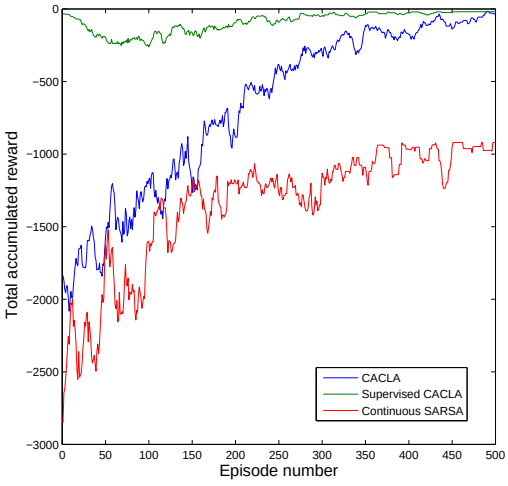

Figure 5: Continuous SARSA versus CACLA and supervised CACLA.

Fig. 5 shows a comparison of CACLA, supervised CACLA and SARSA. The Sarsa algorithm did not converge to a result able to track the pipeline within the alloted number of episodes in any instance. Both versions of CACLA converged to a successful policy in all instances.

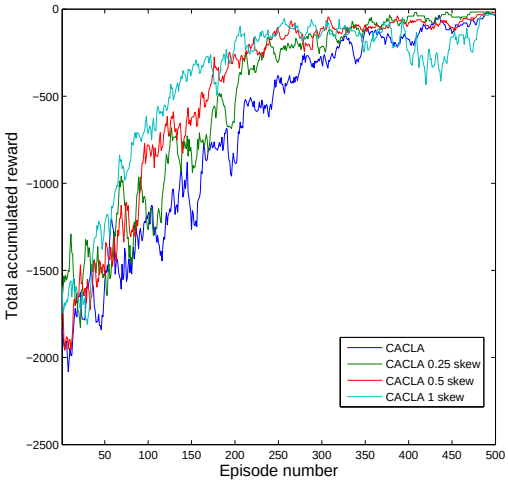

Figure 6: Tabula rasa CACLA with varying skew (α).

Skew updates of $\Delta_\gamma = 11.25°$, $22.5°$ and $45°$ has been tested for the tabula rasa CACLA algorithm. Results are shown in Fig. 6. All cases converged to a successful policy in all instances.

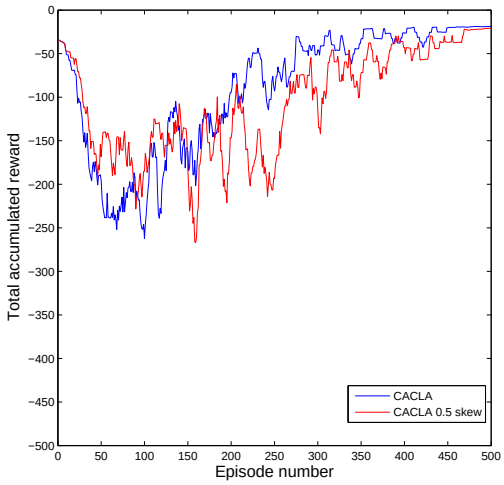

Figure 7: Supervised CACLA with varying skew (α).

Supervised CACLA is compared to supervised CACLA with a skew update of $\Delta_\gamma = 22.5°$ in Fig. 7. Both cases converged to a successful policy in all instances. All figures are an average of 5 runs of 500 episodes, and a sliding average of 10 episodes has been used to smooth the figures.

5.2 Validation

Training of the supervised CACLA algorithm resulted in a successful policy also for right turns. Table 1 shows the results of the RL-algorithm compared to the supervisory heuristic controller previously described.

Algorithm	Acc. reward
CACLA	-23.2349
Heuristic	-27.8590

Table 1: Accumulated reward for supervised CACLA and heuristic controller. More positive reward is better. Accumulated reward per episode converges to a constant as $t \to \infty$ when exploration $\sigma \to 0$.

6 Discussion

The CACLA algorithm clearly outperforms the traditional SARSA algorithm for our application. Adding skew exploration also seems to have a positive effect on early stages of the tabula rasa CACLA algorithm – the stages with more heavy exploration. Comparing the supervised CACLA algorithm with and without skew, we see no advantage of using the skew exploration parameter. A possible explanation for this is that when the solution already is close to an optimum (initialized by the supervisory controller), the skew parameter will just oscillate around the optimal μ because of inaccuracies in the value function estimates. It is believed to be beneficial to exploration with skew to employ a more elaborate update strategy which dampens responses in such cases.

The exploration policy for CACLA and the skew variant does not equate to the gradient of the actor update rule via Sutton et al's [15] *compatible* function approximator. The authors have chosen to keep with the intuition of the original CACLA algorithm in this respect, but a theoretical derivation of the implications should be done.

Since the CACLA algorithm has proven quite robust with respect to parameters such as learning rate and discounting, little effort has gone into tuning these variables – general guidelines as given by Van Hasselt and Wiering [19] have been followed. Better results from the SARSA algorithm might be obtained by tuning these variables. This has not been verified by the authors. The gain scheduling employed by the supervised reinforcement learning algorithm is implemented as a weighted sum. In the general case, this may lead to choosing a worse action than any of the alternatives and thus may gain from using a different strategy such as stochastic choice.

7 Conclusions and Future Work

7.1 Conclusions

This paper has analyzed the application of SARSA, CACLA and supervised CACLA for continuous state and action spaces applied to the task of pipeline following for an AUV. Experiments in a simulation environment have shown supervised CACLA to be the best candidate, as well as the ability to generalize the learned pipeline following strategy to new and unknown pipe geometries.

The use of skew normal distribution for exploration has been proposed, and evidence towards its advantages in the tabula rasa case has been presented. No such evidence has been found in the case of supervised reinforcement learning.

The simulation results show that reinforcement learning is well suited to optimize pipeline following behavior for an AUV.

7.2 Future Work

It is important to analyze the application of skew normal exploration more rigorously, for instance by looking at the situation in which the estimated policy is close to the optimal one. It is also necessary to study the theoretical implications of changing the policy from a Gaussian distribution (see (7-9)). Implementation on the real-world AUV is a necessary next step in evaluating the RL-algorithms for AUV pipeline following.

References

[1] Pieter Abbeel and Andrew Y. Ng. Exploration and apprenticeship learning in reinforcement learning. In *Proceedings of the 22nd international conference*

on machine learning, volume 119 of *ACM International Conference Proceeding Series*, pages 1–8. ACM, 2005.

[2] A. Azzalini. A class of distributions which includes the normal ones. *Scandinavian Journal of Statistics*, 12:171–178, 1985.

[3] Leemon Baird. Residual algorithms: Reinforcement learning with function approximation. In *Proceedings of the International Conference on Machine Learning*. Morgan Kaufmann Publishers Inc., 1995.

[4] Shalabh Bhatnagar, Richard S. Sutton, Mohammad Ghavamzadeh, and Mark Lee. Incremental natural actor-critic algorithms. In *Advances in Neural Information Processing Systems 20*, pages 105–112. MIT Press, Cambridge, MA, 2008.

[5] Gøril M. Breivik, Sigurd A. Fjerdingen, and Øystein Skotheim. Robust pipeline localization for an autonomous underwater vehicle using stereo vision and echo sounder data. In *IS&T/SPIE Intelligent Robots and Computer Vision XXVII: Algorithms and Techniques, accepted for publication*, 2010.

[6] C. Gaskett, D. Wettergreen, and A. Zelinsky. Reinforcement learning applied to the control of an autonomous underwater vehicle. In *Proceedings of the Australian conference on robotics and automation*, 1999.

[7] Vijay R. Konda and John N. Tsitsiklis. On actor-critic algorithms. *SIAM Journal on Control and Optimization*, 42(4):1143–1166, 2003.

[8] Maja J. Mataric. Reward functions for accelerated learning. In *Proceedings of the Eleventh International Conference on Machine Learning*, pages 181–189. Morgan Kaufmann Publishers Inc., 1994.

[9] P. K. Paim, B. Jouvenecel, and L. Lapierre. A reactive control approach for pipeline inspection with an auv. In *OCEANS 2005*, volume 1-3, pages 201–206, 2005.

[10] J. Park and J. W. Sandberg. Universal approximation using radial-basis-function networks. *Neural Computation*, 3(2):246–257, 1991.

[11] Michael T. Rosenstein and Andrew G. Barto. Supervised actor-critic reinforcement learning. In *Learning and Approximate Dynamic Programming: Scaling Up to the Real World*, pages 359–380, 2004.

[12] Thomas Rückstieß, Martin Felder, and Jürgen Schmidhuber. *Machine Learning and Knowledge Discovery in Databases*, chapter State-dependent exploration for policy gradient methods, pages 234–249. Springer Berlin / Heidelberg, 2008.

[13] Juan C. Santamaria, Richard S. Sutton, and Ashwin Ram. Experiments with reinforcement learning in problems with continuous state and action spaces. *Adaptive behavior*, 6(2):163–217, 1997.

[14] Richard S. Sutton and Andrew G. Barto. *Reinforcement Learning: An Introduction*. Adaptive Computation and Machine Learning. The MIT Press, London, England, 1998.

[15] Richard S. Sutton, David McAllester, Satinder Singh, and Yishay Mansour. Policy gradient methods for reinforcement learning with function approximation. In S. A. Solla, T. K. Leen, and K. R. Muller, editors, *Advances in neural information processing systems*, volume 12 of *Advances in neural information processing systems*, pages 1057–1063. MIT Press, 2000.

[16] Sebastian Thrun and Anton Schwartz. Issues in using function approximation for reinforcement learning. In *Proceedings of the 1993 Connectionist Models Summer School*, 1993.

[17] Sebastian B. Thrun. Efficient exploration in reinforcement learning. Technical report, Carnegie Mellon University, 1992.

[18] John N. Tsitsiklis and Benjamin van Roy. An analysis of temporal-difference learning with function approximation. *IEEE Transactions on Automatic Control*, 42(5):674–690, 1997.

[19] Hado van Hasselt and Marco A. Wiering. Reinforcement learning in continuous action spaces. In *Proceedings of the 2007 IEEE symposium on approximate dynamic programming and reinforcement learning*, pages 272–279, 2007.

[20] X. S. Wang, Y. H. Cheng, and W. Sun. Q learning based on self-organizing fuzzy radial basis function network. In J. Wang, Z. Yi, J. M. Zurada, B. L. Lu, and H. J. Yin, editors, *Advances in neural networks*, volume 3971 of *Lecture notes in computer science*, pages 607–615. Springer-verlag Berlin, 2006.

[21] Marco Wiering and Jürgen Schmidhuber. Efficient model-based exploration. In *Proceedings of the fifth international conference on simulation of adaptive behavior on From animals to animats 5*, pages 223–228, 1998.

[22] Ian H. Witten. An adaptive optimal controller for discrete-time markov environments. *Information and Control*, 34(4):286–295, 1977.

[23] S. Zhao and J. Yuh. Experimental study on advanced underwater robot control. *IEEE Transactions on Robotics*, 21(4):695–703, 2005.