

# Real-World Reinforcement Learning for Autonomous Humanoid Robot Charging in a Home Environment

Nicolás Navarro, Cornelius Weber, and Stefan Wermter

University of Hamburg, Department of Computer Science, Knowledge Technologie  
Vogt-Kölln-Straße 30, D - 22527 Hamburg, Germany  
{navarro, weber, wermter}@informatik.uni-hamburg.de  
<http://www.informatik.uni-hamburg.de/WTM/>

**Abstract.** In this paper we investigate and develop a real-world reinforcement learning approach to autonomously recharge a humanoid Nao robot. Using a supervised reinforcement learning approach, combined with a Gaussian distributed states activation, we are able to teach the Nao to navigate towards a docking station, and thus improve the energetic autonomy of the Nao. This work describes the control concept based on the information provided by the naomarks and 6 basic actions and was tested using a real Nao robot within a home environment scenario. This approach promises to be a robust way of implementing real-world reinforcement learning, reduces the number of model assumptions and offers a faster learning than conventional Q-learning or SARSA.

**Keywords:** Reinforcement Learning, SARSA, Humanoid Robots, Nao, Autonomous Docking, Real World.

## 1 Introduction

Reinforcement learning (RL) is a biologically supported learning paradigm [1, 3]. That allows an agent to learn through experience acquired by interaction with its environment. Conventional RL neural network architectures have an input layer and output layer. The input layer represents the agent's current state and the output layer represents the chosen action given a certain input.

The learning is carried out by good and bad feedback during interaction with the environment in a trial and error fashion. In contrast with supervised and unsupervised learning, RL does not use immediate training examples, but a reward (or punishment) is given only after a learning trial has finished (there is no feedback for intermediate steps). The reward is a scalar and indicates whether the result was right or wrong (binary) or how right or wrong it was (real value). The limited feedback characteristics of this learning approach make it a relatively slow learning mechanism, but attractive due to its potential to learn sequences of patterns.

In the literature, RL is usually used within simulated environments or abstract problems [2, 10, 11]. Those kinds of problems require a model of the agent-environment dynamics, which it is not always available or easy to infer. Moreover, a number of assumptions which are not always realistic have to be made, e.g. how is the states transition related to the agent actions, when the reward must be given, how many action and sensor noise will be used if any, etc.

On the other hand, real-world reinforcement learning approaches are scarce [7, 6, 8], mostly, because RL is expensive in data or learning steps, state space tends to be larger, it has to deal with sometimes challenging real-world setup such as safety considerations, real time action execution, changing sensors, actuators and environmental conditions, among many others.

Among the techniques used to improve real-world learning capabilities are: *dense reward functions* [7], which provides performance information in intermediate steps to the agent. Another frequently used technique is the manual *state space reduction* [7, 8] that is a very time consuming task. Other approaches propose modification and exploitation of agent's properties [6], which is not always possible. A final example of these techniques are *bash reinforcement learning algorithms* [8], which use information from past state transitions, instead of only the last transition, to calculate the prediction error function, which are a computationally demanding techniques.

The proven value of RL techniques for navigation and localization tasks motivates us to develop a RL approach for an autonomous docking problem used for recharging. This approach makes use of a supervised RL algorithm and a Gaussian distributed state activation that allows real-world RL. Our approach proves to work with a reduced number of training examples, and is robust and easy to incorporate into conventional RL techniques such as SARSA.

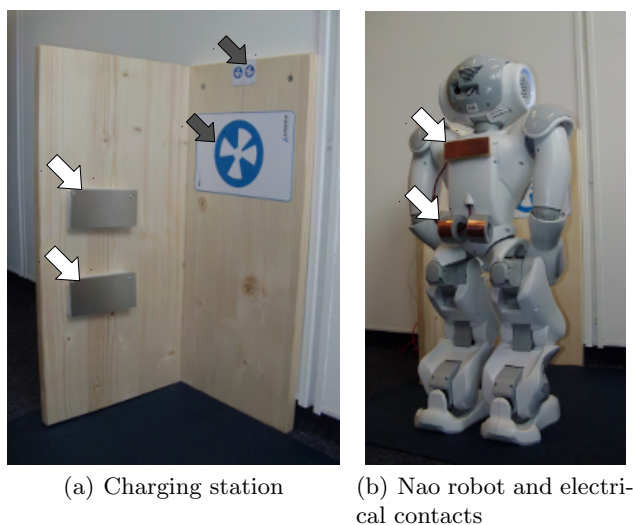
## 2 Problem Overview

There is a number of research approaches studying domestic applications [5, 9] of humanoid robots, in particular using the Nao robot. One of the Nao's limitations for this kind of environment is based on its energetic autonomy which typically does not surpass 45 min. This motivates the development of strategies to increase the robot's operational time minimizing human intervention. In this work we develop real-world reinforcement learning based on SARSA learning, see section 3, applied to an autonomous recharging behavior. This works is validated using a real Nao robot inside a home-like environment.

Several docking station and/or recharging poses are possible. The proposed solution is intended to increase the energetic capabilities of the Nao without major interventions on the robot's hardware or affecting its mobility or sensory capabilities. Despite the challenge to maneuver the robot backwards, we chose a partial backward docking, because it offers advantages such as easy mounting on the Nao, it does not limit the robot mobility, it does not obstruct any sensor, it does not require long cables going to the robot extremities and allows a quick

deployment after the recharging has finished or if the robot is asked to do some urgent task.

The prototype built to develop the proposed autonomous recharging is shown in figure 1(a). On one side white arrows indicate two metallic contacts for the recharging, and on the other side gray arrows indicate three landmarks (nao-marks<sup>1</sup>) used for navigation. The big landmark is used when the robot is more than 40 cm away from the charging station, while the two smaller landmarks are used for an accurate docking behavior.



**Fig. 1.** (a) White big arrows indicate the electrical contacts placed on the docking station and the gray arrows indicate the landmarks position. (b) Robot’s electrical connections.

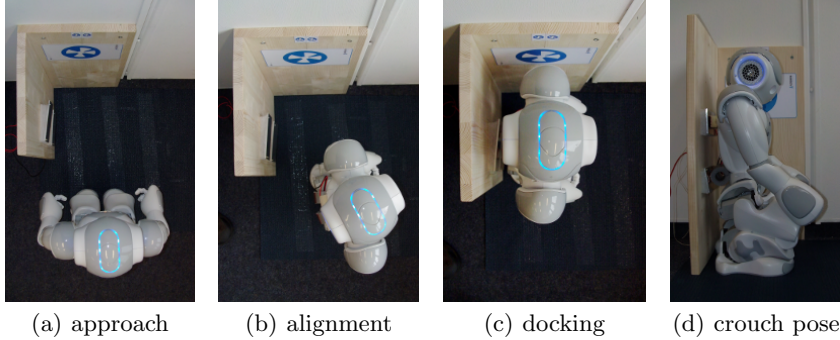
The autonomous recharging was split into two behaviors. The first behavior is a search and approach hard-coded algorithm that searches for the charging station via head scan plus robot rotation. The robot estimates a charging station’s relative position based on geometrical properties of landmarks and moves towards the charging station. This approach places the robot approximately 40cm away from the landmarks, see figure 2(a). Then the robot re-estimates its position and places itself approximately parallel to the wall as shown in figure 2(b).

The second behavior uses the SARSA algorithm to navigate the robot backwards very close to the electric contacts as presented in figure 2(c).<sup>2</sup> After reaching the final rewarded position, a hard-coded algorithm moves the robot to a

<sup>1</sup> 2-dimensional landmark provided by Aldebaran-Robotics

<sup>2</sup> In this docking phase, Nao’s gaze direction is oriented towards the landmarks

crouch pose, see figure 2(d), in which the motors are deactivated and the recharging starts.



**Fig. 2.** Top view of the autonomous robot behavior in its four different phases (approaching, alignment, docking and recharging).

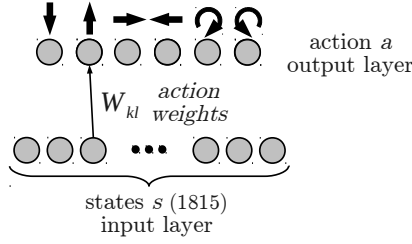
### 3 Network Architecture and Learning

We use a fully connected two layers neural network, see figure 3. The input layer (1815 neurons) represents the robot’s relative position and orientation to the landmarks. The output layer (6 neurons) represents the actions that can be performed: move forward and move backward 2.5cm, turn left or right 9° and move sideward to the left or right 2.5cm. These values were adjusted empirically as a trade-off between speed and accuracy.

The state space is formed by the combination of three variables i.e. distances to the two small naomarks and yaw (pan) head angle. Those three values are discretized as follows: The angular size of each landmark within the visual field is discretized into 10 sub-states for each landmark. These sub-states represent distances from [13, 40] cm in intervals of 2.7cm. In addition we add 2 sub-states to indicate the absence of the corresponding landmark. This leads to a total of 11 sub-states by landmark. The third variable is the head’s pan angle. The robot moves the head to keep the interesting landmark centered in the visual field, which is done to increase the detection rate. The head movements are limited to  $[70^\circ, 120^\circ[$  and the values are discretized with intervals of  $3.3^\circ$  yielding 15 new sub-states. Hence, the total number of used states is obtained by the combination of all the sub-states, i.e.  $11 * 11 * 15 = 1815$ .

As learning algorithm we use SARSA [1, 3], summarized as follow:

For each trial the robot is placed in an initial random position within the detection area. The head yaw value and the landmarks sizes are used to compute the robot internal state. Instead of using the characteristic single state activation



**Fig. 3.** Neural network schematic overview. An example of connections in the used neural network.

of SARSA, we use a Gaussian distributed stated activation with  $\sigma = 0.85$ :

$$S_j = \frac{e^{-\frac{(x - \mu_x)^2 + (y - \mu_y)^2 + (z - \mu_z)^2}{\sigma^2 \cdot 2\pi}}}{\sigma^3 (2\pi)^{2/3}} \quad (1)$$

Where,  $\mu_x$  represents the current sub-state value for “*landmark 1*”,  $\mu_y$  represents the current sub-state value for “*landmark 2*” and  $\mu_z$  represents the current sub-state value for head yaw angle. The variables  $x$ ,  $y$  and  $z$  take all the possible values of the respective sub-set. In this way a normalized state activation is computed centered on  $(\mu_x, \mu_y, \mu_z)$  and extended to the entire state space. Then the action strengths are computed:

$$h_i = \sum_l W_{il} S_j \quad (2)$$

Next, we used a softmax-based stochastic action selection:

$$P_{a_i=1} = \frac{e^{\beta h_i}}{\sum_k e^{\beta h_k}} \quad (3)$$

$\beta$  controls how deterministic the action selection is, in others words the degree of exploration of new solutions. Large  $\beta$  implies a more deterministic action selection or a greedy policy. Small  $\beta$  encourages the exploration of new solutions. We use  $\beta = 70$  to prefer know routes. Based on the activation state vector  $(S_j)$  and on the current selected action  $(a_k)$ , the value  $Q_{(s,a)}$  is computed:

$$Q_{(s,a)} = \sum_{k,l} W_{kl} a_k s_l \quad (4)$$

A binary reward value  $r$  is used. If the robot reaches the desired position it is given  $r = 1$ , zero if does not. The prediction error based on the current and previous  $Q_{(s,a)}$  value is given by:

$$\delta = (1 - r)\gamma Q_{(s',a')} + r - Q_{(s,a)} \quad (5)$$

Time-discount factor  $\gamma$  controls the importance of proximal reward against distal rewards. Small values are used to prioritize proximal rewards. On the contrary, values close to one are used to consider equally all rewards. We use  $\gamma = 0.65$ . The weights are updated using a  $\delta$ -modulated Hebbian rule with learning rate  $\epsilon = 0.5$ .

$$\Delta W_{ij} = \epsilon \delta a_i S_j \quad (6)$$

## 4 Supervised Reinforcement Learning and Experimental Results

Applications of RL usually begin with the agent’s random initialization followed by many randomly executed actions until the robot reaches eventually the goal. After a few successful trials the robot starts to learn action-state pairs based on its previous knowledge. This strategy can be applied to simulated or abstract scenarios. However, in real-world scenarios this approach is prohibitive for several reasons such as real time action execution, safety conditions, changing sensors, actuators and environmental conditions, among many others.

In order to make the docking task feasible in a real-world RL approach, we have decided to skip the initial trial and error learning as presented in [7]. We teleoperate the robot from several random positions to the goal position saving the actions state vectors and reward value. This training set with non-optimal routes was used for *offline* learning. Specifically 50 training examples with an average of 20 action steps were recorded. Then, using this training set, 300 trials were computed for the following three cases: using conventional single state activation, using Gaussian distributed state activation and a truncated Gaussian state activation. The truncated Gaussian state activation is obtained by limiting the values of  $x$ ,  $y$  and  $z$  to  $\pm i$  the value of  $\mu_x$ ,  $\mu_y$ , and  $\mu_z$  respectively and then normalized. We refer to truncated Gaussian distributions by its  $\pm i$  value as  $i$ -rings.

We compare results obtained with the weight for each case after 300 trials. After the training phase using single states activation, the robot is able to reach the goal imitating the teleoperated routes. However, the robot’s actions turn random in the states that have not yet being visited. In contrast, after training with a Gaussian distributed state activation the robot is able to dock successfully from almost every starting point, even in those cases where the landmarks are not detected in one step. This provides the Gaussian state activation a clear advantage in term of generalization. Thus a faster learning than conventional RL algorithm is obtained. For the 1-rings truncated Gaussian activation we observe lightly better results than using conventional single state activation. A partial Gaussian activation may be useful for instance when the states are very different to each other.

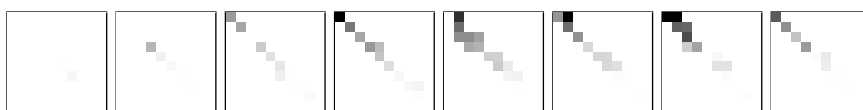
The table 1 summarizes the obtained results. We consider as a successful docking when the robot reaches the desired goal position; as a false positive when the robot’s measurement indicates that is in the goal position but is not touching the metallic contacts. Finally, we present the average number of steps

needed to reach the goal, number that decreases slightly while the robot knows more over the environment.

**Table 1.** Results for different state activation types.

State activation	% of action-state pairs learned	% of successful docking	% of false positive	Avg. n° of steps needed on success
Single	4	60	10	23.6
1-ring	34	70	30	23.5
Gaussian	100	80	10	19.1

Examples of the obtained receptive fields (RFs) after 300 trials are presented below. The goal position is represented in the upper left corner of each picture. White pixels represent unlearned action-state pairs. Darker gray represent a stronger action-state binding and thus the action is more likely to be selected when the robot is this state. The eight different pictures to each case correspond to the different action-state pairs for a particular head angles.



(a) Sample of receptive fields of “*Move to the Left*” after 300 trials with single state activation



(b) Sample of receptive fields of “*Move to the Left*” after 300 trials with Gaussian states activation restricted to one states radius



(c) Sample of receptive fields of “*Move to the Left*” after 300 trials with Gaussian states activation

**Fig. 4.** Receptive fields (RFs) of action units corresponding to the most important actions, i.e. *Move to the Left*, after 300 trials. Dark color represents the weight strength. From left to right the RFs for a few of the possible head positions are presented.

## 5 Conclusions

Motivated by the limited energetic capabilities of the humanoid Nao robot and our need for studying humanoid robots within home environments, we developed an autonomous recharging procedure for the Nao, which does not require human assistance. Autonomous docking for a humanoid Nao robot was developed for a real home like environment. Initial training examples, together with a Gaussian distributed states activation was successfully used for real-world learning.

The use of appropriate training examples proved to be a key factor for real-world learning scenarios, reducing considerably the required learning steps from several thousands learning steps to a few hundred. Additionally, Gaussian distributed states activation demonstrated to be useful for generalization and eliciting a state space reduction effect. The use of these techniques is straightforward to SARSA learning. Promising results were presented, which suggest further opportunities in real-world or simulated scenarios.

During the experimental phase, we noticed that 2-dimensional landmarks restrict considerably the detection rate, and are very noise susceptible. For future work a docking procedure using a 3-dimensional landmark is under development. Additionally, forward and backward movements will be preferred, because of the low performance of sideward movements on the Nao.

The use of a memory of successful action sequences may be of great utility in future applications. This memory can then be used for automatic offline training, when the robot is recharging or doing other less demanding tasks.

**Acknowledgments** This research has been partly supported by the EU project RobotDoc under 235065 ROBOT-DOC from the 7th Framework Programme, Marie Curie Action ITN and partly supported by KSERA project funded by the European Commission under the 7th Framework Programme (FP7) for Research and Technological Development under grant agreement n° 2010-248085.

## References

1. Sutton, R.S., and Barto, A.G.: Reinforcement Learning: An Introduction. The MIT Press, March 1, (1998)
2. Weber, C., Triesch, J.: Goal-Directed Feature Learning. In proceedings of International Joint Conference on Neural Networks. pp. 3355-3362. IEEE Press, Piscataway, NJ, USA (2009)
3. Weber, C., Elshaw, M., Wermter, S., Triesch, J., Willmot, C.: Reinforcement Learning: Theory and Applications, chap. Reinforcement Learning Embedded in Brains and Robots, pp. 11942 (2008)
4. Humanoid Nao Robot, <http://www.aldebaran-robotics.com/>
5. Louludi, A., Mosallam, A., Marturi, N., Janse, P. and Hernandez, V.: Integration of the Humanoid Robot Nao inside a Smart Home: A Case Study. The Swedish AI Society Workshop May 20-21, 2010, Uppsala University. Linkping University Electronic Press, Linkpings universitet. pp. 35-44. (2010)

6. Ito, K., Fukumori, Y., Takayama, A.: Autonomous control of real snake-like robot using reinforcement learning; Abstraction of state-action space using properties of real world. *Intelligent Sensors, Sensor Networks and Information, 2007. ISSNIP 2007.* pp.389-394. (2007)
7. Conn, K., Peters, R.A.: Reinforcement Learning with a Supervisor for a Mobile Robot in a Real-world Environment. *Computational Intelligence in Robotics and Automation, 2007. CIRA 2007.* pp.73-78. (2007)
8. Kietzmann, T.C., Riedmiller, M.: The Neuro Slot Car Racer: Reinforcement Learning in a Real World Setting. *Machine Learning and Applications, 2009. ICMLA '09.* pp.311-316. (2009)
9. KSERA (Knowledgeable Service Robots for Aging) research project, <http://ksera.ieis.tue.nl/>
10. Ghory I.: Reinforcement learning in board games. Tech. Report CSTR-04-004, CS Dept., Univ. of Bristol, May 2004.
11. Provost, J., Kuipers, B. J., and Miikulainen, R.: Self-Organizing perceptual and temporal abstraction for robot reinforcement learning. In *AAAI-04 Workshop on Learning and Planning in Markov Processes*, pp. 79-84, 2004.