Combination of Reinforcement Learning and Neural Networks

Anastasia Noglik and Josef Pauli

Abstract—Reinforcement Learning has been already successfully applied to the problem of a target oriented robot navigation. However one of its drawbacks is the usually observed low convergence rate of the learning progress. To weaken this disadvantage the present work suggests to use a control function which is represented by a neural network to prevent the agent of collisions with objects if a dangerous situation has been recognized. This leads to a better performance of the learning process and to a reduced number of collisions. Since an evolutionary algorithm is used to develop the neural network, little effort is necessary to train the network.

I. INTRODUCTION

Reinforcement Learning (RL) is a sub-area of machine learning and describes methods for solving a class of problems. Inspired by psychological theory, RL is concerned with how an agent ought to take actions in an environment so as to maximize some notion of long-term reward. Reinforcement Learning algorithms attempt to find a policy that maps states of the world to the actions the agent ought to take in those states.

In the last few decades a lot of interesting work have been done in this area. Several algorithms have been developed to solve various problems including robot control, elevator scheduling or telecommunications. RL was also successfully applied in the development of game strategies of backgammon and chess.

However, similar to most other algorithms there are also some main drawbacks, which have to be taken into account. One common issue is the poor convergence of these approaches [9]. The reason of it usually lies in the high dimensionality of the corresponding policy/value function. Also a large number of episodes is needed to find a suitable strategy. In case of planning and learning Sutton et. al. addressed this problem by developing the Dyna-Q algorithm [11]. Here the policy/value function is influenced based not only on the real experience of an agent, but also on its simulated experience, which is produced through the model-based processes including in this algorithm.

In the work presented here, this problem is addressed by incorporating additional context knowledge about the application domain. This additional knowledge can highly improve the convergence of the reinforcement method and usually can be easily extracted from the corresponding domain. However a lot of the existing algorithms discard this kind of knowledge or do not provide any way to incorporate it in the learning process.

There are different forms of context knowledge. One example is knowledge which is extracted from the information which is already used in the learning process of the agent. This context knowledge can be provided to the learning process in form of e.g. a heuristic function, see [2], [5]. An other form of context knowledge is extracted from information about the environment which is not used in the learning process. This context knowledge is used in a control function for the agent.

The functionality of the proposed method is illustrated in case of a goal oriented navigation problem. Here the aim of the agent is to learn a short and collision free path from a given starting state to a target point through a complex environment. The agent has two sources of information available. The idiothetic source is used to specify the agent’s position in a state space. The allosthetic source, in this work the infrared sensors, is used to extract the required additional knowledge. This knowledge serves as a control function to save the agent from dangerous situations, like unintended bumping into the wall. In this method the control function is approximated using an evolutionary gained artificial neural network (ANN) (NEAT Method [10]).

An example for the combination of both methods is the NEAT+Q algorithm, [13]. This algorithm combines the power of RL methods with the ability of NEAT to learn effective representations. The NEAT+Q algorithm is an extension of the NEAT method using RL. The algorithm which is proposed in the present work is an extension of RL using ANN which is gained using NEAT.

The incorporation of the control function into the learning process involves several interesting aspects. First of all the two information sources have to be fused together. Secondly the question of an appropriate discretization of the action space is raised. The way the control function influences the learning process is described in more detail in the remainder of this paper.

The present work investigates the performance of the proposed method by varying the influence strength of the control function on the learning process. The investigations are performed with a Scorpion robot from Evolution Robotics with a certain mechanical design and infrared sensors. Therefore the results are not valid for all models of the robot. But the used methods for investigation and determination of optimized parameters are applicable to all types of agents.

II. BACKGROUND

In the present work the linear gradient Sarsa(λ)-Algorithm is used as a basic method [7]. Tile Coding is selected as discretization method for the state space. The action-state function is approximated by a linear function. The proposed extension uses local information as context knowledge which is different from the global information in the state of the agent. Hence the basic method can be replaced by
other Reinforcement Learning methods. The state space can be discretized or represented by different approaches.

A. **Sarsa(λ) Algorithm**

Reinforcement Learning is a synonym of learning by interaction. Fully adaptive control algorithms which learn both by observation and trial-and-error are a promising approach in machine learning. RL is defined as the learning of a mapping from situations \( S \) (\( S \) is the set of possible states) to actions \( A \) (\( A \) is the set of actions) so as to maximize an accumulated scalar reward or reinforcement signal \( r \). Rewards \( r \) are gratifications or punishments and are a means of informing the agent about the target.

An action-state value function \( Q^\pi(s, a) \) is defined as the value of taking action \( a \in A(s) \) in the state \( s \in S \) under a policy \( \pi \) and provides a measure for the quality of the \((s, a)\)-pair. The function is defined as an expected return of future rewards:

\[
Q^\pi(s, a) = E_{\pi} \left[ \sum_{k=0}^{\infty} \gamma^k r_{t+k+1} | s_t = s, a_t = a \right]
\]

where \( \gamma \) is a discount factor [12].

Sarsa(\( \lambda \)) is an on-policy learning method, which continually estimates \( Q^\pi \) for the behavior policy \( \pi \) [12]. Learning is an iterative process. In the beginning the agent owns a random suboptimal value function and strategy. The basic learning step updates a single \( Q \)-value, \( a_{t+1} \) is obtained from the \( \epsilon \)-greedy policy that uses values from the estimated \( Q \)-function. Subsequently the \( Q \)-value for \((s, a)\) is updated as follows:

\[
Q(s_t, a_t) \leftarrow Q(s_t, a_t) + \alpha [r_{t+1} + \gamma Q(s_{t+1}, a_{t+1}) - Q(s_t, a_t)]
\]

where \( \alpha \) is the learning rate. The \( Q \)-function is approximated by a linear function.

B. **Tile Coding, Function Approximation, State Space, Action Space**

1) **Tile Coding:** The success of RL with large continuous state spaces depends on an effective Q-function approximation. Of the many function approximation schemes proposed, tile coding offers an empirically successful balance among representational power, computational cost, ease of use and it has been widely adopted in recent RL work[8].

To generalize the state representation the table of \( Q \)-values can be approximated using a representation of the state space with tile coding [1]. Tile coding is a form of coarse coding. In tile coding the receptive fields of features are grouped into exhaustive partitions of the input space. Each such partition is called tiling, and each element of the tiling is called tile. Each tile is the receptive field for one binary feature. According to the chosen discretization of the state space, one state is represented by one feature only.

2) **Function Approximation:** The Sarsa(\( \lambda \)) algorithm with function approximation was first explored in [7]. The \( Q \)-function is approximated in the present work by a linear function. The representation of the \( Q \)-function by a parameter vector is defined as:

\[
Q(s, a) := \sum_{i \in F_s} \theta_a(i) \cdot \phi_s(i), \quad F_s := \text{set of features present in } s
\]

The respective \( Q \)-value corresponds then to the value of the currently involved binary feature.

3) **State Space:** The three axes of the state space in the defined navigation problem correspond to the horizontal \( x \) and vertical position \( y \) of the agent and its orientation \( o \). The discretization of the state space significantly influences the performance of the learning process. The discretization \( N_x \times N_y \times N_o \) specifies the number of feature centers along the positions and orientation axes respectively. The \( x, y \)-plane was discretized according to the robot size. Each rectangle of this plane was as big as it does not exceed the size of the robot. An example of such a discretization can be seen in figure 1.

4) **Action Space:** An action is a vector which consists of two scalar values forward velocity \( v \) and rotation velocity \( \beta \), where the unit of \( v \) is centimeters per minute and the unit of \( \beta \) is degrees per minute. The action space of the to be learned Q-Function consists of three discrete actions: forward movement \((v := 10, \beta := 0)\), turn left \((v := 0, \beta := 15^\circ)\), turn right \((v := 0, \beta := -15^\circ)\). The control function which is represented by an ANN gives only real values, \((v, \beta) \in [0, 10] \times [-90^\circ, 90^\circ]\).

C. **Robot and Intelligent Agent**

In the present work the terms mobile robot and intelligent agent are used synonymously. The experiments with the agent are performed as a numerical simulation. But as input to the agent true sensor values are used which have been measured previously with the real robot. This results in a true-to-reality simulation [4]. That is the reason why the intelligent agent in the simulation is called robot. The model of the mobile Scorpion robot (agent) used in the simulation is
equipped with seven infrared sensors \((s_1, \ldots, s_7)\), see figure 2.

D. NEAT Method

Neuro Evolution of Augmenting Topologies (NEAT) applies the evolutionary approach for the development of the topology as well as for the determination of the weights of an Artificial Neural Network (ANN). A NEAT is used in the simulation for the training of an ANN, which is able to take over the control of the robot. Each ANN is an individuum with the same input (7 sensors and 1 bias node) and output (2 control commands for rotation and forward velocity) layers. NEAT has been selected due to a high adaptation ability and relatively simple application. An extensive description of the system provides the work [10].

The counter of the covered angle from the given center in the pre-defined direction has been selected as fitness function. Examples for more fitness functions are described in [6] and [3]. Different scenarios have been used in the simulation during the training phase to train the required skills. For example the agent learns a fast reaction to suddenly occurring objects when its starting point is very close to a wall. The ANN which represents the control function is a kind of reactive behavior pattern for robots. The ANN encodes a direct mapping of the sensor values on the control commands. The resulting ANN intervenes during dangerous situations to prevent collisions with objects.

E. Control Function approximated by ANN

An example of an ANN which has been trained with the above described method, fitness function and scenarios is depicted in figure 3. This ANN has been also tested successfully in reality. Yellow marked edges have a positive weighting, blue marked edges have a negative weighting. The thicker the line is, the higher is the absolute value of the weight for the shown connection. The green lines are the bias connections. The red connections mean a feedback.

III. CONCEPT OF EXTENDED METHOD

To weaken Reinforcement Learning disadvantages the present work suggests to use a control function which is represented by a neural network to prevent the agent of collisions with objects if a dangerous situation has been recognized. This leads to a better performance of the learning process and to a reduced number of collisions.

To extend the basic method by control function, the method has to be modified in such way, that the learning process is affected as little as necessary and the benefit is as high as possible. On the one hand RL should not notice the extension, and on the other hand the control function shall protect the robot from collisions as good as possible. Such a control function can not only be represented with an ANN which is achieved with an evolutionary approach. But the control function can also be gained e.g. by using control theory or fuzzy logic. Such a control function has to be generated only once and can be used as extension of the basic method for different problems. Therefore it is not necessary to consider the time which is needed for generation of the control function in the overall time which is needed for the solution of the problem.

A. Problems and Solutions

The robot receives sensor values which are in the range of \([0; 80]\). The directions of the robot’s sensors are depicted in figure 2. A collision with the environment is signaled, if a sensor measures the minimum value below a predefined threshold, so \(s_{min} = \min_{i=1, \ldots, 7} s_i < s_{\text{limit}}\), whereas here \(s_{\text{limit}} := 15\). The basic and extended methods of the learning process have the following sequence: After the lowest sensor value \(s_{\text{min}}\) falls below the given limit of \(s_{\text{limit}}\) the collision is signaled and the agent is displaced to the starting point. The eligibility traces are set to zero and the episode is continued.

The two important questions are: 1. When is the situation as dangerous, so that a neural network has to take over the control of the robot? 2. When does this intervention disturb dramatically? These questions have strongly influenced the development.

The balance between the intervention in the learning process and the benefit of the control function has to be right so that the method can be used. To resolve the previously mentioned dilemma, several experiments have been accomplished to find an appropriate pair of the following parameters: \(s_{\text{Critical}}\) and \(\beta_{\text{Discretisation}}\), see figure 4 rows 10 and 14. Thus the control function is only used, if \(s_{\text{min}} \in [s_{\text{limit}}; s_{\text{Critical}}]\). The learning process with RL has global information as basis. Each state will be extracted from odometry data and represented in the coordinate system of the starting point. The answer of the neural network is determined based on local information. The artificial neural network provides the answer to the sensor values. The result is a kind of data fusion. The control function is approximated by the neural network with sensor values (local information) as input. The local information can be different although the state from the agent’s perspective is the same. The reasons are the use of different kinds of information, the discretization of the state space and the inaccuracy of the sensor values, for which the variable \(\beta_{\text{Discretisation}}\) is responsible.

B. ANN+Sarsa(\(\lambda\))-Algorithm

The extension of the basic Sarsa(\(\lambda\)) algorithm is depicted in figure 4. The difference is the addition of rows 9 to 18 and 32 to 34. This block is used, if the minimum sensor value \(s_{\text{min}} = \min_{i=1, \ldots, 7} s_i\) goes below the specified \(s_{\text{Critical}}\). In row 11 the answer of the ANN on the sensor values
\((s_1, \ldots, s_7)\) is taken. The rows 14 – 18 are a kind of projection of the continuous action \((v, \beta)\) in the discrete action space \([\text{forward movement}, \text{turn right}, \text{turn left}]\). In this case it is a continuous action space, but it is projected into an ordinary discrete action space by means of the parameter \(\beta_{\text{Discretisation}}\).

It has been decided to use "limited learning" instead of an ANN intervention, see row 13 in figure 4. The Temporal Difference (TD)-error value \(\delta_t\) is set to a constant value \(r_{\text{Reflex}}\). That means, if the ANN takes over the control, the Q-function for the current state and the projected action is determined explicitly independent on the global target and not learned, compare Eq. (1). Hence the agent has no possibility to learn in the critical points.

C. Parameterization of the Extended Method

Two variables are responsible for the right balance between the intervention into the learning process and the improvement by using an ANN. The parameter \(s_{\text{Critical}}\) determines when the ANN is taking over the control. The parameter \(\beta_{\text{Discretisation}}\) is responsible for the projection of the action provided by the ANN in the discrete action space. \(r_{\text{Reflex}}\) outputs always the same value, so that the agent gets a certain value at the critical points. \(r_{\text{Reflex}} = -5\) is more painful for the agent than each step \((r_{\text{step}} = -1)\), but it is not worse than a reward of \(r_{\text{collision}} = -20\). The gratification reward is the value \(r_{\text{target}} = +1000\).

The tests have been performed with the following parameters: \(\alpha = 0.2, \epsilon = 0.1, \gamma = 0.9, \lambda = 0.8\). This is a reasonable parameterization. No procedure for the selection of these learning parameters has been used.

IV. Evaluation Strategy

The performance of the basic method will be compared to that of the proposed method by applying both methods to the navigation problem in the three previously mentioned environments. The discretization of all environments was chosen to be standard Tile Coding (TC). Also the used reward model was equal in all conducted experiments.

The balance between the intervention in the learning process and the benefit of the control function has to be right so that the method can be used. To resolve the previously mentioned problem, several experiments have been accomplished to find an appropriate pair of the following parameters: \(s_{\text{Critical}}\) and \(\beta_{\text{Discretisation}}\). Thereby two criteria have been used to compare the performance of the two methods.

The first criterion is the learning progress defined as:

\[
\text{LearningProgress}_{BE} := \frac{1}{N} \sum_{i=1}^{N} s_{\text{step}_i}
\]

where \(s_{\text{step}_i}\) is the average number of steps in the \(i\)-th episode. According to the definition smaller values of this criterion indicate a better learning process. In each episode in total 30 tests have been conducted. In case of the first two environments \(N\) was chosen to be 250. In the third environment it is set to \(N = 500\). The computed values of this criterion can be found in the next section in tables I and IV.

The second criterion is the average number of wall contacts:

\[
\text{Wallcontacts} = \frac{1}{J} \sum_{j=1}^{N} \text{Wallcontacts}_{s_j}
\]

where \(\text{Wallcontacts}_{s_j}\) is the average number of wall contacts during one episode, in which in total 30 tests have been conducted. This ensures a certain statistical significance. Again in case of the first two environments \(N\) was set to 250, and in case of the third environment to 500. This criterion was chosen, because a small number of wall contacts is important in case of a navigation problem. The computed values of this criterion can be found in tables II and V.

To evaluate the influence of the ANN on the learning progress of the extended method the following third criterion has been developed. The average number of network applications depends on the configuration of the parameters and is listed in table III. Relative deviations emphasize the positive or negative influence of the extension with an ANN in comparison to the standard algorithm. The deviation value is calculated with

\[
\text{Deviation} := \left(1 - \frac{\text{number}_{\text{standard}}}{\text{number}_{\text{ANN}}}\right) \times 100\%.
\]

LearningProgress as well as WallContacts are inserted instead of number for each parameter combination.

V. Results

In the following the results for two simple environments, see figure 1 and 5, and one more complex environment are presented. A part of the complex environment is shown in figure 6. In all experiments one tiling has been used. In environment 1 and 2 the tiling has \(10 \times 10 \times 24\) tiles, in environment 3 the tiling has \(49 \times 39 \times 24\) tiles.

In table I the learning progress is listed for different parameter sets, \(s_{\text{Critical}} \in \{15, 20, 25, 30, 35\}\) and \(\beta_{\text{Discretisation}} \in \{1, 2, \ldots, 9\}\). In the lower part of table I relative deviations are listed in relation to the basic method. The tests have been performed in environment 1.

A value of 25 of the parameter \(s_{\text{Critical}}\) results in a positive impact for all values of the parameter \(\beta_{\text{Discretisation}}\). The best combination for the tested parameters \((s_{\text{Critical}}, \beta_{\text{Discretisation}})\) is \((25, 6)\). This combination results in an improvement of more than 20% for the selected criterion, the learning progress, in comparison to the basic method.

The results for the second criterion, the average number of wall contacts, are listed together with the relative deviations in table II. The parameter value \(s_{\text{Critical}} = 30\) shows the best results with improvements between 69% and 76%. But the best parameter combination for the learning progress \((25, 6)\) provides also a relative high reduction of wall contacts per episode of 59%. The improvement of the learning progress is much higher for the parameter combination \((25, 6)\) than for any parameter combination with a \(s_{\text{Critical}}\) of 30 with
### TABLE I
**Learning progress (average number of steps) for environment 1 for different parameter sets. Columns: varying parameter $\beta_{\text{Discretisation}}$ between 1 and 9. Rows: varying parameter $s_{\text{critical}}$ between 15 and 35. Lower part: relative deviation to the basic method.**

<table>
<thead>
<tr>
<th>$s_{\text{critical}}$</th>
<th>1</th>
<th>2</th>
<th>3</th>
<th>4</th>
<th>5</th>
<th>6</th>
<th>7</th>
<th>8</th>
<th>9</th>
</tr>
</thead>
<tbody>
<tr>
<td>15</td>
<td>386.31</td>
<td>386.31</td>
<td>386.31</td>
<td>386.31</td>
<td>386.31</td>
<td>386.31</td>
<td>386.31</td>
<td>386.31</td>
<td>386.31</td>
</tr>
<tr>
<td>20</td>
<td>458.71</td>
<td>406.55</td>
<td>417.48</td>
<td>451.70</td>
<td>418.66</td>
<td>426.97</td>
<td>446.28</td>
<td>422.27</td>
<td>430.91</td>
</tr>
<tr>
<td>25</td>
<td>362.54</td>
<td>344.69</td>
<td>325.13</td>
<td>333.65</td>
<td>314.66</td>
<td>301.91</td>
<td>365.4</td>
<td>334.89</td>
<td>313.72</td>
</tr>
<tr>
<td>30</td>
<td>415.17</td>
<td>385.20</td>
<td>417.87</td>
<td>411.52</td>
<td>369.50</td>
<td>377.83</td>
<td>368.47</td>
<td>380.90</td>
<td>365.78</td>
</tr>
<tr>
<td>35</td>
<td>1190.59</td>
<td>690.74</td>
<td>794.55</td>
<td>783.61</td>
<td>1045.50</td>
<td>1046.23</td>
<td>1795.96</td>
<td>696.66</td>
<td>790.50</td>
</tr>
<tr>
<td>Standard</td>
<td>0%</td>
<td>0%</td>
<td>0%</td>
<td>0%</td>
<td>0%</td>
<td>0%</td>
<td>0%</td>
<td>0%</td>
<td>0%</td>
</tr>
<tr>
<td>Relative deviation</td>
<td>-4.72%</td>
<td>-9.62%</td>
<td>-1.41%</td>
<td>2.32%</td>
<td>3.18%</td>
<td>7.7%</td>
<td>-4.97%</td>
<td>0.15%</td>
<td>0.15%</td>
</tr>
</tbody>
</table>

### TABLE II
**Number of collisions with objects (wall contacts) for environment 1 for different parameter sets. Columns: varying parameter $\beta_{\text{Discretisation}}$ between 1 and 9. Rows: varying parameter $s_{\text{critical}}$ between 15 and 35. Lower part: relative deviation to the basic method.**

<table>
<thead>
<tr>
<th>$s_{\text{critical}}$</th>
<th>1</th>
<th>2</th>
<th>3</th>
<th>4</th>
<th>5</th>
<th>6</th>
<th>7</th>
<th>8</th>
<th>9</th>
</tr>
</thead>
<tbody>
<tr>
<td>15</td>
<td>3.91</td>
<td>3.91</td>
<td>3.91</td>
<td>3.91</td>
<td>3.91</td>
<td>3.91</td>
<td>3.91</td>
<td>3.91</td>
<td>3.91</td>
</tr>
<tr>
<td>20</td>
<td>4.7</td>
<td>4.85</td>
<td>4.33</td>
<td>4.21</td>
<td>4.18</td>
<td>4.17</td>
<td>4.04</td>
<td>4.33</td>
<td>4.18</td>
</tr>
<tr>
<td>25</td>
<td>4.94</td>
<td>4.43</td>
<td>4.6</td>
<td>5.02</td>
<td>4.69</td>
<td>4.94</td>
<td>4.73</td>
<td>4.63</td>
<td>4.63</td>
</tr>
<tr>
<td>30</td>
<td>1.91</td>
<td>1.8</td>
<td>1.65</td>
<td>1.75</td>
<td>1.56</td>
<td>1.6</td>
<td>1.72</td>
<td>1.69</td>
<td>1.58</td>
</tr>
<tr>
<td>35</td>
<td>1.39</td>
<td>1.16</td>
<td>1.21</td>
<td>1.3</td>
<td>1.38</td>
<td>1.28</td>
<td>1.73</td>
<td>1.16</td>
<td>1.32</td>
</tr>
<tr>
<td>Standard</td>
<td>0%</td>
<td>0%</td>
<td>0%</td>
<td>0%</td>
<td>0%</td>
<td>0%</td>
<td>0%</td>
<td>0%</td>
<td>0%</td>
</tr>
<tr>
<td>Relative deviation</td>
<td>-20.02%</td>
<td>-23.83%</td>
<td>-10.7%</td>
<td>-7.65%</td>
<td>-6.43%</td>
<td>-3.16%</td>
<td>-10.67%</td>
<td>-6.89%</td>
<td>-6.89%</td>
</tr>
</tbody>
</table>

### TABLE III
**Number of ANN applications for environment 1 for different parameter sets. Columns: varying parameter $\beta_{\text{Discretisation}}$ between 1 and 9. Rows: varying parameter $s_{\text{critical}}$ between 15 and 35.**

<table>
<thead>
<tr>
<th>$s_{\text{critical}}$</th>
<th>1</th>
<th>2</th>
<th>3</th>
<th>4</th>
<th>5</th>
<th>6</th>
<th>7</th>
<th>8</th>
<th>9</th>
</tr>
</thead>
<tbody>
<tr>
<td>15</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>20</td>
<td>15.76</td>
<td>18.4</td>
<td>16.61</td>
<td>15.38</td>
<td>15.87</td>
<td>16.16</td>
<td>15.42</td>
<td>15.88</td>
<td>15.88</td>
</tr>
<tr>
<td>25</td>
<td>33.3</td>
<td>31.49</td>
<td>30.75</td>
<td>30.33</td>
<td>29.12</td>
<td>29.38</td>
<td>31.25</td>
<td>31.14</td>
<td>28.92</td>
</tr>
<tr>
<td>30</td>
<td>53.99</td>
<td>50.75</td>
<td>54.39</td>
<td>52.81</td>
<td>48.46</td>
<td>49.3</td>
<td>50.04</td>
<td>49.42</td>
<td>48.14</td>
</tr>
<tr>
<td>35</td>
<td>103.6</td>
<td>94.19</td>
<td>94.63</td>
<td>94.7</td>
<td>100</td>
<td>100.88</td>
<td>115.84</td>
<td>91.95</td>
<td>97.19</td>
</tr>
<tr>
<td>Standard</td>
<td>0%</td>
<td>0%</td>
<td>0%</td>
<td>0%</td>
<td>0%</td>
<td>0%</td>
<td>0%</td>
<td>0%</td>
<td>0%</td>
</tr>
</tbody>
</table>

### TABLE IV
**Learning progress (average number of steps) for environment 2 for different parameter sets. Columns: varying parameter $\beta_{\text{Discretisation}}$ between 1 and 9. Rows: varying parameter $s_{\text{critical}}$ between 20 and 30. Lower part: relative deviation to the basic method.**

<table>
<thead>
<tr>
<th>$s_{\text{critical}}$</th>
<th>1</th>
<th>2</th>
<th>3</th>
<th>4</th>
<th>5</th>
<th>6</th>
<th>7</th>
<th>8</th>
<th>9</th>
</tr>
</thead>
<tbody>
<tr>
<td>20</td>
<td>473.83</td>
<td>482.11</td>
<td>456.29</td>
<td>535.16</td>
<td>495.75</td>
<td>477.24</td>
<td>523.53</td>
<td>484.17</td>
<td>487.27</td>
</tr>
<tr>
<td>25</td>
<td>439.58</td>
<td>431.54</td>
<td>417.76</td>
<td>418.8</td>
<td>431.89</td>
<td>470.48</td>
<td>474.77</td>
<td>470.02</td>
<td>446.54</td>
</tr>
<tr>
<td>30</td>
<td>574.74</td>
<td>560.71</td>
<td>561.58</td>
<td>650.46</td>
<td>897</td>
<td>872.91</td>
<td>555.76</td>
<td>601.73</td>
<td>863.05</td>
</tr>
<tr>
<td>Standard</td>
<td>0%</td>
<td>0%</td>
<td>0%</td>
<td>0%</td>
<td>0%</td>
<td>0%</td>
<td>0%</td>
<td>0%</td>
<td>0%</td>
</tr>
<tr>
<td>Relative deviation</td>
<td>-6.49%</td>
<td>-8.35%</td>
<td>-2.55%</td>
<td>-20.25%</td>
<td>-11.42%</td>
<td>-7.25%</td>
<td>-17.66%</td>
<td>-8.81%</td>
<td>-9.51%</td>
</tr>
</tbody>
</table>
at the same time a similar reduction of wall contacts.

Another influence criterion is shown in table III. The number of ANN applications for different parameter sets. The correlation is not surprising. The higher the parameter 

\( s_{\text{Critical}} \) is, the more often the ANN is used. The parameter 

\( \beta_{\text{Discretisation}} \) has no significant impact on the number of ANN applications.

To verify the general correlations the tests are performed for different parameter sets in another environment, see figure 5. The results for the learning progress are listed in table IV and the results for the number of wall contacts are listed in table V. A smaller parameter range can be chosen, since for the parameter 

\( s_{\text{Critical}} \) the values 20 and 35 showed negative results in previous experiments. A parameter of 

\( s_{\text{Critical}} = 25 \) showed also in this environment 2 the best results for the learning progress. But the best value for the parameter 

\( \beta_{\text{Discretisation}} \) has shifted. The reason could be the high number of angles in this environment. A finer projection is needed in the discrete action space. For the parameter set (25,5) the learning progress has been improved by 3%. But for the parameter set (25,6) the agent showed a learning progress which is worse by 6%. The number of wall contacts is also higher in comparison to environment 1. But there is still an improvement of almost 30% using the parameter set (25,5), see table V. To understand the influence of the different parameter values on the learning progress, several corresponding curves have been plotted in figure 7. The red curve shows the learning progress using the basic method in the environment shown in figure 1. Starting with the initial value of 10000, maximal possible steps in one episode, it decreases steeply down to a value of 63 after the 20-th episode. Between the 20-th and the 250-th episode there is almost no improvement. The green, blue and violet curves show the learning progress of the proposed method for different parameter values. The green curve with the parameters 

\( s_{\text{Critical}} = 20 \) and \( \beta_{\text{Discretisation}} = 5 \) shows a similar development to that of the basic method, however in contrast to the basic method the number of steps starts to decrease earlier. The similarity evolves from the fact that because of the chosen 

\( s_{\text{Critical}} \) the impact of the ANN on the learning process is too small. After its activation it does not have much elbowroom to move the robot away from the wall. As mentioned above, if the distance to the wall is smaller than \( s_{\text{limit}} \), so \( s_{\text{min}} < s_{\text{limit}} := 15 \) a collision is signaled and the agent is placed to the starting point again. However, other parameter sets of the proposed method lead to different developments of the learning curves. With the greater values of the parameter 

\( s_{\text{Critical}} \), the activated ANN has much more impact on the learning process. As shown by the blue and violet curve the robot already reaches its target during the first episodes. Moreover it needs from the beginning less than 4000 steps to reach the target.

Considering this figure a general trend can be identified. The higher the value of the parameter 

\( s_{\text{Critical}} \) is, the less steps are needed for the agent to reach its target during the first episodes. The overall shape of the corresponding curves also becomes more complanate. Thus an appropriate impact of the ANN on the learning process leads to a 20% better learning progress. But the disadvantage of the proposed method can also be observed in figure 7. The minimum number of required steps in one episode is reached later than using the basic method. The conclusion is, that the robot learns faster in the first few episodes. But it learns slower in the subsequent episodes.

In figure 8 the number of wall contacts is plotted over the episodes. Again the different curves correspond to the results of the basic method and the proposed method with different parameter sets. After the 15th episode all curves show a similar development. However in the first episode the basic method and the proposed method parameterized with 

\( s_{\text{Critical}} = 20 \) and \( \beta_{\text{Discretisation}} = 5 \) produce more than 200 wall contacts. This greatly effects the agents health. Much better are the results presented by the blue and violet curves. By applying the corresponding methods to the real robot, its probability to survive is much greater, since less wall contacts have to be endured.

The correlation between the number of ANN applications and the number of wall contacts depending on the number of episode is shown in figure 9 for a parameter combination of 

\( (s_{\text{Critical}} = 25, \beta_{\text{Discretisation}} = 5) \). Both curves have been scaled to the same level to point out the similar progress. The reason is that the number of situations, in which the ANN does not help is fixed. The result is a collision with a wall

---

**TABLE V**

<table>
<thead>
<tr>
<th></th>
<th>1</th>
<th>2</th>
<th>3</th>
<th>4</th>
<th>5</th>
<th>6</th>
<th>7</th>
<th>8</th>
<th>9</th>
</tr>
</thead>
<tbody>
<tr>
<td>standard</td>
<td>4.35</td>
<td>4.33</td>
<td>4.33</td>
<td>4.33</td>
<td>4.33</td>
<td>4.33</td>
<td>4.33</td>
<td>4.33</td>
<td>4.33</td>
</tr>
<tr>
<td>20</td>
<td>0%</td>
<td>0%</td>
<td>0%</td>
<td>0%</td>
<td>0%</td>
<td>0%</td>
<td>0%</td>
<td>0%</td>
<td>0%</td>
</tr>
<tr>
<td>25</td>
<td>12.4%</td>
<td>17.15%</td>
<td>25.25%</td>
<td>23.49%</td>
<td>28.56%</td>
<td>9.77%</td>
<td>14.46%</td>
<td>15.89%</td>
<td>26.47%</td>
</tr>
<tr>
<td>30</td>
<td>54.65%</td>
<td>54.25%</td>
<td>57.83%</td>
<td>52.7%</td>
<td>53.46%</td>
<td>52.37%</td>
<td>58.19%</td>
<td>50%</td>
<td>52.35%</td>
</tr>
</tbody>
</table>

LEGEND:

- **s**: Parameter value
- **Critical**: Range of parameter values
- **Discretisation**: Range of parameter values
- **Standard**: Reference parameter value
- **ANN**: Artificial Neural Network

The table shows the number of collisions with objects (wall contacts) for environment 2 for different parameter sets. Columns: varying parameter \( \beta_{\text{Discretisation}} \) between 1 and 9. Rows: varying parameter \( s_{\text{Critical}} \) between 20 and 30. Lower part: relative deviation to the basic method.
01 Initialize $\vec{\theta}$ arbitrarily
02 Repeat (for each episode):
03 $\vec{e} = \vec{0}$
04 $s, a \leftarrow$ initial state and action of episode
05 $\mathcal{F}_s \leftarrow$ set of features present in $s$
06 Repeat (for each step of episode):
07 For all $i \in \mathcal{F}_s$:
08 $\vec{e}_a(i) \leftarrow \vec{e}_a(i) + \phi_s(i)$
09 $s_{min} \leftarrow \text{MIN}(s_1, s_2, ... s_7)$
10 if $s_{min} < s_{\text{Critical}}$
11 $a' := (\beta, v) \leftarrow \text{ANN}(s_1, s_2, ... s_7)$
12 Take action $a'$, observe reward $r$, and next state, $s'$
13 $\delta \leftarrow r_{\text{Reflex}}$
14 if $|\beta| < \beta_{\text{Discretisation}}$
15 $a \leftarrow \text{forward movement}$
16 else if $\beta > 0$
17 $a \leftarrow \text{turn right}$
18 else $a \leftarrow \text{turn left}$
19 else
20 Take action $a$, observe reward $r$, and next state, $s'$
21 $\delta \leftarrow r - \sum_{i \in \mathcal{F}_{s'}} \theta_a(i) \cdot \phi_s(i)$
22 With probability $1 - \epsilon$:
23 For all $a \in A(s')$:
24 $Q_a \leftarrow \sum_{i \in \mathcal{F}_{s'}} \theta_a(i) \phi_{s'}(i)$
25 $a \leftarrow \text{arg max}_a Q_a$
26 else
27 $a \leftarrow$ a random action $\in A(s)$
28 $\mathcal{F}_{s'} \leftarrow$ set of features present in $s'$
29 $Q_a \leftarrow \sum_{i \in \mathcal{F}_{s'}} \theta_a(i) \phi_{s'}(i)$
30 $\delta \leftarrow \delta + \gamma Q_a$
31 $\vec{\theta} \leftarrow \vec{\theta} + \alpha \delta \vec{e}$
32 if $s_{min} < s_{\text{Critical}}$
33 $\vec{e} \leftarrow \gamma \lambda \vec{e}$
34 else $\vec{e} \leftarrow \vec{0}$
35 $s \leftarrow s'$
36 until $s$ is target

Fig. 4. Algorithm of the proposed ANN+Sarsa($\lambda$)-Method with $\epsilon$-greedy policy. The difference is the addition of rows 09 to 18 and 32 to 34. In row 11 the ANN gives the answer on the sensor values $(s_1, ..., s_7)$. The rows 14 – 18 are a kind of projection of the continuous action $(v, \beta)$ on the discrete action space {forward movement, turn right, turn left}.
and a similar profile of the curves. The learning curve and the progress of the number of ANN applications are shown in figure 10. The profile of both curves is also similar. The more seldom dangerous situations occur during the episodes, the more seldom the ANN is used, and the faster the agent reached his target. The progress of the three values number of steps to target, number of ANN applications and number of wall contacts is very similar. The above described results are confirmed for the more complex environment 3, see figure 11. The learning progress is shown for the basic method and two sets of parameters $s_{\text{Critical}} = 25$, $\beta_{\text{Discretisation}} = 6$ and $s_{\text{Critical}} = 25$, $\beta_{\text{Discretisation}} = 5$. The optimal parameters have been determined in previous experiments. The agent with the proposed method reaches the target already in the first episode like in environment 1. The agent with the basic method reaches the target beginning with the 80th episode. But then it learns faster. The average number of successful target achievements is listed in table VI. The proposed method enables the agent to reach the target 430 and 439 times respectively out of 500 trials (episodes). The agent with the basic method reaches the target 417 times averaged over 30 experiments. The progress for the number of wall contacts from one episode to the next one in environment 3 is similar to the simple environment 1, see figure 12.

VI. CONCLUSION

A new method has been proposed which allows the integration of additional context knowledge in the learning process. The proposed method has been tested for a navigation problem. The extension with context knowledge results in a faster learning. The method uses a neural network additionally to the standard Sarsa($\lambda$) algorithm to avoid agent collisions with the surrounding obstacles. The neural network starts working only if the actual state of an agent was recognized as dangerous. In the remaining situations the usual Sarsa($\lambda$) algorithm is used. Several parameters are used to control the influence of the

<table>
<thead>
<tr>
<th>Table VI</th>
<th>NUMBER OF SUCCESSFUL TARGET ACHIEVEMENTS IN 500 TRIALS (EPISODES), AVERAGE OUT OF 30 EXPERIMENTS</th>
</tr>
</thead>
<tbody>
<tr>
<td>basic method</td>
<td>417.063</td>
</tr>
<tr>
<td>Proposed method (25, 5)</td>
<td>439.476</td>
</tr>
<tr>
<td>Proposed method (25, 6)</td>
<td>430.571</td>
</tr>
</tbody>
</table>
ANN. Different parameter sets lead to different results. A very frequent application of an ANN yields on the one hand to a very low wall contact rate. On the other hand no improvements but disturbances of the overall learning process can be observed. Thus several experiments have been conducted to obtain optimal parameters for the three examined environments. Thereby the above described dilemma was solved: using the estimated parameters the impact of the neural network was sufficient to improve the learning progress and to reduce the number of wall contacts to an acceptable minimum.

Because of the enhanced convergence of the learning process, less computational power is required to obtain a suitable solution. Thereby the presented method can also be applied to a more complex environment.

REFERENCES