Deep Reinforcement Learning for Cognitive SONAR Systems

Nico Neumann[1], Bastian Kaulen[1], Sören Christensen[2], Gerhard Schmidt[1]

[1] *Digital Signal Processing and System Theory (DSS), CAU zu Kiel, Kiel, Germany, E-Mail: {nin, bk, gus}@tf.uni-kiel.de*

[2] *Stochastics Research Group, CAU zu Kiel, Kiel, Germany, E-Mail: christensen@math.uni-kiel.de*

Abstract

In recent years it has been shown that reinforcement learning can also be used for very complex tasks with the help of neural networks. Thereby strategies are often found that go beyond results achieved by humans. This allows the use in autonomous tasks in a variety of different areas and applications. One possible application is to control a cognitive SONAR system. The goal is to scan a port area optimally in order to detect and find incoming objects as quickly as possible. Therefore, an environment simulation was designed that represents the port area under consideration of random occurring reflections that result in different observations for each scan as well as the use of different novel SONAR scanning modes. This work presents an implementation and evaluation of an environment for the training of algorithms to enable the autonomous observation of a port area.

SONAR Modes

The goal of the SONAR system containing a neural network that was generated by deep reinforcement learning is fast and reliable detection of objects entering a harbor area. For this purpose, a decision is required between three different SONAR modes which, for simplicity, differ only in the range properties and are briefly presented in the following. These three so far novel and purely theoretical modes are however already possible by simulations.

Near field sector scan

The first mode is the near field sector scan. In this mode adaptive filters are computed by continuous transmission of signals, which adapt to the impulse responses in specific directions. However, in order not to overdrive the receiving hydrophones, the transmitting power must be reduced and this shrinks the maximum distance for possible detections. The chosen boundaries in this mode allows a detection from $y_{\mathrm{nf,min}} = 0\,\mathrm{m}$ to $y_{\mathrm{nf,max}} = 375\,\mathrm{m}$. This is shown in Fig. 1 as a red shaded area.

Intermediate sector scan

The second mode is based on the principle of Multiple-Input Multiple-Output (MIMO) signal processing in which orthogonal signals are transmitted. This enables intermediate sector coverage. The minimum distance is set to $y_{\mathrm{if,min}} = 48\,\mathrm{m}$, because the hydrophones can overdrive during transmission and are therefore assumed to be blind in the time of transmission. The maximum distance is limited by the distance dependent attenuation and is here set to the value $y_{\mathrm{if,max}} = 700\,\mathrm{m}$. This is depicted in Fig. 1 as a blue shaded area.

Far field sector scan

The third mode is also based on the MIMO principle. Here classical coherent pings are sent in certain directions one after another. However, the different pings are constructed using orthogonal signals. Therefore, Single-Input Multiple-Output (SIMO) performance is achieved in the specific directions. Since the receivers could overdrive as well during transmitting, the section increases, in which the SONAR system is blind. This distance depends on the one hand on the transmit signal length and on the other hand on the number of pings. Here the minimum value was set to $y_{\mathrm{ff,min}} = 675\,\mathrm{m}$. The maximum distance of detections is also determined by the distance dependent propagation and was set to $y_{\mathrm{ff,max}} = 1000\,\mathrm{m}$. This is shown in Fig. 1 as a green shaded area.

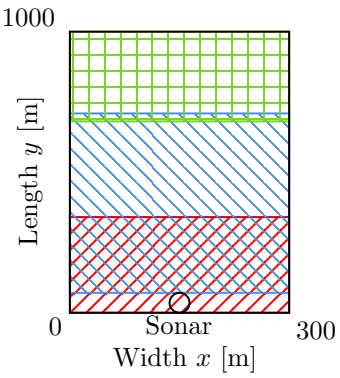

Figure 1: Visualization of the different sector scans. Red represents the near field, blue the intermediate and greed the far field scan.

With these three novel modes, the neural network is forced to find a decision, which significantly influences the performance of the SONAR system and provides a good first indicator if this concept of reinforcement learning is applicable in SONAR applications.

Reinforcement Learning

Reinforcement learning bases on the Markov Decision Process (MDP) which is an extension of Markov chains and is used for modeling decision-making situations. For the finite case a MDP consists of a set of states $\boldsymbol{S}$, actions $\boldsymbol{A}$ and rewards $\boldsymbol{R}$ with a finite number of elements. The agent is the learner and decision maker while the part it interacts with is known as the environment. The agent selects an action and the environment responds to this action with a new situation. The agent also receives a reward from the environment which it tries to maximize over time by the choice of actions. The interaction happens at each sequence of discrete time steps t. The agent is in an environment state $S_t \in \boldsymbol{S}$ at each discrete time step t and picks an action $A_t \in \boldsymbol{A}$ on basis of that state. The agent then receives a reward $R_t \in \boldsymbol{R}$ based on the previous selected action and is in the new state

S_{t+1}. In the first time step $t = 0$ the environment and the agent are in a state $S_t = S_0$ and the agent performs an action $A_t = A_0$ and receives the reward $R_{t+1} = R_1$. Thus the agent and the environment move on to the next state $S_{t+1} = S_{0+1} = S_1$ which represents the state in which the environment is at the time step $t + 1$. In time step $t = 1$ the environment is in state $S_t = S_1$ and the agent takes action $A_t = A_1$ and receives the reward $R_{t+1} = R_2$ and the environment moves on to the next state $S_{t+1} = S_{1+1} = S_2$. This results in a sequence that begins as shown in Eq. 1 where the associated state, action and reward elements for each time step are highlighted [3]:

$$(\underbrace{S_0, A_0, R_1}_{t=0}, \underbrace{S_1, A_1, R_2}_{t=1}, \underbrace{S_2, A_2, R_3}_{t=2}, \dots). \tag{1}$$

A graphical representation of this interaction between the agent and environment is given in Fig. 2 where the action selection of the agent is based on a policy called π.

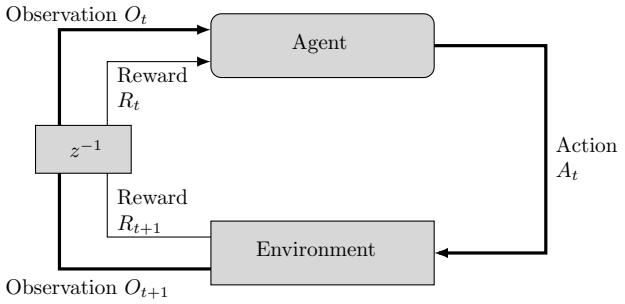

Figure 2: The simplified agent-environment interaction in reinforcement learning.

Q-learning has the ability to learn at every time step and it is an off-policy method which allows for updates with data collected with any policy. The policy to collect data from the environment by interacting with it is known as behavior policy while the (continuously) improving policy is referred to as target policy. The behavior policy must strike a compromise between the exploration of new options that may be better than known ones and the exploitation of options that are already known for the highest reward yet. This is often done with the simple ϵ-greedy algorithm which selects a random action with a probability of ϵ or the most popular action with a probability of $1 - \epsilon$. Thus the target policy is chosen to be greedy and therefore is constantly improving. The update rule is given in Eq. 2 and α ($0 < \alpha \leq 1$) indicates the learning rate while γ ($0 \leq \gamma \leq 1$) is the discount factor. The learning rate α determines how much the state value is changed for each update while the discount factor γ is used to weight future rewards and immediate rewards against each other and defining their importance [3]:

$$Q(S_t, A_t) \leftarrow \underbrace{Q(S_t, A_t)}_{\text{old value}} + \underbrace{\alpha}_{\substack{\text{learning}\\\text{rate}}} \cdot \Big[\underbrace{R_{t+1}}_{\text{reward}} + \underbrace{\gamma}_{\substack{\text{discount}\\\text{factor}}} \cdot \underbrace{\max_a Q(S_{t+1}, a)}_{\substack{\text{estimate of}\\\text{optimal}\\\text{future value}}} - \underbrace{Q(S_t, A_t)}_{\text{old value}} \Big]. \tag{2}$$

Rules of Environment

To define the behavior of the environment a set of rules is needed. The main goal is to detect an incoming object (diver) as soon as possible. The goal point of the diver is a ship that is placed randomly between four selectable values that define an area in which the ship could be placed (Fig. 3). The diver's route is made up from a definable interval (magenta-colored) from which two points $P_{rd1,2}$ are randomly picked and the diver point P_D is randomly selected from a straight line between these two points (Fig. 4) and not known to the agent. For simplification the shortest path between P_D and the nearest ship point is chosen and interpolated to the outermost point as shown in Fig. 4 but the arrival of the diver in the port might be delayed. To prefer the fast finding a bonus reward $B(t)$ was used as shown in Eq. 3:

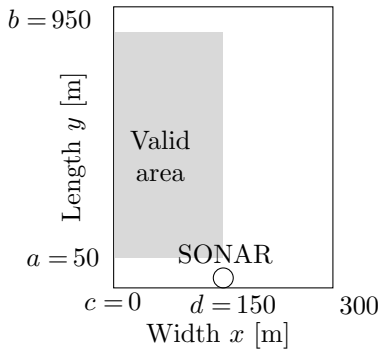

Figure 3: The ship is placed randomly between the given values of $a - d$ while the SONAR system is located at the bottom middle.

$$B(t) = \begin{cases} B_0 - B_{reduce} \cdot t, & \text{if } 0 \leq B_{reduce} \cdot t < B_0 \\ 0, & \text{otherwise.} \end{cases} \tag{3}$$

This bonus reward is given in addition to the terminal state reward but only for the terminal state S_T as stated in Eq. 4. The terminal state is reached when the diver arrived at the ship or was found i_{df} times in a row (successive diver detections). To count the diver as found it has to be within a defined distance from the maximum located by the SONAR system.

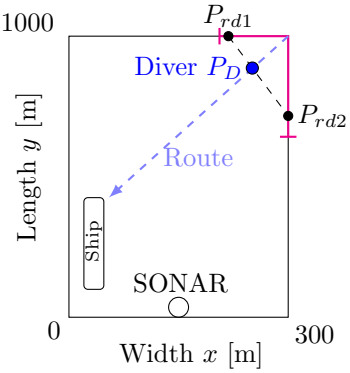

Figure 4: The goal is to detect the diver which takes the shortest way to the ship as soon as possible.

$$R'(S_T) = R(S_T) + B(t) \tag{4}$$

The distance argument r of the transmission loss $TL = 10 \cdot \log_{10}(r)$ is choosable and offers the options: *none* resulting in $TL_{none} = 1$, *cylindrical* resulting in $TL_{cyl} = 1/r$ and *spherical* resulting in $TL_{sph} = 1/r^2$. The port area is represented in polar coordinates and thus has an angle of 180° and a maximum distance of 1000 m. The SONAR is capable of doing $i_{beams} = 30$ beams which results in a resolution of 6° while the resolution is 10 m

per cell. Therefore, a 2-D array with a dimension of 30x100 is used to represent the random and calculated signal-to-noise ratio (SNR) values. The array values are represented by a normal (Gaussian) distribution around the specified noise level NL. Only the diver's position and the ship points are calculated based on the active SONAR equation from Eq. 5 and inserted at the appropriate array indices.

$$SE = \underbrace{SL - 2TL + TS}_{\text{signal}} + \underbrace{DI}_{\text{gain}} - \underbrace{NL}_{\text{noise}} - \underbrace{DT}_{\text{threshold}} \tag{5}$$

The overall signal-to-noise values val from the observation array are normalized by $SL_{max} + DI_{max}$ where DI_{max} is the highest directivity index of all modes and SL_{max} the highest source level. Values which are outside of the scanning field are marked with the value $Q = 2$ but this is also changeable. The normalization is shown in Tab. 1. The diver's position is estimated

SNR value val [dB]	**Normalized value**
$val \leq 0$	$norm_{min} = 0$
$val/(SL_{max} + DI_{max})$	$(norm_{min} \ldots norm_{max})$
$SL_{max} + DI_{max} \leq val$	$norm_{max} = 1$
outside of scanning area	$Q = 2$

Table 1: Normalizing of the observation signal-to-noise ratio values.

by determining the maximum SNR value in the observation array after executing the SONAR mode and removing unrealistic areas. The distance between the actual diver position $p(r, \theta)$ in Polar coordinates and the current maximum value in the SONAR observation array is determined. Therefore, the array indexes of the maximum value are converted back into the Polar coordinates $q(s, \psi)$ by multiplying with the corresponding resolution. The Euclidean distance is calculated with Eq. 6. For instance, the 2-D array index [13][90] would be converted back to an angle of $13 \cdot 6° = 78°$ and a distance of $90 \cdot 10\,\text{m} = 900\,\text{m}$ which means that the maximum SNR is located at $q(900\,\text{m}, 78°)$. If the diver is located at $p(905\,\text{m}, 80°)$ the calculated distance would be $d_{euc,polar}(p, q) \approx 32\,\text{m}$.

$$d_{euc,polar}(p, q) = \sqrt{r^2 + s^2 - 2rs\cos(\theta - \psi)} \tag{6}$$

The resulting distance is used to calculate the reward value. The reward function is given in Eq. 7 with d being the distance and the default values are $R_{min} = 0$ and $R_{max} = 100$ and thus smaller distances gives higher rewards as shown in Tab. 2. The distance from the above example would result in a reward of 3.125.

$$R(d) = \max\left(R_{min}, \min\left(\frac{1}{d} \cdot R_{max}, R_{max}\right)\right) \tag{7}$$

Environment

The simulation is written in Python 3 and provides about 26 individual configurable parameters that result from the environment rules. An option exists to shrink the 2-D observation array (30x100) to a 1-D array (1123x1) which reduces the amount of inputs to the neural network. The 2-D array could be used for an image processing approach where the values outside of the port are 0. An example output for the near field sector scan is shown in Fig. 5.

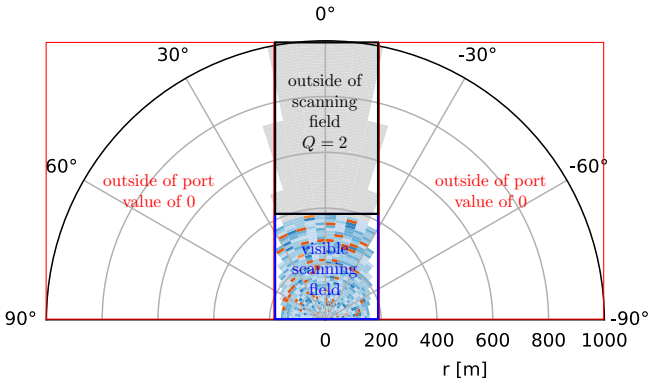

Figure 5: Illustration example of the near field sector scan for the differences between the returned 2-D (full rectangle) and 1-D (blue and black rectangle) SONAR observation.

Every time step in the environment simulates the behavior of the real-world environment. The general flow with input and outputs is depicted in Fig. 6. The action is an integer number representing the actual selected action, the observation is the observation of the environment which must always have the same shape and size. The reward is a floating point value representing the reward for the selected action and done indicates if the training episode is finished (reached the terminal state S_T).

Distance d [m]	**Reward** $R(d)$
0	100
0.1	100
1	100
2	50
5	20
10	10
100	1
1000	0.1

Table 2: Expected rewards for given distances d.

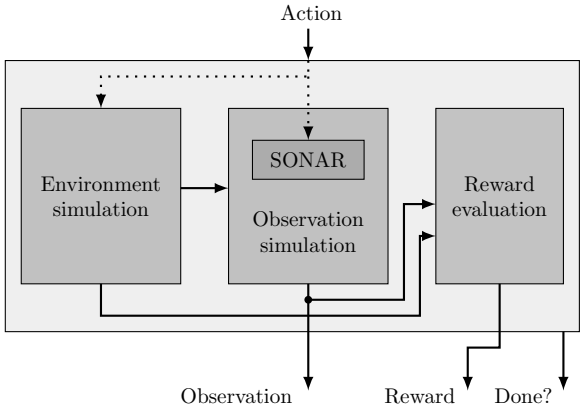

Figure 6: Illustration of the flow for one step in the environment.

Realization

For the training a deep Q-network (DQN) [1] algorithm from the TensorFlow Agents[1] library is used. As input to the neural network the 1-D observation array approach

[1] `https://www.tensorflow.org/agents`

is used. The agent performs a total of 10000 iterations where every time a full episode is collected with a replay buffer capacity of 1000 time steps. Initially two episodes are collected. The batch size is 32 and the learning rate η is 10^{-3}. The agents performance is evaluated over 10 episodes. The environment settings were chosen in a manner to simplify the learning process so that for example the sending power is 100 W with no transmission loss and a low background noise of $\mu_{NL} = 5$ with $\sigma_{NL} = 2$. The diver doesn't use an underwater scooter and thus swims with a speed between 0 and 1 m/s. The maximum distance in which a diver counts as found is 100 m and only the diver reflects the emitted signal. The neural network consists of two fully-connected layers with 75 and 40 neurons. Therefore, the trainable parameters are about 87000.

Evaluation

The average episode length of the agent 'DQN' is quite high with a mean of 1528.2 time steps for the training and 1506.8 time steps for the evaluation. The average return metric is given in Fig. 7 and the minimum reward is 88.3, the maximum is 4601.3 and the average reward is 752.5 for the training. The average reward of the evaluation is 689.3. Especially in the last episodes the training return increased strongly and therefore also the evaluation reward increased strongly. This indicates that a better policy was found than the agent used in the episodes before. Thus the agent learned that there are better actions to take to increase the total reward per episode than the ones used before. In Tab. 3 the results of different agents are shown. Obviously our goal of finding the diver quickly is not important to the agent as he is just trying to maximize the reward. Therefore, the agent 'DQN' found an *unexpected* way to exploit the environment rules. The agent chooses a combination of near field sector scan and far field sector scan while ignoring the intermediate sector scan. The learned policy π after 10000 training iterations of the agent was to choose about 85000 times the near field scan mode and about 85000 times the far field scan mode, but nearly no intermediate scans (all numbers are on the basis of 100 episodes). This allows to circumvent the termination condition and maximizing the reward with this strategy.

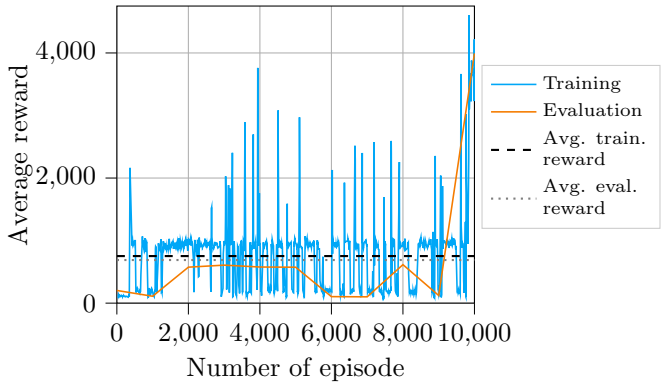

Figure 7: Visualization of the average reward metric for training and evaluation.

Agent	Action(s)	Avg. steps / episode	Avg. reward / episode
random	randomly chosen	470.1	641.4
near	near field scan	2438.9	588.6
interm.	intermediate scan	588.7	111.9
far	far field scan	13.6	106.6
DQN	learned policy π after 10000 iterations	1723.9	3517.4

Table 3: Results for different agents in the simplified environment averaged over 100 episodes.

Discussion and Outlook

In this work the concept of deep reinforcement learning was combined with SONAR to reach the goal of a cognitive SONAR system. Therefore, the SONAR modes, the simulation environment and the rules were briefly introduced and the results of the simulation were presented. The evaluation of the agent results revealed that the environment represents exactly what a human would assume because the far field sector scan is the best mode to choose if the diver appears from the upper right side. But it also shows that the reward function is not bulletproof as it allows the agent to maximize the reward by doing something else than the intended fast finding of the diver. The high flexibility of the environment created allows it to be used as a foundation for further research on cognitive SONAR system and different deep reinforcement learning algorithms. It also shows the common problem of defining the reward function so that it exactly does what is intended and only that. The reward function is the essential concept of reinforcement learning but also very difficult to design [2]. Therefore, different approaches for the reward function should be tested since there are many different methods possible. One could give bonus rewards, discourage specific actions in specific states, give positive and/or negative rewards, limit reward values within a range, etc. Once a bulletproof reward function was found more modes could be implemented for the cognitive SONAR so that it also takes the energy consumption into account since it is a rare case that a diver is in the port so a continuous scan might not be necessary.

References

[1] Volodymyr Mnih et al. Human-level control through deep reinforcement learning. *Nature*, 518(7540):529–533, February 2015. Number: 7540 Publisher: Nature Publishing Group.

[2] Andrea Lonza. *Reinforcement learning algorithms with Python: learn, understand, and develop smart algorithms for addressing AI challenges.* Packt Publishing, 2019.

[3] Richard S. Sutton and Andrew G. Barto. *Reinforcement Learning: An Introduction.* The MIT Press, Cambridge, MA, USA, 2018.

[4] A. D. Waite. *Sonar for Practising Engineers.* Wiley, Chichester, 2002.