robotics

MDPI

Article

Application of Reinforcement Learning to a Robotic Drinking Assistant

Tejas Kumar Shastha *, Maria Kyrarini and Axel Gräser

Friedrich Wilhelm Bessel Institut Forschungsgesellschaft m.b.H., 28359 Bremen, Germany; maria.kyrarini@uta.edu (M.K.); graeser@iat.uni-bremen.de (A.G.)

* Correspondence: tejas@uni-bremen.de

Received: 24 October 2019; Accepted: 17 December 2019; Published: 18 December 2019

Abstract: Meal assistant robots form a very important part of the assistive robotics sector since self-feeding is a priority activity of daily living (ADL) for people suffering from physical disabilities like tetraplegia. A quick survey of the current trends in this domain reveals that, while tremendous progress has been made in the development of assistive robots for the feeding of solid foods, the task of feeding liquids from a cup remains largely underdeveloped. Therefore, this paper describes an assistive robot that focuses specifically on the feeding of liquids from a cup using tactile feedback through force sensors with direct human–robot interaction (HRI). The main focus of this paper is the application of reinforcement learning (RL) to learn what the best robotic actions are, based on the force applied by the user. A model of the application environment is developed based on the Markov decision process and a software training procedure is designed for quick development and testing. Five of the commonly used RL algorithms are investigated, with the intention of finding the best fit for training, and the system is tested in an experimental study. The preliminary results show a high degree of acceptance by the participants. Feedback from the users indicates that the assistive robot functions intuitively and effectively.

Keywords: reinforcement learning; human–robot interaction; assistive robotics; drinking assistant; human-in-the-loop control

1. Introduction

In recent years, reinforcement learning (RL) has been proposed as a solution to create individualized and customized human–robot interaction (HRI). Interactive robots that adapt their behavior to the user's needs are used in different applications such as social robots [1,2] and robot-assisted therapy [3]. However, there is limited work to address HRI applications which require direct contact. Assistive robotic manipulators have been under great interest in research, as well as the commercial market, to assist people suffering from severe motor impairments to perform activities of daily living (ADLs). The review study conducted by Naotunna et al. [4] outlines how the population of people suffering from impairments is increasing. However, the corresponding workforce for caretaking is not able to keep up. Hall et al. [5] report that there is good acceptance for assistive robotics in healthcare and ADLs. Drinking and eating assistive robotic tasks are considered highly prioritized [6] and require the robot to be in direct contact with the user. Therefore, there is a clear and present need for economical and intuitive meal assistant robots (MAR).

The focus of this paper, and indeed its novelty, is the application of RL techniques in order to satisfy the requirement of a drinking task, with direct HRI. Every human has preferences of how they would like to eat or drink and these preferences have to be respected by the assistive robot. The advantage of the approach outlined in this paper is that it is intended to be designed as a framework that respects the user's preferences.

1.1. Benefits of Proposed Solution

The application of RL was initially envisioned as a way for the users with a disability to train the robot according to their individual requirements. However, it quickly became apparent with preliminary research that physical training would be extremely taxing (requiring 1000+ user inputs) and unsafe. This problem was overcome using a software training emulator, and, coincidentally, two completely different advantages were discovered, which then validated the application of RL to this problem nevertheless:

(a) Easier modification of the system—when the system model needs to be updated, a traditional force control program would require extensive code refactoring. However, using the presented RL-based approach, wherein the model of the system and the controller are decoupled, only the model needs to be updated. An emulator trains the algorithm with the updated model in a matter of seconds, whereas the control program executing on the robot remains completely unchanged. This provides a huge advantage in terms of software development and testing cycles.

(b) Deep reinforcement learning—this technique allows for even further flexibility since it can potentially handle models with a very large number of states. The level of abstraction provided in formulating the model of the environment enables this, and the usage of novel software training techniques means that newer versions can be rapidly developed. This becomes invaluable for the next planned stages, where more redundant safety sensors will be included, extending the state space exponentially. Using a simple "if–else" based programmatic state machine becomes too complex to handle thereafter.

An assistive robotic arm is used as the robotic platform and a cup equipped with a force sensor enables safe HRI. The main goal of this paper is to enable the user to easily and effectively drink some liquid from the cup gripped by the robot, by interacting with it using their lips only. The precise contributions over existing literature are discussed in the next chapter.

The paper is organized as follows: Section 2 provides a quick survey of the state of the art and the motivation for this undertaking, describing why the solution presented here is unique and relevant. Section 3 describes the assistant system, and Section 4 outlines the application of RL as a solution for this assistant system. Section 5 describes the experimental study conducted to test the developed solution. Finally, Section 6 concludes the paper and pontificates further development.

2. Related Work and Motivation

The motivation for developing the robotic solution presented in this paper can be clearly understood once the existing trends in meal assistive robots are investigated. Due to the high impact of delivering solutions for assisted eating and drinking, several research projects, as well as commercial products, have been developed. However, all of these solutions focus specifically on the task of eating and relegate the task of drinking as a secondary function. A few such existing solutions are next considered, and their shortcomings with respect to drinking are discussed.

2.1. State-Of-The-Art

Neater Eater [7] is a simple but economical product that provides assistance with only the eating task but at a very affordable price. My Spoon [8] is a collaboration project that utilizes electromyography (EMG) signals from calves as click events, thereby providing more degrees of freedom with meal assistance. The Meal Support System [9] uses a laser rangefinder to detect the presence and formation of food on a tray, allowing the system to handle a wider variety of dishes than other robots. However, the common shortcoming in these systems is that they all focus on eating exclusively and do not provide any support for drinking at all.

Another class of MARs is those that focus on the eating task, but do provide some small provision for drinking, but usually in a very limited capacity. Handy 1 developed in [10,11] is a classic example of this case. It was one of the first MARs developed, and later vastly improved, but for the drinking task

only provides a spring-loaded cup with a straw. The assistive robot developed by Yamamoto et al. [12] was among the first for such novel applications as computer vision and speech synthesis. ICRAFT developed by Lopes et al. [13] incorporates eye tracking with a Creative Light Infra-Red LED and camera to select the desired dish in the Northeastern University, Boston, USA. Mealtime Partner by the company Mealtime Partner Inc., Texas, USA [14] is a successful commercial product in the market today. In all of these cases, a cup with a straw is simply held in place with a rigid brace.

The final group of MARs to consider actually comes closer to solving the drinking task, but still lacks in certain aspects. Schröer et al. [15] have developed the most state of the art system with respect to feeding liquids. It focuses mainly on a brain–machine interface (BMI) for the user input of Go/No-Go signals and image processing techniques to find the position of the mouth under partial occlusion. KARES II developed by Bien et al. [16] is a wheelchair mounted manipulator designed for 12 tasks including feeding liquid. It focuses on multimodal communication and EMG for stop signals. The drinking tasks in both of these make use of robots manipulating cups directly and without straws. However, they follow a pre-set trajectory once initiated and the user only has the control to start or stop.

2.2. Motivation and Unique Contribution

It can be observed from the review of the state of the art that MARs, either as commercial products or research projects, are all focused on the feeding of solid food. The feeding of liquids is largely relegated to a simple straw. In some cases, cups are used, but the trajectories are pre-programmed. While this type of solution is cheap, it is not completely intuitive and it can be improved.

With this motivation in mind, the robotic drinking assistant that shall be presented in this paper was developed. The unique contributions thereof are summarized as follows:

1. Human in the (open) loop force control—a robotic assistant with *direct* HRI, where the human and robot influence each other, will be presented. It provides a more intuitive and natural feeling drinking experience to the user, akin to traditionally drinking from a cup, while keeping the cost low. The user has a greater degree of control over the robot, and moves the cup as preferred, which is an improvement over fixed trajectories or straws which are seen in the current literature. The previous related works as presented above do not have the human in the control loop and instead use human input as triggers to only start/stop preprogrammed robot movements that the user cannot influence (i.e., indirect HRI).
2. User customization—'force profiles' of each user are tweaked until satisfactory and then manually stored for future use, based on trial and error to find the best fit, which allows for the personalization of the MAR.
3. RL in meal assistance robots—this work is the first to apply RL to a MAR, allowing for greater flexibility in rapid prototyping and further developments, as outlined in the previous chapter.
4. Direct comparison of RL algorithms—this system proved to be a good testbed for comparing the training of the five different algorithms (not performance), which provides valuable insights (shown in Section 5.1).

The solution presented in this paper builds upon the work presented in a previous paper by Goldau, Shastha et al. [17] that shows the task of the robot approaching the face of the user based on vision and getting a feeding cup with spout close to the user's mouth. The preliminary drinking task was implemented using two force sensors, mounted on the spout of the feeding cup, with only open-loop force control [17]. The initial experimental results from this previous work provided good feedback from users (especially the person with tetraplegia). However, the user with tetraplegia would prefer to use a normal cup and better control over the drinking process, and this feedback guided most of the development presented in this paper.

3. Robotic Drinking Assistant

The robot in use is a seven degrees of freedom Kinova Jaco manipulator arm with a three-finger gripper [18] to grasp the *smart cup*, which is a standard plastic cup fitted with a FlexiForce A201 force sensor [19] on the outer rim, and a Bluetooth module fixed underneath it to transmit the readings of the force sensor (Figure 1a). It was previously developed by Koller et al. [20] and it was adapted for use in this paper by switching to a normal cup instead of a feeding cup and using a single force sensor instead of two. This change was made based on the feedback of the user with tetraplegia, as she felt the normal cup felt more natural. The removal of one sensor also reduces the cost. The movement of the cup is along the Sagittal plane of the user [21], which is effectively the manipulation of purely the Euler Roll angle of the end effector (see Figure 1b).

(**a**) (**b**)

Figure 1. (**a**) Setup of the Robotic Drinking Assistant. (**b**) Euler angles of the robot.

The rotation of the cup is controlled by how much force the user applies with their lips on to the force sensor. The total range of forces is divided into three main regions. If the user applies 0 to 0.1 N force, this causes the cup to lower itself. If the user applies force between 0.1 N and 0.5 N, the cup is held still by the robot. If the user applies force between 0.5 N and 5.0 N, the cup is raised upwards. 5.0 N is the safety limit, and if exceeded, it causes the robot to fall back away from the user and ends the drinking task. The absolute force values detailed here were determined after some initial testing in the lab, and the empirical test results from [17]. However, they are not fixed and can be, and indeed have been, changed to match user comfort.

This control strategy of controlling the cup was developed based on preliminary tests conducted in the lab as well as with one test with the user with tetraplegia, according to what felt most intuitive and technically feasible considering practical constraints. In the previous version of the smart cup (described in [17]), two sensors were available and, therefore, both the direction and magnitude of the forces applied by the subject could be accounted for. The change to the simple cup, resulting from the feedback of the user with tetraplegia, forced the removal of one of the sensors due to physical constraints. Due to this, the above-mentioned control strategy was developed where only the magnitude could be considered.

4. Reinforcement Learning Solution for Robotic Drinking Assistance

To implement the above-mentioned control strategy using RL on the robotic arm, the so-called Policy-Driven Controller (Figure 2) was developed. This is a control node that runs on the Jaco arm, and works with the smart cup, in order to realize the drinking assistant.

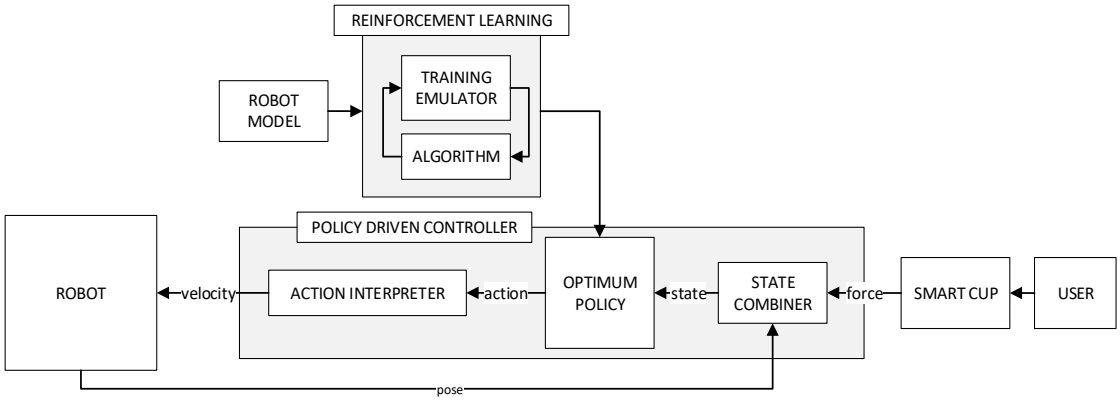


Figure 2. Policy-Driven Controller (PDC).

In the first phase, the RL algorithms are trained with the emulator and an optimum policy is generated, which is then plugged into the actual controller. In the second phase, the controller is executed in the live environment. The state combiner is responsible for encoding the state of the model from the system variables. The optimum policy is then referred to, in order to select the best action for this state. The action interpreter then applies this action through meaningful velocity commands to the robotic arm. The formulation of the Markov model, the selection of RL algorithms, and their training will presently be discussed.

4.1. Markov Decision Process of Reinforcement Learning

The model of the environment created in order to be used with the Markov decision process is referred to as a *Markov model* in this paper. The five main factors to be defined for such a model are States, Actions, State transition probability distribution, Rewards function, and Discount factor [22,23]. These shall now be explained in the context of this paper:

States: The two variables of the environment that affect the decision-making process are the current pose of the robot, and the force input from the user. These two *sub-states* shall combine together to form the states of the model.

The first sub-state is the pose of the robot arm. Considering the maximum upper and lower limits of the roll angle for the end effector due to practical constraints, the complete space of end effector roll angle manipulation is divided into *na* total number of equal parts i.e., the arm *sub-states*. (*na* was empirically set to 10 for best results). The size of each part (θ) can easily be expressed as follows

$$\theta = \frac{Max\ upper\ angle - Max\ lower\ angle}{na}.$$

Thus, θ also defines the magnitude of rotation of the end effector roll angle per step. The arm sub-states are defined in this manner for two reasons. First, the absolute values of roll angles cannot be directly used since this would cause a state explosion, as well as suffer from drift and inaccuracies of the robot. As long as *na* is small enough, θ encapsulates these errors. Second, the *Max lower angle* limit varies for each execution, depending on the starting position of the user, causing inconsistencies in the definition of each individual sub-state. Defining a dynamic roll space that is always divided into equal parts solves this problem. The values that the arm sub-state can take are in the limits [0,9].

The second sub-state is the force input from the user. The entire region of force values is once again divided into three parts due to the same aforementioned reasons. These parts are directly derived from the control strategy i.e., three parts divided for the purpose of raising cup, holding cup, and lowering cup. The values that the force sub-state can take are in the limits [0,2].

As a result of these definitions, the total number of states nS, which is simply the combination of both sub-states is easily shown as

$$nS = 3 \times n$$

where 3 and na are the number of force and arm sub-states respectively. Since na is selected as 10, the model has 30 states.

The final Markov state is simply a composition of the arm sub-state and force sub-state. This can be done in any number of ways as desired, but the following empirical formula was defined for this implementation:

$$sm := (3 \times sa) + s$$

where, sm = Markov state value, sa = arm sub-state value, sf = force sub-state value. The values of Markov states derived per this definition are in the limits [0,29]. To understand the virtue of this formulation with an example—the arm and force sub-states of 3 (sa) and 4 (sf) respectively compose into the Markov state (sm) of 13 whereas the sub-states 4 (sa) and 3 (sf) respectively compose into the Markov state (sm) 15.

Actions: The three actions defined for this model were to rotate the cup, or rotate it down or do nothing (see Table 1). These three actions provide complete control to the user to drink from the cup. The additional movement of fallback for safety is considered a special case and not modeled as an action.

Table 1. Markov model actions.

Action	Definition	Name
1	The cup shall rotate up	ACTION_DOWN
2	Do nothing	ACTION_STAY
3	The cup shall rotate down	ACTION_UP

Transition Probability Distribution: For a given arm sub-state, an action can either transition it one step higher/lower in the arm sub-states table or make no change. But for a given action, it is a deterministic change. The user's force input sub-state on the other hand, is a random transition since the user can choose to give any input desired. It is, however, still limited in possible values since there are only three force regions. Therefore, it is random with discrete values. As a result, the total transition of the Markov states (which is a combination of the two sub-states) is random with a finite set of possible next states. Assuming the user is equally inclined to choose any of the possible inputs, the transition probability to each possible next state is equal.

An example from this distribution is: $P(3|2) = [(0.33,\ 6),\ (0.33,\ 7),\ (0.33,\ 8)]$. If action 3 (raise cup) is taken on state 3 (arm sub-state = 1, force sub-state = 0), the possible next states are 6, 7, 8 (for arm sub-state = 2, and 3 possible force sub-states), each with equal probability.

Reward Function: The reward function is highly shaped. Assignment of rewards to a state ×action pair is based on the force sub-state as this signifies the relationship between the intent of the user and the response of the agent. A positive reward is provided to the action that follows the user intent, for example, if the user provides force input to raise the cup and the action selected is to raise the cup. Other cases receive a negative reward. The border cases are special, in that if the robot is already at a maximum angle limit, but the user intends to exceed the limit, the hold cup action still needs to be incentivized regardless of user input. To ensure this, a positive reward is assigned to the hold cup action to make sure that the cup does not actually exceed the safety angle limits.

Discount Factor: It is selected to be low (0.1) in order to prioritize immediate rewards. This is very important because the agent has to select the action that gives the best reward right away, as opposed to trying to accumulate a greater reward over time and choosing undesirable actions.

A Markov model can be designed in order to train an RL algorithm that either works perpetually or for a limited period of time/trials. To experiment with both approaches, two special cases of the

model were developed: terminating and non-terminating. As the names suggest, the terminating model takes the three final states (i.e., max upper angle of rotation) as the terminating states and the non-terminating model has no such definition. Therefore, the terminating model describes a time-limited model in that the target is for the user to reach a terminating state to finish drinking from the cup within a finite amount of time. This is easier to comprehend and hence better during development. The non-terminating model is perpetual because the user may not always want to completely finish drinking from the cup, and the agent needs to function for an infinite amount of time. This is closer to practical requirements and is more desirable for the final solution.

With the full formal definition of each element of the 5-tuple, a complete Markov model is prepared for the robotic drinking assistant environment, and is summarized in Figure 3.

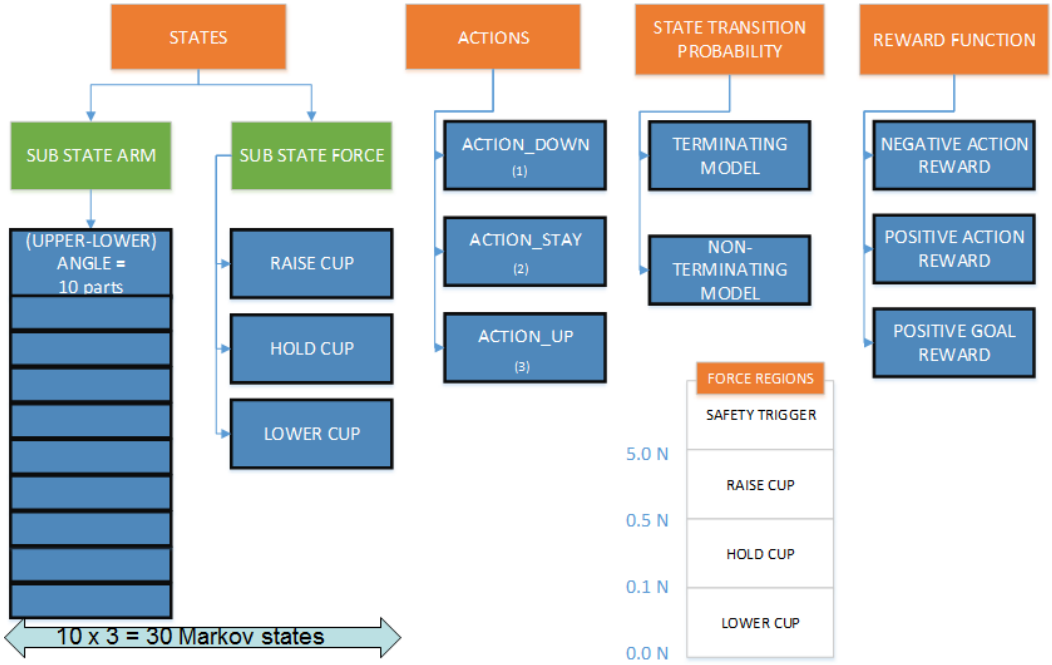


Figure 3. Complete Markov model.

4.2. Training Setup

Training of any reinforcement learning algorithm requires a large number of trials. For robotics, this can be an expensive process. In the case of HRI, this cost is even more pronounced because of not only effort by the user, but safety as well, as the robot is prone to making many "bad" actions. For this reason, training via a software emulator was designed. The emulator is simply a piece of code that generates equally distributed random user inputs for the force and tracks the pose of the robot to maintain the internal state. The emulator generates the complete range of forces from the user, testing and exploring the full state space. It makes use of the complete Markov model in order to do this. In this manner, training is done completely offline and within a fraction of the time, typically a few minutes. The results generated were tested with the Policy-Driven Controller in a live environment found to be perfectly applicable, affirming the fact that the model and the emulator are accurately derived from the real environment. The emulator also keeps track of training metrics such as episodes to convergence, rewards per episode, the max variance of converging variables, etc. The emulator is intended to be used as a development tool at first, to be eventually replaced with live training once the best algorithm and hyper-parameters are evaluated as well as more safety systems are emplaced. This paper presents the preliminary results of emulator training.

4.3. Reinforcement Learning Algorithms

Five reinforcement learning algorithms were trained with the previously described environment. In order to explain them, the phrases "online" and "offline" used hereafter refer to the necessity of the live environment/emulator for training online, and that fact that a mathematical Markov model is sufficient for offline training. The algorithms applied are all iterative, and they are stopped once *convergence* has been achieved. Convergence is said to occur based on slightly different factors for each algorithm, but generally deals with minimizing a so-called *converging factor*, which is the primary variable being iteratively calculated by the algorithm. This method shall be referred to as *variation minimization* in this paper. These algorithms are briefly explained as follows:

- Value iteration (VI) and policy Iteration (PI) [24]. The approach of VI is to optimize a so-called value function over many iterations until convergence, and extracting the optimum policy based on it. PI functions in a similar manner by computing the value function but optimizes the policy directly. Since they are both offline, the training emulator was not used, and the training process is drastically simpler and faster compared to the other algorithms. The resulting optimum policy obtained served as not only a benchmark for further algorithms but also a reference for the rapid prototyping of the PDC for live testing. The value function is iteratively calculated in both cases, and convergence is said to occur when it drops to 0.00001. Convergence was achieved consistently after five episodes for non-terminating and six episodes for terminating models respectively for PI, and two episodes for both models for VI.
- Q learning and state–action–reward–state–action (SARSA) [25,26]. These two algorithms are similar since they both compute and optimize the Q table as the convergence factor. The difference is that the SARSA algorithm takes into account one more state action pair than Q learning (QL). These two are of greater interest since they are online and avoid the need for a full model, while at the same time being simple/fast enough to compute solutions efficiently. Being model-free also helps with rapid prototyping and development whenever the environment is modified. The learning rate was selected to be 0.8, the epsilon greedy approach to exploration vs. exploitation was used with a decay rate of 0.1 and the convergence threshold was set to 0.001.
- Deep Q network (DQN) [27–29]. This final deep reinforcement learning class of algorithms combines the advantages of deep learning as well as RL. Whereas the Q table is still computed, it is used to train a neural network instead of saving as a lookup table. This allows for even more number of states to be modeled (which could be useful to capture data from the environment at a greater resolution), at a reduced cost of training time. The learning rate and epsilon factor were the same as before. The neural network used had four fully connected regular layers of 24 neurons with *relu* activation. There were 30 inputs (states) and 3 outputs (actions). Loss function was mean square error which serves as the convergence factor, the threshold of which was 0.0001.

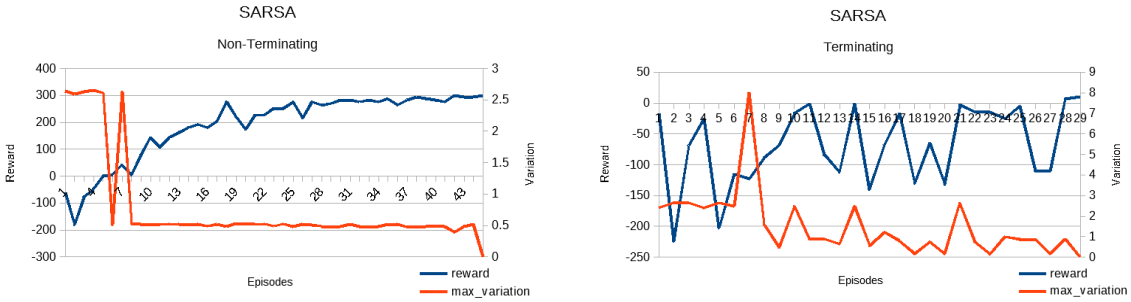
These five pre-existing algorithms were chosen, as they represent different classes of RL algorithms that are readily available. PI and VI belong to the dynamic programming class, which are offline and faster, providing a good benchmark. However, both require a complete model. Q-Learning and SARSA are model-free approaches, providing more flexibility. On the other hand, both require online training. DQN allows for training of systems with a very large number of states, at the cost of more training time. This was considered a good platform for comparing the training process of these algorithms. The resulting optimum policy was expected, and later experimentally verified, to be the same for all five algorithms, since the same Markov model was considered.

5. Training, Experiment Study and Results

5.1. Training the Algorithms

The threshold values for convergence are experimentally determined, based on a number of test training runs until consistent results were obtained. Variation minimization is selected over rewards

maximization because the latter requires a larger number of training episodes, which is undesirable in the case of training an environment with HRI. Also, variation minimization is more reliable to compare between the terminating and non-terminating models based on observed results. This is substantiated in Figure 4a,b, where the SARSA training rewards profile varies largely between the two model variants, but the variation profile converges similarly.



(**a**) (**b**)

Figure 4. (**a**) State–action–reward–state–action (SARSA)—non-terminating model; (**b**) SARSA—terminating model.

The training of the five algorithms was carefully monitored over six training runs each and the comparison of their performances is shown in Table 2. PI and VI, being offline approaches, were trained with the full model. The other 3 algorithms, being online, trained with the emulator. The protocol was, the hyper-parameters (including convergence threshold) were tuned until the best results (optimum policy vs. training time) were obtained for each of them.

Table 2. Comparison of training of reinforcement learning algorithms (RLA).

Name of RLA	Convergence Factor	Convergence Threshold	Average No. of Episodes to Convergence (Non-Terminating Model)	Average No. of Episodes to Convergence (Terminating Model)
PI	Value function	0.00001	5	6
VI	Value function	0.00001	2	2
QL	Q Table	0.001	**12.8**	40.2
SARSA	Q Table	0.001	49.2	31.1
DQN	Loss function	0.0001	49.7	**31**

As can be seen from Table 2, the dynamic programming algorithms are significantly faster, which is to be expected. They are, however, not preferable to take forward since they are model-based and future development might not always be able to be modeled perfectly. They serve therefore as benchmarks to compare the other algorithms. Between QL and SARSA, we see that QL is faster for the non-terminating model and SARSA is better for the terminating model. Overall, QL was found to be fastest when training the non-terminating model and DQN the fastest for the terminating model. Since the non-terminating model is preferable, QL was selected to be the overall best algorithm to take forward into the feasibility study. Surprisingly, DQN trained within a time period comparable to SARSA.

5.2. Experiment

An experimental study was conducted, to test the feasibility of the robotic drinking assistant developed with reinforcement learning. The test involved sixteen able-bodied subjects and one tetraplegic user. Of them, seven were male and nine were female (one of whom was the user with tetraplegia). The average age was 25.0 ± 2.8. All participants gave their informed, signed consent to participate in this study. This study is part of the German national funded project MobiLe and the

ethical contract for the studies within the project has been approved by the German Society of Social Work (in German: DGSA-Deutsche Gesellschaft für Soziale Arbeit). The aim of the experiments was to test two types of controllers—first a simple force controller with a hardcoded state machine ("if–else" table, same user control schema) and second, the RL based PDC. This was done in two phases for each-first without any liquid in the cup to get acclimatized with the control strategy, and then to try to actually drink some water from the cup. The users chose to repeat the dry runs without water as many times as they felt they needed to feel comfortable with the robot. They then repeated the experiment with water until they were able to successfully drink water without spilling. This was done for both controllers. The operator stood by for safety reasons (to handle emergency situations) but did not intervene otherwise. At the end, the users were provided with a feedback form consisting of some questions, the answers to be graded on a 5-point Likert scale with '1' being strongly disagreed and '5' being strongly agreed. These questions were generated based on the learning from the previous work [17], derived in principle from the system usability scale [30], to collect the information we felt was most necessary to gauge the performance of the drinking assistant.

5.3. Results

Once the study was concluded, the resulting feedback from the users was compiled and is shown in Table 3:

Table 3. Subject responses from the feedback forms. The scale is 1–5, 5 is best.

No.	Category	Questions	Average Response	Standard Deviation
1	Introduction	I could understand the instructions clearly	4.75	0.56
2		I felt comfortable during the experiments	4.69	0.46
3		I felt safe during the experiments	4.81	0.53
4	Force Control	The overall schema was intuitive	4.63	0.6
5		The thresholds for force regions were convenient	4.19	0.81
6		The robot behaved as I expected (No surprises)	4.44	0.86
7		The speed of operation was appropriate	4.56	0.7
8		I could comfortably drink water	4.31	0.98
9	Policy Driven Control	The overall schema was intuitive	4.69	0.58
10		The thresholds for force regions were convenient	4.5	0.71
11		The robot behaved as I expected (No surprises)	4.56	0.61
12		The speed of the cup per step was appropriate	4.44	0.61
13		The number of steps to completely drink water was appropriate	4.5	0.79
14		I could comfortably drink water	4.44	0.79
15	General Control Strategy	The path taken by the cup from start to finish was comfortable	4.69	0.46
16		The movement of the robot was aggressive/disturbing	1.5	1.06
17		I would use this system if I had need of it	4.19	1.13

5.4. Discussion

Table 3, category 'Introduction', shows users were comfortable performing experiments, Table 3, categories Force Controller and Policy Driven Controller, show the users felt that the PDC functioned slightly better than Force Controller. This is because the PDC was initially modeled to match the baseline performance of the Force Controller, but as the experiments proceeded over the course of 3 days, the parameters of the model were slowly improved in small increments based on feedback from the subjects. This improved the state and action encoding for better user comfort, such as tweaking the force regions explained in Section 3, or the step angle θ as explained in Section 4.1. The newer versions of the model could be very quickly trained with the emulator and immediately used by the controller.

By the end of the 3rd day, the PDC was performing well above the baseline. This shows the advantage of rapid prototyping with the RL based approach. Table 3, category General Control Strategy, shows that the subjects have provided generally accepting feedback. This provides a good basic idea that the controller designed, the force thresholds defined (in some cases tweaked to user comfort) and the optimum policy selected are all intuitive and comfortable to the subjects.

Figures 5 and 6 show the pattern in which the experimental subject 12 (chosen at random for presentation here) exerted forces on the cup for the Force Controller and the PDC, respectively. At this point, the PDC was already more refined compared to the Force Controller. An increment in the roll angle (blue line) corresponds to the robot lowering the cup (i.e., delivering water to the user) and vice versa. When the force (red line) crosses the upper threshold (green line), the robot lowers cup and when the force falls below the lower threshold (yellow line), the robot raises the cup. When force is within the two thresholds, the cup is still (blue line is flat) and the user sips water. Figure 6 is more erratic in blue and red lines, because this is the full run with water, and the user moved the cup up and down several times in order to get smaller sips. But the general upward trend in the blue line shows how the user gradually lowers the cup more and more, to drink all the water. The final drop in the blue line is when the user has finished drinking, and the robot moves back. It can be observed that the profile in the Force Controller is more erratic than the PDC, with several small movements and repetitions of some movements. This is once again because several minor improvements (step angle, number of steps, initial pose, etc.) could be rapidly incorporated into the PDC as opposed to the Force Controller. Figure 7 shows one trial with subject 14, when the user applied too much force on the cup, triggering the safety fallback procedure. The blue line falls slightly low and stays flat while the cup is slightly raised and then pulled back. This demonstrates how the system is safe from large contact forces.

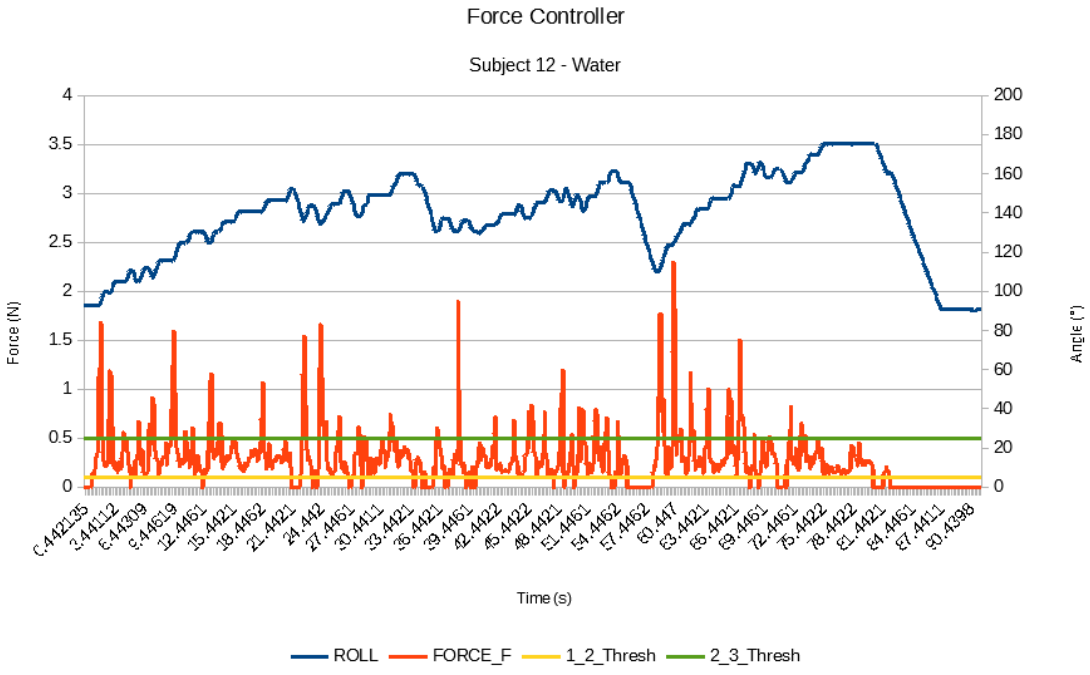


Figure 5. Forces exerted during the trial with water for Force Controller. Blue line is the roll angle of the robot (i.e., the orientation of cup), the green line is upper trigger threshold, the yellow line is lower trigger threshold, and the red line is the force exerted by subject. Y-axis is force and X-axis is time.

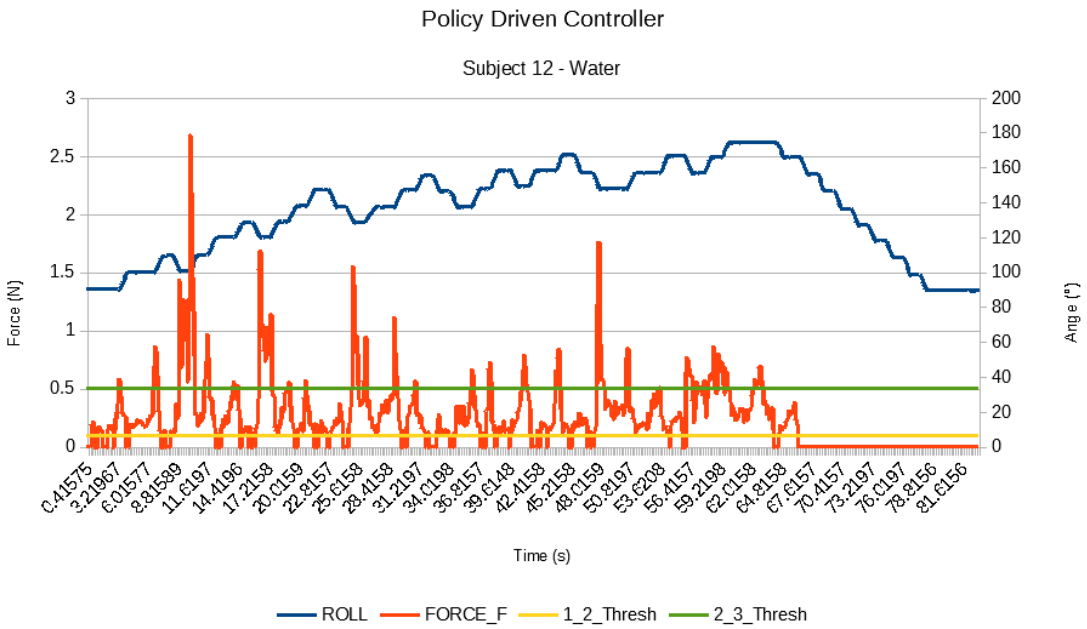


Figure 6. Forces exerted during the trial with water for PDC. Blue line is the roll angle of the robot (i.e., the orientation of cup), the green line is upper trigger threshold, the yellow line is lower trigger threshold, and the red line is the force exerted by subject. Y-axis is force and X-axis is time.

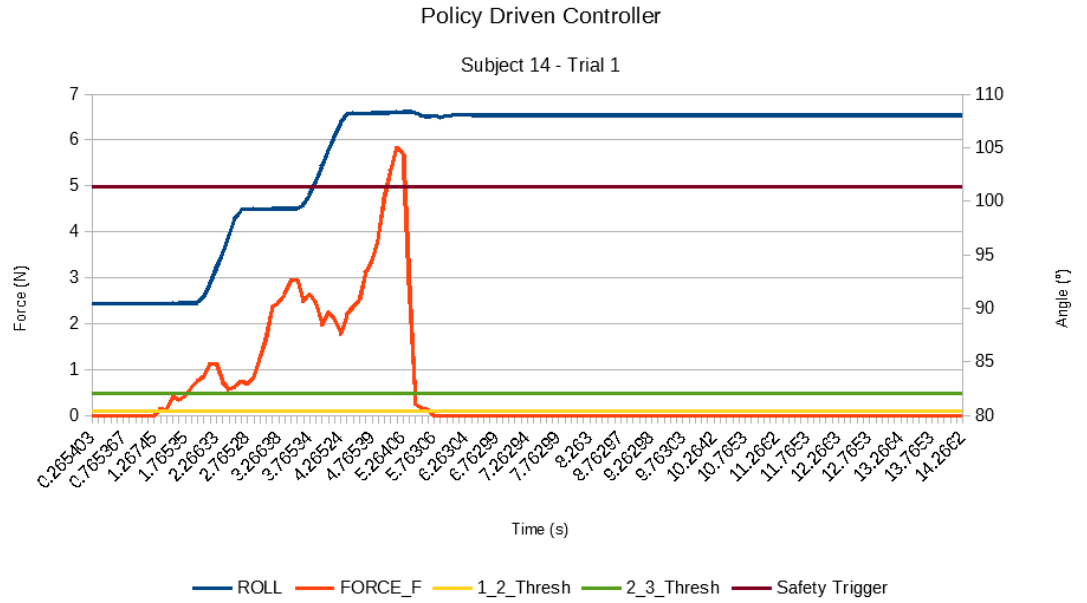


Figure 7. Force exerted during a safety trigger. The purple line is the safety threshold. Blue line is the roll angle of the robot (i.e., the orientation of cup), the green line is upper trigger threshold, the yellow line is lower trigger threshold, and the red line is the force exerted by the user. Y-axis is force and X-axis is time.

5.5. Feedback from the User with Tetraplegia

The user with tetraplegia required eight trials to succeed, but still felt the system was very intuitive and liked it overall. The force trigger values had to be modified quite a bit before she was comfortable, but after saving this profile she was able to use it easily. This demonstrates the flexibility of the system per user. Since this user had also participated in the previous study presented in [19], she was able to make a comparison and agreed that this newer version, specifically the PDC, was a large improvement,

especially when compared to drinking from a feeding cup with a spout. She also provided feedback that the robot approaching from the front was a bit stressful and would prefer that the robot approaches from the side instead. The other subjects had small comments to make such as modifying the initial pose, water dripping at times, tweaking the force regions, etc. Nevertheless, overall, the general acceptance of this robotic drinking assistant is seen to be high.

6. Conclusions and Future Work

Reinforcement learning has been effectively applied to the task of a robotic drinking assistant so as to allow persons suffering from tetraplegia to be able to drink some liquid directly from a cup held by the robotic manipulator arm. To solve this task, five different algorithms are applied. The training was carried out effectively with a software emulator program and the comparison of the training metrics revealed a clear choice of Q Learning for the model used. The solution developed is simple, avoids straws, and provides the user with the ability to freely manipulate the cup. Safety limits are placed on the force-based control in order to avoid any harm to the user.

A preliminary feasibility test was carried out with some able-bodied subjects and one user with tetraplegia. The feedback from these subjects showed that the system developed is indeed intuitive and effective. The user with tetraplegia was able to drink some water from the cup and found the control scheme to be convenient.

Whereas the ideal case would be to test this system with more persons with disabilities, due to lack of availability of such a test group, the experiment was carried out with able-bodied people and one user with tetraplegia. The results are still indicative of the good performance of the system, and effectively recognize the areas of future improvement. The precise benefits and contributions of the proposed solution were explained in Sections 1.1 and 2.2, and they form a good basis for future work.

Future work will consider the usage of shear force sensors that will allow the user to not only control the up-and-down rotation of the cup, but also forward-and-backward translation. This will provide even more control to the user. Additional redundant safety sensors will be included, and additional configurations of the robot will also be investigated. The DQN solution proposed here holds an immense advantage for this purpose.

Author Contributions: Conceptualization, T.K.S. and M.K.; methodology, T.K.S.; software, T.K.S.; validation, T.K.S. and M.K.; formal analysis, investigation, resources, T.K.S.; writing—original draft preparation, T.K.S.; writing—review and editing, M.K. and A.G.; supervision, M.K. and A.G.; project administration, A.G.; funding acquisition, A.G. All authors have read and agreed to the published version of the manuscript.

Funding: The research was supported by the German Federal Ministry of Education and Research (BMBF) as part of the project MobiLe (Physical Human–Robot-Interaction for Independent Living), having the grant number 16SV7867.

Acknowledgments: The authors would like to thank all the persons involved, in particular, the user with tetraplegia, Lena Kredel, who took part in the experimental study.

Conflicts of Interest: The authors declare no conflict of interest.

References

1. Ritschel, H.; Seiderer, A.; Janowksi, K.; Wagner, S.; Andre, E. Adaptive linguistic style for an assistive robotic health companion based on explicit human feedback. In Proceedings of the 12th ACM International Conference on Pervasive Technologies Related to Assistive Environments, Rhodes, Greece, 5–7 June 2019; pp. 247–255.
2. Ritschel, H. Socially-aware reinforcement learning for personalized human-robot interaction. In Proceedings of the 17th International Conference on Autonomous Agents and MultiAgent Systems, Stockholm, Sweden, 10–15 July 2018; pp. 1775–1777.
3. Tsiakas, K.; Dagioglou, M.; Karkaletsis, V.; Makedon, F. Adaptive robot assisted therapy using interactive reinforcement learning. In Proceedings of the International Conference on Social Robotics, Kansas City, MO, USA, 1–3 November 2016; pp. 11–21.

4. Naotunna, I.; Perera, C.J.; Sandaruwan, C.; Gopura, R.A.R.C.; Lalitharatne, T.D. Meal assistance robots: A review on current status, challenges and future directions. In Proceedings of the 2015 IEEE/SICE International Symposium on System Integration (SII), Nagoya, Japan, 12–13 December 2015; pp. 211–216.
5. Hall, A.K.; Backonja, U.; Painter, I.; Cakmak, M.; Sung, M.; Lau, T.; Thompson, H.J.; Demiris, G. Acceptance and perceived usefulness of robots to assist with activities of daily living and healthcare tasks. *Assist. Technol.* **2019**, *31*, 133–140. [CrossRef] [PubMed]
6. Chung, C.S.; Wang, H.; Cooper, R.A. Functional assessment and performance evaluation for assistive robotic manipulators: Literature review. *J. Spinal Cord Med.* **2013**, *36*, 273–289. [CrossRef] [PubMed]
7. Neater Solutions Ltd. 2017 Brochure. Available online: http://www.neater.co.uk/neater-eater/ (accessed on 8 March 2019).
8. Zhang, X.; Wang, X.; Wang, B.; Sugi, T. Real-time control strategy for EMG-drive meal assistance robot—My Spoon. In Proceedings of the 2008 International Conference on Control, Automation and Systems, Seoul, Korea, 14–17 April 2008; pp. 800–803.
9. Ohshima, Y.; Kobayashi, Y.; Kaneko, T. Meal support system with spoon using laser range finder and manipulator. In Proceedings of the 2013 IEEE Workshop on Robot Vision (WORV), Clearwater Beach, FL, USA, 15–17 January 2013; pp. 82–87.
10. Topping, M.J.; Smith, J.K. The development of handy 1. A robotic system to assist the severely disabled. *Technol. Disabil.* **1999**, *10*, 95–105. [CrossRef]
11. Bühler, C.; Heck, H.; Topping, M.; Smith, J. Practical experiences using the 'Handy 1'assistive robot far various ADL tasks. In Proceedings of the 5th European Conference for the Advancement of Assistive Technology AAATE 99, Düsseldorf, Germany, 1–4 November 1999.
12. Yamamoto, M.; Sakai, Y.; Funakoshi, Y.; Ishimatsu, T. Assistive robot hand for the disabled. In Proceedings of the International Conference on Systems, Man, and Cybernetics (Cat. No. 99CH37028), Tokyo, Japan, 12–15 October 1999; Volume 1, pp. 131–134.
13. Lopes, P.; Lavoie, R.; Faldu, R.; Aquino, N.; Barron, J.; Kante, M.; Magfory, B. Icraft-Eye-Controlled Robotic Feeding Arm Technology. Available online: https://ece.northeastern.edu/personal/meleis/icraft.pdf (accessed on 1 September 2018).
14. Mealtime Partner Inc. Hands-Free eating and Drinking Products for Individuals with Disabilities. Available online: http://www.mealtimepartners.com/ (accessed on 8 March 2019).
15. Schröer, S.; Killmann, I.; Frank, B.; Voelker, M.; Fiederer, L.; Ball, T.; Burgard, W. An autonomous robotic assistant for drinking. In Proceedings of the International Conference on Robotics and Automation (ICRA), Seattle, WA, USA, 25–30 May 2015; pp. 6482–6487.
16. Bien, Z.; Chung, M.J.; Chang, P.H.; Kwon, D.S.; Kim, D.J.; Han, J.S.; Kim, J.H.; Kim, D.H.; Park, H.S.; Kang, S.H.; et al. Integration of a rehabilitation robotic system (KARES II) with human-friendly man-machine interaction units. *Auton. Robot.* **2004**, *16*, 165–191. [CrossRef]
17. Goldau, F.F.; Shastha, T.K.; Kyrarini, M.; Gräser, A. Autonomous multi-sensory robotic assistant for a drinking task. In Proceedings of the 2019 IEEE 16th International Conference on Rehabilitation Robotics (ICORR), Toronto, ON, Canada, 24–28 June 2019; pp. 210–216.
18. Kinova. Jaco User Guide. Available online: https://www.robotshop.com/media/files/PDF/jaco-arm-user-guide-jaco-academique.pdf (accessed on 25 February 2019).
19. Tekscan, Flexiforce Datasheet. Available online: https://www.tekscan.com/products-solutions/electronics/flexiforce-quickstart-board (accessed on 1 September 2018).
20. Koller, T.L.; Kyrarini, M.; Gräser, A. Towards robotic drinking assistance: Low cost multi-sensor system to limit forces in human-robot-interaction. In Proceedings of the 12th ACM International Conference on PErvasive Technologies Related to Assistive Environments, Rhodes, Greece, 5–7 June 2019; pp. 243–246.
21. National Cancer Institute, Anatomical Terminology. Available online: https://training.seer.cancer.gov/anatomy/body/terminology.html (accessed on 26 February 2019).
22. Sutton, R.S.; Barto, A.G. *Reinforcement Learning: An Introduction*; MIT Press: Cambridge, MA, USA, 2018.
23. Alpaydin, E. *Introduction to Machine Learning*; MIT Press: Cambridge, MA, USA, 2014.
24. Bertsekas, D.P. *Dynamic Programming and Optimal Control*, 3rd ed.; MIT Press: Cambridge, MA, USA, 2011.
25. Rummery, G.A.; Niranjan, M. *On-Line Q-Learning Using Connectionist Systems*; University of Cambridge, Department of Engineering: Cambridge, UK, 1994; Volume 37.
26. Watkins, C.J.; Dayan, P. Q-learning. *Mach. Learn.* **1992**, *8*, 279–292. [CrossRef]

27. Keras Documentation, Layers, Core Layers. Available online: https://keras.io/layers/core/ (accessed on 3 March 2019).
28. Schmidhuber, J. Deep learning in neural networks: An overview. *Neural Netw.* **2015**, *61*, 85–117. [CrossRef] [PubMed]
29. Arulkumaran, K.; Deisenroth, M.P.; Brundage, M.; Bharath, A.A. Deep reinforcement learning: A brief survey. *IEEE Signal. Process. Mag.* **2017**, *34*, 26–38. [CrossRef]
30. Brooke, J. System Usability Scale (SUS). Available online: https://www.usability.gov/how-to-and-tools/methods/system-usability-scale.html (accessed on 15 November 2019).