Robot self-preservation and adaptation to user preferences in game play, a preliminary study

Álvaro Castro-González Farshid Amirabdollahian Daniel Polani María Malfaz Miguel A. Salichs

***Abstract*— It is expected that in a near future, personal robots will be endowed with enough autonomy to function and live in an individual's home. This is while commercial robots are designed with default configuration and factory settings which may often be different to an individual's operating preferences. This paper presents how reinforcement learning is applied and utilised towards personalisation of a robot's behaviour. Two-level reinforcement learning has been implemented: first level is in charge of energy autonomy, i.e. how to survive, and second level is involved in adapting robot's behaviour to user's preferences. In both levels Q-learning algorithm has been applied. First level actions have been learnt in a simulated environment and then the results have been transferred to the real robot. Second level has been fully implemented in the real robot and learnt by human-robot interaction. Finally, experiments showing the performance of the system are presented.**

I. Introduction

Many robotic experts have predicted the explosion of personal and social robots in the next decades. This means that robots are likely to coexist with people. For example, it is reasonable to foresee that people will be assisted by personal robots towards augmenting their capabilities and enhancing their weaknesses. It is important that these robots adapt their behaviours to different users and be able to make their own decisions.

In order to engage in such capacities, personal robots have to be endowed with learning capabilities which will be used to find out the best action at each situation.

It is foreseen that mass production of robots will lead to a production of robots with default configurations. These configurations will not match all users' preferences, thus it is really useful if a system is able to adapt its behaviour to different users' requirements. This leads to adaptive systems which adjust themselves to deal with different users using parameters that will be tuned in order to discover what a user likes or dislikes.

Moreover, these robots must be able to take their own decisions. The notion of autonomy contributes more to the usefulness of the robotic tools. Towards this, a robot will have to be proactive. Focusing on social robots, decision making is intended to engage with humans during human-robot interaction.

Álvaro Castro-González, María Malfaz and Miguel A. Salichs are with the RoboticsLab at the Carlos III University of Madrid, 28911, Leganés, Madrid, Spain acgonzal@ing.uc3m.es; mmalfaz@ing.uc3m.es; salichs@ing.uc3m.es

Farshid Amirabdollahian and Daniel Polani are with the Adaptive Systems Research Group at the University of Hertfordshire, Hatfield, United Kingdom f.amirabdollahian2@herts.ac.uk; d.polani@herts.ac.uk

Consequently, our primary problem is to increase the autonomy of robots when they have to make a decision. Therefore, a robot has to learn what to do and when, and it will depend on diverse factors like who the robot is interacting with or what the robot is doing right now.

In this work, we present how reinforcement learning can be applied to robots in the real world with two different approaches.

- Survival level: at this level, energy is the key issue. Here, the robot has to learn how to survive and it will learn the right moment to go to the charger to recharge its battery. Since the robot probably has to *die* several times to learn it, reinforcement learning has been applied in a simulated world. Later, the knowledge will be moved to the real robot and further tested.
- Interaction level: here robot has to learn how to behave with various users so they will not have to instruct the robot what to do. Therefore, robot has to learn what the users like and dislike in order to engage with them in longer interactions.

Finally, we end up with a sort of adaptive "toy" for different users which is able to determine when is the right moment to look for energy.

The proposed systems will learn when the robot must recharge its batteries and what users like to select as the next action based on their preferences.

The remaining parts of this paper is organized as follows. First, a revision of other related works is presented. Next, a brief introduction to reinforcement learning is presented focussing on the well-known Q-learning algorithm. Then, our approach is introduced and, finally, the experiments and their results are shown.

II. Different approaches

Several authors have experimented with learning on robots. In particular, reinforcement learning has been widely used [1]. Hausknecht [2] applied Machine Learning algorithms to the task of kicking the ball but the learning process is partially automated: a human was required to relay the distance of the kick and reposition the Aibo after each kick. Fidelman [3] improves grasping skill achieved via fully autonomous machine learning with all training and computation executed on-board the robot, and layered learning is applied on a physical robot. Jong et al. [4] introduce an algorithm that fully integrates modern hierarchical and model-learning methods in the standard reinforcement learning setting, although the algorithm is just tested on simulations.

Besides, gait learning on Aibo has been previously explored: Kohl [5] applied four learning algorithms to learn a parametrized walk. Saggar [6] presents results on using the policy gradient algorithm to learn a stable, fast gait. Both are fully implemented and tested on the Aibo robot platform.

All previous works focus on learning at low levels: motor level. However, Kalyanakrishnan [7] applied reinforcement learning in a multi-agent environment where the system has to decide what action to do in a robot team. Here, it seeks to maximise the duration of the episode, and robots are rewarded based on the time elapsed after every action. The system has been proved in a simulated virtual world.

Some works include people on the learning process: Kuhlmann [8] proposes a system that learns both from reinforcements provided by the environment and from human-generated advices, in a simulated soccer task.

Reinforcement learning has already been used on behaviour-based robots [9]. Ullerstam and Mizukawa [10] teach behaviour patterns based on interactions between humans and Aibo. The reinforcement learning system is taught by a remote control, used by the human and connected to Aibo. In our work, the robot also learns behaviour pattern but through a direct natural human-robot interaction. The user will behave as if s/he is interacting with a real dog.

Tapus et al. present [11] a behaviour adaptation system capable of adjusting its social interaction parameters toward customised post-stroke rehabilitation therapy based on the user's personality traits and task performance. They focus on the relationship between the level of extroversion-introversion of the robot and the user.

The work presented in this paper incorporates high-level decision making learning both on a physical robot (it will deal with direct human-robot interaction) and on a virtually simulated robot (it will be transferred to and tested on a real robot).

III. Reinforcement Learning

Reinforcement learning allows an agent to learn, through interaction with the environment, how to behave in order to fulfil a certain goal. The agent and the environment are continuously interacting, the agent selecting actions and the environment responding to those actions and presenting new situations to the agent. The environment and the proper agent also give rise to rewards that the agent aims to maximise over time. This type of learning allows the agent to adapt to the environment through the development of a policy. This policy determines the most suitable action in each state in order to maximise the reinforcement. The goal of the agent is to maximise the total amount of reward it receives over the long run [12].

The agent that uses reinforcement learning tries to learn behaviour through trial and error interactions with a dynamic environment. An agent is connected with its environment via perception and action. On each step of iteration the agent receives information about the state s of the environment. The agent then chooses an action a. The action changes the state of the environment (s'), and then the agent receives a reinforcement signal r. The goal of the agent is to find a policy, mapping to states to actions, that maximises some long-run measure of reinforcement [13].

Reinforcement learning has been successfully implemented in several virtual agents and robots [14] [15] [16] [17] [18] [19] [20]. One of the main applications, for robots or agents, is the learning of complex behaviours as a sequence of basic behaviours. Those complex behaviours allow to optimise the adaptation of the agent or robot to its environment. The reinforcement learning algorithm named Q-learning [21] has become one of the most used in autonomous robotics [22].

A. Q-learning algorithm

The goal of the Q-learning algorithm is to estimate the Q values for every state-action pair. The Q value is defined as the expected reward for executing action a in state s and then following the optimal policy from there [23]. Every $Q(s,a)$ is updated according to:

$$Q(s,a) = (1-\alpha)\cdot Q(s,a) + \alpha\cdot(r+\gamma V(s')) \qquad (1)$$

where:

$$V(s') = \max_{a\in A}(Q(s',a)) \qquad (2)$$

is the value of the new state s' and is the best reward the agent can expect from the new state s'. A is the set of actions, a is every action, r is the reinforcement, γ is the discount factor and α is the learning rate.

The learning rate α ($0 < \alpha < 1$) controls how much weight is given to the reward just experienced, as opposed to the old Q estimate [24]. This parameter gives more or less importance to the learnt Q values than new experiences. A low value of α implies that the agent is more conservative and therefore gives more importance to past experiences. If α is high, near 1, the agent values, to a greater extent, the most recent experiences.

Parameter γ ($0 < \gamma < 1$) is known as the discount factor, and defines how much expected future rewards affect decision now. A high value of this parameter gives more importance to future rewards. A low value, on the contrary, gives much more importance to current reward [24].

A policy π defines the behaviour of the agent. A deterministic policy $\pi : S \longrightarrow \Pi(A)$ is a function that relates, with probability 1, the actions $a \,\epsilon\, A$ that must be taken, with every state $s \,\epsilon\, S$.

As previously mentioned, the final goal of the agent is to learn the optimal policy, the one that maximise the total expected reward. This policy relates, with probability 1, the actions that must be taken in every state. Once the optimal function $Q^*(s,a)$ is obtained it is easy to calculate the optimal policy, $\pi^*(s)$, considering all the possible actions for a certain state and selecting the one with the highest value:

$$\pi^*(s) = \arg\max_a Q(s,a) \qquad (3)$$

IV. Applying Reinforcement Learning

In this paper, authors propose a system where reinforcement learning is applied to the robotic platform Aibo [25]. Aibo is a well-known pet-style robot which was designed to maintain a lifelike appearance. Using this robot, two main goals have to be achieved:

1) Energy autonomy: the robot has to learn when it must charge its battery and therefore how to *survive*.
2) User's preferences: the robot has to remember which behaviours are most rewarded by the user.

In both cases Q-learning will be employed as the learning algorithm, but the approaches will be different:

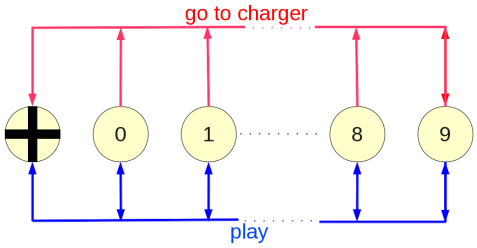
1) Energy autonomy: The learning has been simulated in a virtual environment and, later, the knowledge has been transferred to the real robot [26].
2) User's preferences: The learning has been achieved in the real world through direct interaction between users and the robot.

A. Energy Autonomy

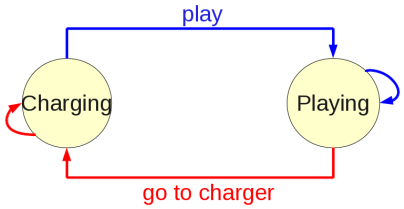
Autonomous robots must be able to recharge their batteries by themselves in order to obtain energy independence. Additionally, the precise moment when this must be done is also crucial: if the robot is continuously charging its battery it will not do other tasks or will frequently interrupt others; on the other hand, if the robot does not go to the docking station on time, it risks runing out of battery. Since this leads the robot to *die* and as the robot could *die* several times to discover the right action to do, simulation was chosen as the best way to learn from errors.

In order to learn when to go to the charger before the batteries expire, we defined a two-dimensional state space: the first dimension considers what the robot is doing: *charging* (the robot is plugged to the docking station) or *playing* (the robot-dog is playing with the user); the second one is the level of the battery which has been divided in levels from 9 (full battery) to 0 (almost empty). In addition, the state *dead* is included to identify when the battery is depleted. Besides, two actions were defined: to *play* and to *go to charger*. The former means to execute any game included in its set of actions. The latter denotes the action that moves the robot to the docking station and remains there until the batteries are full. Simulations were run considering the dynamic evolution of the robot's battery. The states and transitions for both levels are depicted on figure 1.

Figure 1a shows the transitions from one battery level to other. State 9 means full battery, state 0 is the lowest battery level, and the black cross represents when the robot has run out of batteries and it is not working anymore. *Play* action always transits from higher levels to lower ones or to the same state, because this action implies a waste of energy. Since *go to charger* moves the robot to the docking station and charge its battery, the next state after this action will be 9 (full battery). However, during this action the robot can discharge its battery and transits to the *dead* state.



(a) Battery levels states



(b) Action states

Fig. 1. Both dimensions of state space for survival learning

The second dimension, where the kind of action the robot is executing is considered, is presented on figure 1b. The *play* action triggers the transition to *playing* state or it remains in this state. In the same way, *go to charger* implies a transition to *charging* state.

The learning has been broken into subsequences called episodes and hence this is an episodic task. Episodes must have default initial states and terminal states. Each episode starts at the default initial state and keeps running until the terminal state is reached. After that, a new episode starts again. In this work, the default initial state is when the robot is playing and its battery is complete (playing-9) and the final state is, just as expected, when the robot *dies*. The learning session was limited to two thousand episodes.

During each episode, at each iteration an action must be selected. This selection will consider the utility of an action *a* from a state *s*: this is the Q-value *Q(s,a)* presented on section III-A. Action selection has to deal with the balance of exploration and exploitation [12]. At exploration all actions are equally chosen and therefore level of exploration represents the probabilities of executing actions different than those with the highest Q-values. At exploitation, the action with highest Q-value is chosen. In order to compensate it, the action probabilities are weighted according to a function of their estimated value. This is termed as the softmax method for action selection. In this work, Boltzmann distribution has been used as the softmax method and probability for selecting an action is given by (4).

$$P_s(a) = \frac{e^{\frac{Q(s,a)}{T}}}{\sum_{b \epsilon A} e^{\frac{Q(s,b)}{T}}} \tag{4}$$

where $Q(s,a)$ is the given Q-value for action a in state s, and A represents all possible actions in state s; T is the *temperature* and it weights exploration and exploitation. High T gives the same likelihood of selection to all possible

actions and the next action will be almost randomly selected; low T enforces actions with high values.

In this experiment, *temperature* has been empirically set to a value of 200 which denotes a high temperature. Consequently exploration will dominate during the learning. This is reasonable since the robot has to try all options in order to learn which action is the best one for each state.

The rewards determine what the robot learns by reinforcement learning. The aim was that the robot discovers when to recharge its batteries in order to continue living, the goal can be expressed as to keep the robot playing as long as possible because this means that it survives. Therefore, just after the action *play* has been executed, it is rewarded by one hundred units.

For this learning session the parameters used by Q-learning were tuned empirically:

- Learning rate (α) was set to 0.1 during the first one thousand episodes. During the last 1000 episodes α was reduced to 0.01.
- Discount factor (δ) was fixed to 0.9.

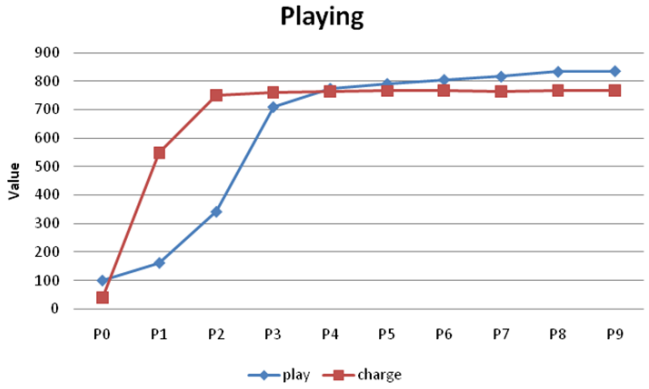
After the simulation finished, the learnt Q-values for both dimensions are displayed on figure 2. To make it clearer, the Q-values have been split into two graphs. Figure 2a represents the Q-values for all battery levels when the robot is *playing*. Figure 2b presents the state formed by *charging* and the battery levels are plotted. Vertical axes represent Q-values and the horizontal axes correspond to the different battery levels when it is *playing* (from *P9* to *P0*) or charging (from *C9* to *C0*).

All battery levels have been explored because, during the learning, the length of the actions were considered as well. This means that before an action, the robot could be at state P6 and after it, it has transited to P2. Next time, for the same action, it could move from P4 to P3. So there is not a linear transitions either in the virtual world or in the real world.

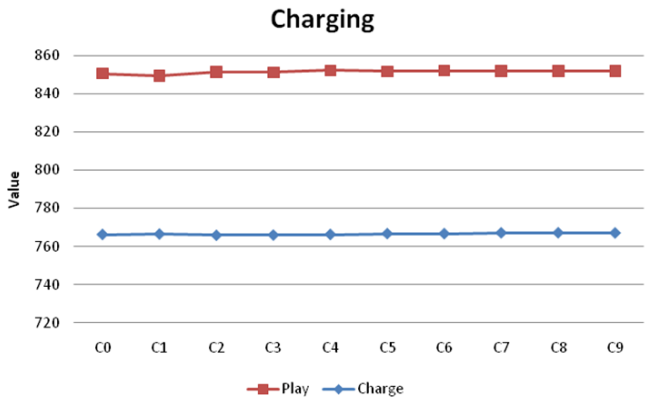
Looking at figure 2a, during *playing* state, from level 9 to level 4, the highest q-values correspond to the action *play*. This seems like obvious seeing that the battery level is high and, then, the robot must *play* and enjoy. In levels 1-3 charge action should be the selected one: the battery level is low and it must go to recharge it. It is interesting to mention the results for level 0: at this point, *play* action q-value is higher than q-value for charge action. This seems a contradiction since the batteries are almost depleted, but this situation can be explained as at this level, the robot learnt that there is not enough time to reach the docking station, so the robot prefers to play instead of dying . Let's say that AIBO thinks *"Since I'm going to die, let's have fun first!"*.

For *charging* state on figure 2b, the learnt Q-values are very clear to interpret: if the robot is *charging*, it means that *charge* action has been executed and, as it was mentioned before, this action implies a full charge of robot's battery. Hence, the next best action will be always to *play*. This corresponds to what is plotted on figure 2b where Q-values for *play* action are always over *charge* Q-values.

In summary, a policy has been learnt in order to keep the robot alive and it enforces the robot to play as often as possible without risking its "life". In case the robot is playing with battery at minimum (level 0), it must play because it is the best thing to do since it will *die* anyway.



(a) Q-Values for *playing* state



(b) Q-Values for *charging* state

Fig. 2. Q-values learnt in the two-dimension state space from simulation

B. User's preferences

Social robots are designed to interact with people but persons differ one from other. Different persons have different preferences and it is likely that people's preferred behaviours would vary between them. Based on this, it is speculated that in the next future, robots will adopt their behaviours to the circumstances and the particular users. The idea is to design a system which is able to learn the user preferences over the time and to engage the user in longer interactions by means of the acquired knowledge of the user preferences.

For this purpose, in the same manner used the previous section, reinforcement learning is used as the learning algorithm. In this case, learning in a simulated virtual environment cannot be applied since human interaction is required during the learning. Therefore, learning will be achieved in the real world using Q-learning.

Our robot Aibo is endowed with a set of behaviours and users will define their preferences among them. Users interact with AIBO several times and they will experience all possible behaviours. Three behaviours are implemented:

- Aibo searches for the pink ball and follows it.

- The robot stands on its four legs and turns to any sound user makes (clapping, whistling, calling Aibo,...)
- Aibo walks to the user and "offers" his paw to be touched (figure 4)

These behaviours can be identified by the coloured lights close to Aibo's ears as well as its attitude. If user likes what the robot is doing, he had to stroke the robot on its head or on its back. If user does not like it, he has to wait until the next behaviour is executed.

The reward is linked with the strokes, so the more the user strokes the robot the more reward the robot will receive. Austermann [27] demonstrated that touch rewards from a human can be considered sufficiently reliable for being used to teach a robot by reinforcement learning.

The set of actions is formed by the diverse behaviours presented:

- follow the ball: using the camera placed at robot's head, the robot will turn around looking for the pink ball. Once it has found it, it will approach it.
- shake the paw: the robot will approach the user who is identified by a coloured ball and, once it has reached the user, the robot sits down and shakes its paws waiting for user's reward. In this situation the reward comes from stroking the robot's head or body, or touching the robot's paw.
- turn to sound: making use of the microphones at the robot ears, it is able to identify the source of a noise. Then, the robot will rotate to the direction of the sound.

The states are based on the action the robot is doing. All possible states are:

- following the ball
- shaking the paw
- turning to sound
- not acting: this state is employed to identify the beginning and the end of an episode.

The relationship between states and actions can be observed on figure 3. The rounded corner boxes represent the states and the coloured arrows connecting two states are the feasible actions at each state. As an action ends when it is not rewarded any more, i.e. when the user does not like it, it is not logical to execute the same action just after it has ended. That is why we do not allow to repeat the same action and, therefore, there are not arrows starting and ending in the same state.

In this setup, episodes are defined by the user-robot interaction: while the user likes what the robot is doing, it will remain acting. The episodes start, from the *not acting* state, when the robot selects and executes the first action and they end at the moment the user does not reward an action; in other words, when the user does not like the action and he does not stroke it. It can be seen as any transition to the *not acting* state causes the end of an episode.

This condition produces a transition to the *not acting* state where the episode dies. After that, the robot newly starts other episode from the *not acting* state.

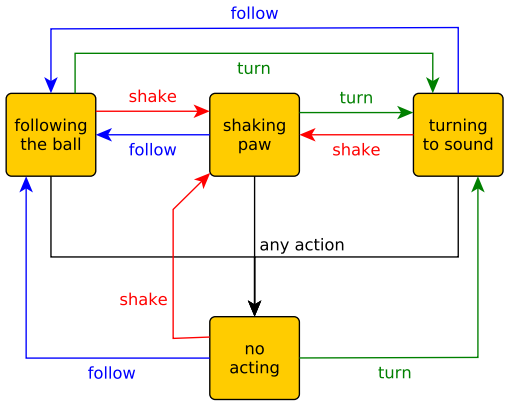


Fig. 3. State-action graph representing all possible combinations

Based on the procedures presented on Toth's work [28] with real dogs and trying to emulate their behaviours, the robot dog repeatedly asked for reward. All actions were interrupted every minute with a thirty seconds break where the robot dog will move its head, bark and flash lights, trying to attract the user's attention and obtaining rewards. Meanwhile, where a behaviour is exhibited, if the robot is not stroked ever, after three breaks in a row, it understands that the user does not like it and the reward will be zero. Once the user has patted the robot, it will keep on doing the same action until the user does not stroke it anymore. This is interpreted by the robot as if the user has got bored and it is time to change the behaviour. In this situation, the reward will correspond with the time between the first and the last stroke.

In this case, the parameters for the learning are:

- Learning rate (α) was set to 0.5 in order to speed up the learning.
- Discount factor (δ) was fixed to 0.8.

V. Experiments

In order to test the learning approaches presented in the previous section, multiple experiments were conducted. It was decided to accomplish the experiments in a controlled environment. All experiments took place in the arena: a nine squared meters square delimited by fifty centimetres high walls (figure 4).

For the first part, energy autonomy (section IV-A) has been proved in the real robot dog. The q-values learnt in the simulation were transferred to the robot and tested. This experiment was split in five sessions where each one was limited to 150 minutes hours. During this sessions, the robot decided to charge its battery on time and the rest of the time it was occupied with other tasks inside the arena. In conclusion, the robot was *living* several hours without *dying*.

Considering the users' preferences (section IV-B), three participants without prior knowledge to the game objectives were recruited to asses the proposed method. The experiment was divided into three sessions: two learning sessions where the robot semi-randomly selected the behaviours to exhibit,

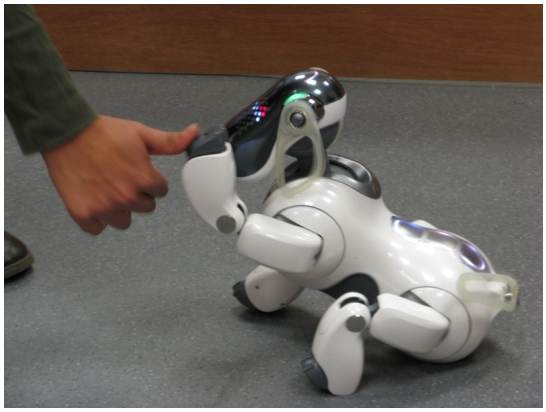


Fig. 4. User-Aibo interaction during the experiments in the arena

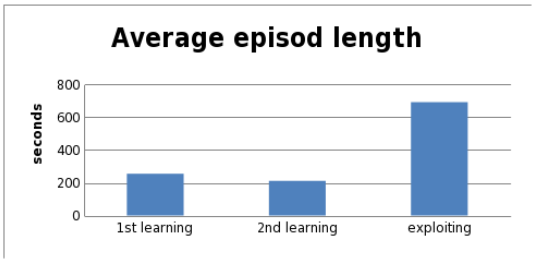


Fig. 5. Average episode length

and the last session where the q-values tuned in the two previous learning sessions were exploited. For semi-random selection, an action is randomly selected but the action corresponding to the current state is not considered. For example, if current state is *shaking the paw* then *shake paw* action will not be selected. In the exploiting session, the action selection is done using a soft-max algorithm [12] with a low temperature, which means that the action-state pairs with higher q-values will be more probably selected.

As it was explained in section IV-B, each session starts with the user and the robot inside the arena where both will interact. The session ends when the user feels tired or bored, and he does not want to interact anymore. This denotes that the duration of each session is not defined or limited by external factors but by the user. Throughout the sessions, all programmed behaviours were experienced by the user.

Before users came into the arena, an instruction/information sheet was provided. They were informed about the different behaviours and how they had to reward the robot. Besides, they were warned that if they do not like a behaviour, they just have to wait for a while and Aibo will change it. Also that when they want to finish the session they merely have to leave the arena.

During each session, there are several episodes. The episodes end when the user does not like what the robot is doing and consequently that particular action is not rewarded. Then, the robot will change the behaviour. The user does not notice or realize the transition in between two consecutive episodes. The sessions end when the user does not want to interact more because of the tiredness or boredom and he walks out of the arena. As a preliminary analysis, we focus on how many time the user is continuously rewarding robot's behaviours, this is, how many time the robot has been selecting appealing actions to user. Therefore, we consider the length of each episode during the three sessions. This is illustrated on figure 5.

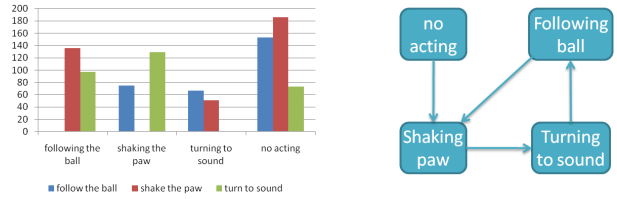
The length of episodes represents how long the user has enjoyed what the robot was doing. Additionally, since the reward is connected with the strokes, the more the user likes the action, the more strokes are received. Consequently, the more rewards, and then the longer duration for the interaction. Therefore, we can say that longer episodes reflect more favourite robot's behaviours.

As depicted, during the learning sessions, the episode length are around 200 seconds. The average length of each episode during the exploiting session is roughly three times the length of the learning sessions. This can be interpreted as the action selection being based on the learnt Q-values for each single user, this is the learnt favourite behaviours of the user, allowing the user-robot interaction to take more time. In this case, action selection has been achieved based on the most rewarded actions by that user.

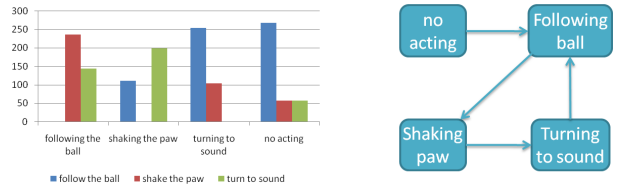
Moreover, using the learnt Q-values, the best sequence of actions in terms of interaction length can be determined. Remembering from section III-A, the Q-value represents the most suitable action from a state. Here suitability corresponds to the robot's behaviour that engages the user in longer interaction. Hence, by selecting the action with the highest Q-value from each state, the best sequence is obtained.

Figure 6 presents this situation for two users: on the left, Q-values are plotted and on the right the most rewarded sequence is shown for each person. Since the initial state is always *not acting*, the first action will be selected from the three actions: *shake*, *follow* and *turn*. From this point on, the robot will just choose between the two remaining possible actions as the third one corresponds to the action related with the current state. User A's favourite sequence (figure 6a) starts shaking the robot's paw, then the robot will turn towards sounds, and after it will follow the ball. In contrast, user B (figure 6b) initially enjoys the robot following the ball, next he likes to shake robot's paw and subsequently the robot will turn to the sound source. Both users' sequences ends up with a transition to the first state and from there, the loop starts again. However, other user may not enjoy at all a particular behaviour. Therefore, this behaviour will not be rewarded and consequently it will not be part of his ideal best sequence. For example, a user may not like *turning* behaviour, so his ideal best sequence will just transit between *following* and *shaking* behaviours.

The performance of the system will be notably affected by the amount of time the user is interacting with the robot. Longer experiments will lead to more precise results because the system will gather more information to tune the Q-values. However, based on the observed results from the experiments presented in this paper, we can affirm that the robot learns



(a) User A's Q-values and his favourite sequence



(b) User B's Q-values and his favourite sequence

Fig. 6. This figure illustrates the learnt Q-values for two users and their respective favourite sequence of actions

a policy to bring users to longer interactions.

In future studies, considering a population big enough, it is possible to determine the most attractive robot's behaviour for the users in general by means of the same technology.

One question to have in mind is regarding when the user is bored of his *favourite* robot's behaviour, or when the user changes his preference, or even if the user really prefers a predictable *pet*: are you more attracted to a partner playing always your favourite game or do you prefer an un predictable partner who is able to surprise you? This is up to you.

A. Participants' reactions

Since the users were informed about what the robot should do during each behaviour, if the robot did not behave as expected, they did not reward the action. For example, the robot turns to the wrong side when the user is clapping or in another case, the robot does not find the ball when it is presented leaving the participant disappointed while cutting the session shorter.

Some users were frustrated because they expected Aibo to behave more like a lifelike dog, but it does not. Other users tried to reward the robot before it started the corresponding action because they knew what it was going to do.

Furthermore, even when users did not like robot's behaviour, it was interesting to observe how they somehow interacted with the robot but without rewarding it in the preselected way. This is interpreted as the user does not like what the robot is doing but s/he is expecting something else.

VI. Conclusion

In this paper, authors have presented how reinforcement learning can be applied to robots operating in the real world.

In a first approach, the robot learnt when to recharge its battery after the required knowledge was transferred from a simulated learning model to a real robot. Based on this, the robot is able to work for hours without running out of energy. It is interesting to highlight that the robot has learnt that when its battery level is extremely low, it will not have time enough to plug to the charger so it decides to "enjoy" the last period of its *life time*.

In a second level, Aibo has learnt to identify the user preferences. In this case learning has been accomplished in the real world through real user-robot interaction. The robot is able to learn the proper policy to engage users in longer interactions. This learning process is fully automated. This is shown by the experiments run with three different users. Besides, it is possible to identify their favourite sequence of action selection based on actions' Q-value.

VII. Acknowledgments

The authors would like to thank the funds provided by the Spanish Government through the project called "A new approach to social robotics" (AROS), of MICINN (Ministry of Science and Innovation).

References

[1] W. Smart, P. Kaelbling, *et al.*, "Effective reinforcement learning for mobile robots," in *Robotics and Automation, 2002. Proceedings. ICRA'02. IEEE International Conference on*, vol. 4. IEEE, 2002, pp. 3404–3410.

[2] M. Hausknecht and P. Stone, "Learning powerful kicks on the aibo ers-7: The quest for a striker," *RoboCup 2010: Robot Soccer World Cup XIV*, pp. 254–265, 2011.

[3] P. Fidelman and P. Stone, "The chin pinch: A case study in skill learning on a legged robot," *RoboCup 2006: Robot Soccer World Cup X*, pp. 59–71, 2009.

[4] N. Jong and P. Stone, "Hierarchical model-based reinforcement learning: R-max+ maxq," in *Proceedings of the 25th international conference on Machine learning*. ACM, 2008, pp. 432–439.

[5] N. Kohl and P. Stone, "Machine learning for fast quadrupedal locomotion," in *PROCEEDINGS OF THE NATIONAL CONFERENCE ON ARTIFICIAL INTELLIGENCE*. Menlo Park, CA; Cambridge, MA; London; AAAI Press; MIT Press; 1999, 2004, pp. 611–616.

[6] M. Saggar, T. DaSilva, N. Kohl, and P. Stone, "Autonomous learning of stable quadruped locomotion," *RoboCup 2006: Robot Soccer World Cup X*, pp. 98–109, 2009.

[7] S. Kalyanakrishnan, Y. Liu, and P. Stone, "Half field offense in robocup soccer: A multiagent reinforcement learning case study," *RoboCup 2006: Robot Soccer World Cup X*, pp. 72–85, 2009.

[8] G. Kuhlmann, P. Stone, R. Mooney, and J. Shavlik, "Guiding a reinforcement learner with natural language advice: Initial results in robocup soccer," in *The AAAI-2004 Workshop on Supervisory Control of Learning and Adaptive Systems*. Citeseer, 2004.

[9] S. Mahadevan and J. Connell, "Automatic programming of behavior-based robots using reinforcement learning," *Artificial intelligence*, vol. 55, no. 2-3, pp. 311–365, 1992.

[10] M. Ullerstam and M. Mizukawa, "Teaching robots behavior patterns by using reinforcement learning," in *SICE Annual Conference, Sapporo Japan*, 2004.

[11] A. Tapus, C. Ţăpuş, and M. Matarić, "User-robot personality matching and assistive robot behavior adaptation for post-stroke rehabilitation therapy," *Intelligent Service Robotics*, vol. 1, no. 2, pp. 169–183, 2008.

[12] R. S. Sutton and A. G. Barto, *Reinforcement Learning: An Introduction*. MIT Press, Cambridge, MA, A Bradford Book, 1998.

[13] L. P. Kaelbling, L. M. Littman, and A. W. Moore, "Reinforcement learning: A survey," *Journal of Artificial Intelligence research (JAIR)*, vol. 4, pp. 237–285, 1996.

[14] M. Malfaz and M. Salichs, "Using muds as an experimental platform for testing a decision making system for self-motivated autonomous agents," *Artificial Intelligence and Simulation of Behaviour Journal*, vol. 2(1), no. 1, pp. 21–44, 2010.

[15] C. Isbell, C. R. Shelton, M. Kearns, S. Singh, and P. Stone, "A social reinforcement learning agent," in *the fifth international conference on Autonomous agents, Montreal, Quebec, Canada*, 2001.

[16] E. Martinson, A. Stoytchev, and R. Arkin, "Robot behavioral selection using q-learning," in *of the IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS), EPFL, Switzerland*, 2002.

[17] B. Bakker, V. Zhumatiy, G. Gruener, and J. Schmidhuber, "A robot that reinforcement-learns to identify and memorize important previous observations," in *the 2003 IEEE/RSJ International Conference on Intelligent Robots and Systems, IROS2003*, 2003.

[18] C. H. C. Ribeiro, R. Pegoraro, and A. H. RealiCosta, "Experience generalization for concurrent reinforcement learners: the minimax-qs algorithm," in *AAMAS 2002*, 2002.

[19] A. Bonarini, A. Lazaric, M. Restelli, and P. Vitali, "Self-development framework for reinforcement learning agents," in *the 5th International Conference on Developmental Learning (ICDL)*, 2006.

[20] A. L. Thomaz and C. Breazeal, "Transparency and socially guided machine learning," in *the 5th International Conference on Developmental Learning (ICDL)*, 2006.

[21] C. J. Watkins, "Models of delayed reinforcement learning," Ph.D. dissertation, Cambridge University, Cambridge, UK, 1989.

[22] C. Touzet, *The Handbook of Brain Theory and Neural Networks*. MIT Press, 2003, ch. Q-learning for robots, pp. 934–937.

[23] W. D. Smart and L. P. Kaelbling, "Effective reinforcement learning for mobile robots," in *International Conference on Robotics and Automation (ICRA2002)*, 2002.

[24] M. Humphrys, "Action selection methods using reinforcement learning," Ph.D. dissertation, Trinity Hall, Cambridge, 1997.

[25] M. Fujita, "Aibo: Toward the era of digital creatures," *The International Journal of Robotics Research*, vol. 20, no. 10, pp. 781–794, 2001. [Online]. Available: http://ijr.sagepub.com/content/20/10/781.abstract

[26] M. Taylor and P. Stone, "Transfer learning for reinforcement learning domains: A survey," *The Journal of Machine Learning Research*, vol. 10, pp. 1633–1685, 2009.

[27] A. Austermann and S. Yamada, "Teaching a pet-robot to understand user feedback through interactive virtual training tasks," *Autonomous Agents and Multi-Agent Systems*, vol. 20, no. 1, pp. 85–104, 2010.

[28] L. Tóth, M. Gácsi, J. Topál, and Ádám Miklósi, "Playing styles and possible causative factors in dogs' behaviour when playing with humans," *Applied Animal Behaviour Science*, vol. 114, no. 3-4, pp. 473 – 484, 2008. [Online]. Available: http://www.sciencedirect.com/science/article/B6T48-4S7HWF2-1/2/866c25b91c29adf35c44f9d6db3c1faa