Human Interaction for Effective Reinforcement Learning

L. Adrián León, Ana C. Tenorio, Eduardo F. Morales

Instituto Nacional de Astrofísica, Óptica y Electrónica
Luis Enrique Erro No. 1, Sta. Ma. Tonantzintla, Puebla, 72840, México

Abstract. Programming a robot to perform a new task normally involves a time consuming process. Reinforcement learning has been used in robotics for learning new tasks through its interaction with the environment. This, however, normally involves long training times. In this paper, we combine several techniques that include human intervention to accelerate the reinforcement learning process. In particular, a user provides initial demonstrations of the task that provide preferences over the search space. The states and actions are represented in an abstracted qualitative way which reduces the state-action space, produces more general policies, and for the case of programming by demonstration, simplifies the correspondence problem. Before converging into an (sub)optimal policy, the robot tries to complete the task, during which the user can provide on-line feedback in the form of commanding actions or qualifiers over the performance of the robot. It is shown that this user-based approach can produce significant reduction in the learning process when compared to more traditional approaches.

Keywords: robot learning, reinforcement learning, programming by demonstration, reward shaping

1 Introduction

Service robots are becoming increasingly popular and it is expected that, in the near future, they will become as common in homes as computers are today. Each user, however, may have different needs for their service robots and such needs may change with time. To personalize service robots to the user's requirements, non-expert users will have to be able to program new robot tasks in natural and accessible ways.

Robot learning of new tasks has been an active research topic within the Reinforcement Learning (RL) community. The idea is to allow the robot to learn a control policy from its interaction with its environment [14]. Even if the user could easily specify a goal and assuming the robot has a suitable representation for states and actions, RL is a very time-consuming process. Also exploratory actions by a robot in a home environment could be disastrous without human supervision (e.g., the robot could break delicate ornaments or fall down the stairs, etc.). In this paper, we describe a learning framework to instruct a robot how to perform a new task in a "natural" way involving human intervention that avoids some of the previously mentioned problems.

In particular, we combine RL with Programming by Demonstration (PbD) [2]. In PbD the user shows the robot how to perform a task and the robot reproduces this task according to its capabilities. Some disadvantages of this approach are that the performance of the system depends on the skills of the teacher and that this approach normally requires of special hardware and controlled conditions. In our approach, we are not restricted to a single expert demonstration under controlled conditions. One or more non-expert users can show the robot how to perform the target task, the demonstrations are expected to be suboptimal and noisy and can even follow different strategies.

Trace logs of the demonstrations are obtained and transformed into a relational qualitative representation and then given to a reinforcement learning algorithm. This means that the states and actions, over which the learning process takes place, are not completely defined in advance, but are constructed from the user's demonstrations, in effect defining preferences for the learning task. Also, this more abstracted representation allows the system to learn generalized policies applicable to different instances of the task.

Contrary to previous approaches the user can provide on-line feedback, if necessary, while the robot is attempting to complete the task with its current policy, to accelerate the learning process. This feedback is traduced into additional rewards in an kind of dynamic reward shaping function as it depends on when the user decides to provide feedback and what kind of feedback is given.

All these ingredients form a human-oriented framework to teach a new task to a robot by non-expert users. We have previously reported results following some of these ideas in [18, 15, 7]. In this paper, we present a general learning framework that comprises and extends, in a unified way, our previous work and provides additional results for a pick-&-place task.

This paper is organized as follows: Section 2 describes the most closely related work to this research. In Section 3 the proposed system is described in detail. Section 4 details the experiments and discusses the results obtained in this research. In Section 5, conclusions and future research work directions are given.

2 Background and Related Work

There are several related areas to our research, these include: reinforcement learning, programming by demonstration, reward shaping, relational representations, and human intervention. In the following sub-sections we will concentrate mainly on reinforcement learning approaches.

Reinforcement Learning (RL) is a technique used to learn in an autonomous way a control policy (π) in a sequential decision process. It can be characterized as a Markov Decision Process (MDP): $M =< S, A, R, P >$ where, S is a set of states, A is a set of actions, R is a reward function, and P a probability state transition function. In RL, for a given state $s \in S$ at time t, an agent (the robot) chooses an action $a \in A$, transitions (moves) into a new state s' and receives a reward $r_t \in R$. A sequence of actions eventually leads the agent to a terminal (goal) state. Formally, the control policy $\pi(s, a)$ is a mapping function that gives the probability of taking action a when in state s. The goal of RL is to learn an optimal control policy π^* that produces the maximum total expected reward for the agent [14].

Learning an optimal control policy normally requires the exploration of the whole search space and very long training times and different approaches have been suggested to produce faster convergence rates, like reward shaping and human feedback.

The idea of reward shaping is to give additional rewards to a learning agent to guide its learning process and converge faster [11, 6]. In effect, the learning algorithm is running on a transformed MDP, $M' =< S, A, T, R' >$, where $R' = f(R, s, s')$. Normally f has been used as an additive function, i.e., $R' = R + F$, but in general, it does not need to be the case. So in the new MDP when an agent takes in state s action a and moves to s' it receives a reward defined as $R + F$.

Reinforcement learning including feedback has been considered in some recent approaches [17, 10, 3, 9, 8]. Hardware devices such as joysticks, keyboards, among others, are used to provide such feedback. Other approaches use voice commands to teach how to perform a task, as presented in [13, 19]. In [13] a human instructor demonstrates how to do a task and gives instructions with voice commands. Their verbal instructions, however, are very similar to control structures of a programming language that can be difficult to give by general end-users, and the learning is focused on learning by demonstration. By contrast, the method that we propose uses a more natural spoken interaction and uses reinforcement learning in addition to the demonstrations.

Some authors have provided feedback from the user and incorporated it into the reinforcement learning algorithm [4, 1, 5]. In [1] the robot first derives a control policy from user's demonstrations and the teacher modifies the policy through a critiquing process. A similar approach is taken in [4], however the user's critique is incorporated into the optimization function used to learn the policy. In [5], the authors combine TAMER, an algorithm that models a hypothetical human reward function, with eight different reward shaping functions.

Contrary to previous works, our method starts with traces of demonstrations provided by a teacher and the user can provide, through voice commands, feedback that can be given at any time during the learning process, directly affecting the reward function. We also use a more powerful (relational) representation language to create more general policies, as explained later.

3 Learning from Human Demonstration and Shaping

Our learning framework involves three stages: (i) human demonstrations of how to perform a task, (ii) a transformation of the trace-logs into a relational representation, and (iii) a reinforcement learning scheme with possible on-line feedback by the user (see Figure 1). The general idea is to involve the user during the learning process in a more "natural" way.

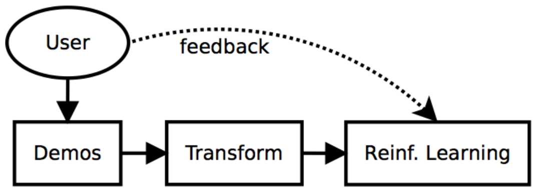


Fig. 1. General learning scheme

The demonstrations can be given by the one or by several users, they can follow different strategies, they are expected to be sub-optimal, and the information from them can be noisy. We want from them to learn how to perform the task in an (sub)optimal way to the robot, for which we use a modified reinforcement learning algorithm. We also want to learn a policy that can be used by the robot, even under different, although similar, settings. For example, if the user teaches the robot how to exit a room, we would like the robot to be able to exit any room. For that purpose we change the low-level sensor information from the demonstrations into a more abstracted relational representation from which generalized policies can be learned. Finally, in order to accelerate the learning process, the user can intervene, at any time, by providing on-line feedback to the system, which is translated into additional rewards, in a kind of dynamic reward shaping function. These steps are described in more detail in the following sections.

3.1 Relational representation

There are several advantages for using a high-level representation for learning tasks: (i) An abstracted state or action can represent several more specific states and actions, thus reducing the search space, and (ii) the learned policies represented in this abstracted formalism can be used in other, although similar, domains without any further learning.

For example, an abstracted state could be represented by the conjunction of two predicates: *place(in-room,State) and doors_detected(right,close,State)*, representing that the robot is in a room and a door is not too far away to its right. So all the instances (potentially infinite) that satisfies these two conditions are represented by this single state.

Similarly, low level actions are transformed into generalized actions represented by an action predicate and the set of applicable state predicates as preconditions. For instance:

If place(in-room,State) and doors_detected(right,close,State)
Then turn(right,State)

meaning that when the robot is in a room with a door to its right then one possible action is to turn to the right, representing all the possible movements to the right of the robot under these conditions.

The states and actions relevant for the task are not predefined in advanced, as in traditional RL algorithms, but are constructed from the user's demonstrations. This has two immediate effects: (i) Only a subset of states and actions are considered for learning and consequently the algorithm can converge faster. (ii) There are no guarantees of finding optimal policies and the robot may not know what to do when encountering new states. Also during the learning process the agent performs exploratory actions that can lead the robot to new (unseen) states. In such cases, a new abstracted state is constructed, and the robot can ask the user what to do or perform a primitive action, and construct a new action for that state. The states and actions are incrementally constructed from the user's demonstrations as described in Algorithm 1.

Every sensor produces information at a particular rate. Once new information has been received from all the sensors (a frame), it is transformed into states and actions, if new. For instance, the definition of the predicate *doors_detected*, involves finding two discontinuities from laser readings. A discontinuity is defined as an abrupt variation in the measured distance of two consecutive laser readings. A door is detected if a right discontinuity (increased distance from two consecutive readings) is followed by a left discontinuity (decreased distance from two consecutive readings). The door's orientation angle and distance values are calculated by averaging the values of the right and left discontinuities angles and distances. In our research, the user needs to define the predicates, i.e., how to transform the low-level information from the sensors into a set of relational predicates, such as *place*, *doors_detected*, etc., and how to transform the information from the actuators into action predicates, such as *turn*, etc.

Once a set of states and actions are defined from the sensor readings obtained in the demonstrations, a reinforcement learning algorithm is used to learn a policy for the task, as described in the following section.

Algorithm 1 Transformation into a Relational Representation.

```
Given: $\tau_1, \tau_2, \dots \tau_n$, a set of $n$ demonstrations of how to perform a task
Given: a set of predicate definitions
for $i = 1$ to $n$ do
    $k \leftarrow$ number of $frames$ in demonstration $i$
    for $j = 1$ to $k$ do
        Transform $frame_{ij}$ ($frame$ $j$ from demonstration $i$) into a set of applicable
        state predicates (r-state) and action predicate (r-action)
    end for
end for
Output: r-state-r-action pairs
```

3.2 Reinforcement Learning

As the demonstrations correspond to different examples of the same task and as they might have been generated by different users, there can be several actions associated to the same state. *RL* is then used to develop a control policy that selects the best action for each state. The goal of this stage is to improve over the traces performed by the user.

Algorithm 2 gives the pseudo-code for Q-learning using this relational representation, although other RL algorithms could be used as well.

Algorithm 2 The rQ-learning Algorithm, where α is the learning rate and γ the discount rate.

```
Initialize $Q(s_R, a_R)$ arbitrarily
repeat
  Initialize $s$
  $s_R \leftarrow rels(s)$ % set of relations on state $s$
  for each step of episode do
    Choose $a_R$ from $s_R$ using a persistently exciting policy (e.g., $\epsilon$-greedy)
    Randomly choose action $a$ applicable in $a_R$
    Take action $a$, observe $r$ (reward), $s'$ (next state)
    $s'_R \leftarrow rels(s')$
    $Q(s_R, a_R) \leftarrow Q(s_R, a_R) + \alpha(r + \gamma max_{a'_R} Q(s'_R, a'_R) - Q(s_R, a_R))$
    $s_R \leftarrow s'_R$
  end for
until $s$ is terminal
```

This is like a normal Q-learning algorithm, however, the action is a randomly selected action from the possible instantiations of the r-action, and once it is executed, the sensor readings from the resulting state are transformed into a set of relations to identify the corresponding r-state. The reward function (r) is a traditional reward function defined in advanced by the user, in our experiments we used the values of 100 when reaching a goal, -10 when reaching a state out of the robot's working area, and -1 otherwise.

The user's demonstrations focus the search space into a small set of possible actions and also are used to seed initial Q-values for the states and actions visited by the user. All the state-action pairs involved in each demonstration are used as a tried episode in the RL algorithm, and consequently their Q-values are affected accordingly. During the exploration stage the robot can visit new states and create new state-action, either by asking the user what action to perform in such new state or choosing a random primitive action. The robot learns policies in terms of this representation. For instance, if in the previous state the robot learns that the best action is to turn right, even if the robot is in a completely new environment, as long as it recognizes a room and a door to its right it will turn to its right.

3.3 Dynamic Reward Shaping

A natural way of teaching is to instruct a student how to perform a task, let the student try the task on its own, and give feedback to the student while she/he is attempting to complete the task. This is basically the idea followed with our dynamic reward shaping approach.

Our reward function is defined as: $R = R_{RL} + R_{user}$ where R_{RL} is the reward function from Algorithm 2 that comes from the definition of the task and R_{user} is the reward obtained from voice commands given by the user. The main difference with previous reward shaping functions is that in our case the rewards can be given sporadically and can be contrary to what it is needed for achieving a goal. The feedback from the user is given while the agent is attempting to complete an episode using its current policy, either as a critique to the states reached and actions followed by the robot or as a command for the robot to execute an alternative action. User's feedback is transformed into rewards and used at that instance as a shaping reward. If the user does not intervene, the agent follows a normal reinforcement learning process.

This approach, however, pose several problems: (i) Even with simple voice commands, speech recognition systems are not completely reliable which can produce a noisy reward shaping function, (ii) the user can provide her/his rewards with certain delay, and (iii) the user is not consistent with her/his feedback and it can vary over time, at first providing a substantial amount of feedback and after a while just giving occasional suggestions.

On the other hand, this feedback can be used to correct user's demonstrations, corrrect delayed rewards, or even create new alternative goals by persistently giving positive feedback to certain states.

4 Experiments and Results

In this paper, we focus on learning how to perform a manipulation task (pick and place) using demonstrations from the user and a Kinect sensor to capture the movements of the user. We used a 6 DoF robot manipulator, named Armonic Arm 6M (see Figure 2 right), in our experiments.

The interaction between the different components of the system is shown in Figure 3, where the initial demonstrations are used to seed the initial Q-values and the system follows a process where the user can intervene during the RL process.

In the demonstrations, the instructor shows the robot the task to learn with his/her arm movements (see Figure 4). The 3D positions of the hand and of the objects in the environment are tracked using the Kinect® sensor. Each state $s \in S$ is described by a six-term tuple with the following elements: $s = (H, W, D, dH, dW, dD)$, where:

- H = Height: $\{Up, Down\}$
- W = Width: $\{Right, Left\}$
- D = Depth: $\{Front, Back\}$

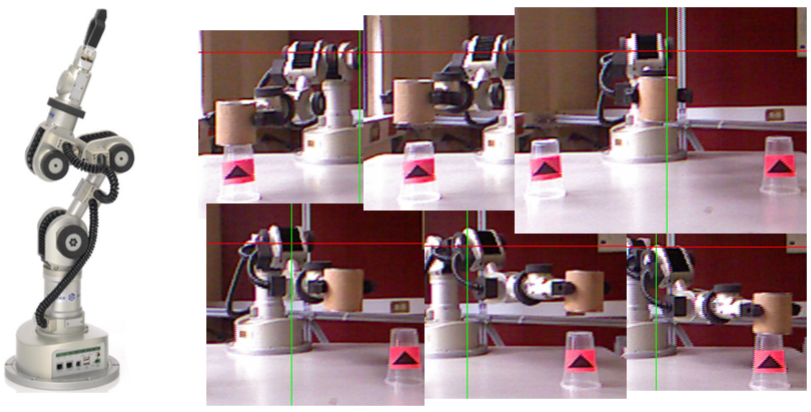


Fig. 2. Robot Katana Armonic Arm 6M

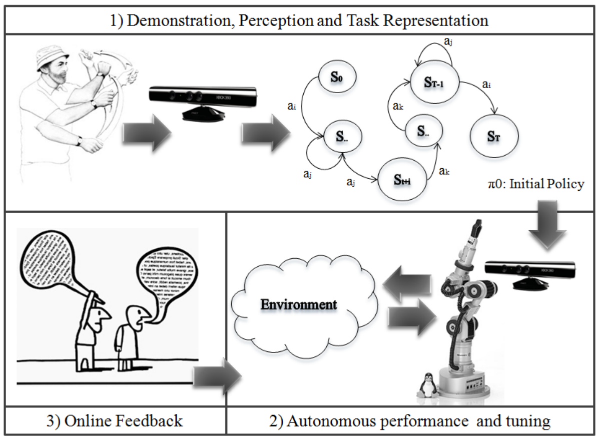

1) Demonstration, Perception and Task Representation
S0
S..
ST-1
S..
St+1
ST
ai
aj
ak
π0: Initial Policy
Environment
3) Online Feedback
2) Autonomous performance and tuning

Fig. 3. The imitation and feedback learning framework.

- dH = Height distance to target: $\{VeryFar, Far, Close, VeryClose, Over\}$
- dW = Width distance to target: $\{VeryFar, Far, Close, VeryClose, Over\}$
- dD = Depth distance to target: $\{VeryFar, Far, Close, VeryClose, Over\}$

In this case, this is just a discretization of states, however, in other more complex domains, the predicate definition can involve predicate variables.

Each action $a \in A$ is described as a movement in one direction with information of how much to move the manipulator, $a = (D, pD)$, where:

- D : Direction $\{Up, Down, Right, Left, Front, Back\}$
- pD : a real value that defines the magnitude of the movement performed by the robot according to how close it is from an object. For example, a *right* movement will have a greater displacement to the right when it is far from the target object than a *right* movement when it is close to the target object.

The main advantage of this representation is that, since it is a relative relation between the human or robotic arm with the target object, it does not need to have any special transformation between the traces shown by the user and the traces used by the robot. On the other hand, the states and the learned policies, as it will be shown later, are relative to the target object so the initial position of the robot arm and the initial and final position of the target object can be completely different from the positions shown by the user, and the learned policy is still suitable for the task.

During the execution of actions it is possible to produce continuous actions by combining the discrete actions of the current policy. This is performed as a lineal combination of the discrete action on each direction with the larger Q-values. The lineal combination is proportional to the magnitude of the used Q-values and the updating function over the Q-values is also proportionally performed over all the involved discrete actions.

While the robot is exploring the environment to improve over its current policy, the user can provide on-line voice feedback to the robot. We build over the work described in [16], where a fixed vocabulary was defined for the user's commands. In the experiments, we associated rewards to certain words of the vocabulary: +100 for reaching the goal, +50 for "excellent", +10 for "good", −50 for "terrible", and −10 for "bad". Similar rewards were used for the rest of the vocabulary and the phrases. We used Sphinx 3 as speech recognizer based on the corpus DIMEx100 [12].

For the experiments, we designed different conditions to test the individual parts of the proposed system:

1. Using only Reinforcement Learning (RL)
2. Human demonstrations followed by Reinforcement Learning (HD)
3. Human demonstrations followed by Reinforcement Learning interleave with simulation (S)
4. Human demonstrations followed by Reinforcement Learning, interleaved Simulation and User's Feedback (FB)

Table 1. Translation into English of part of the vocabulary used in the experiments. We used individual words and simple short phrases. We considered six possible actions: Up, Down, Right, Left, Front, Back and a small set of qualifiers arranged into five categories (goal, very good, good, bad, and very bad).

WORDS	SHORT PHRASES
forward	move forward
backward	move backwards
left	turn to your left
right	go to your right
up	move upwards
down	move down
end	stop here
good	continue like this
bad	not that way
excellent	very good
terrible	not like that
goal	until here

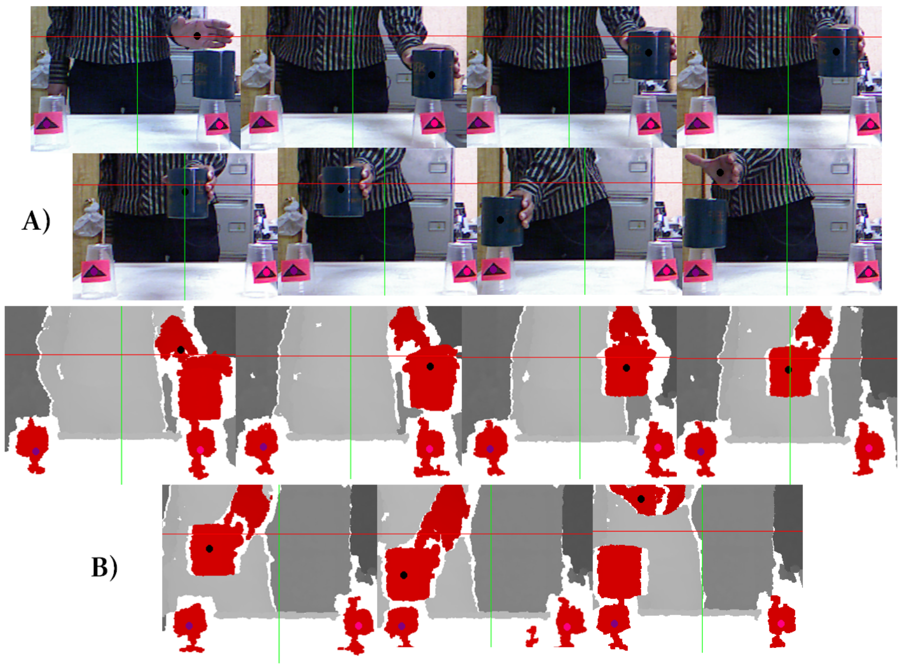
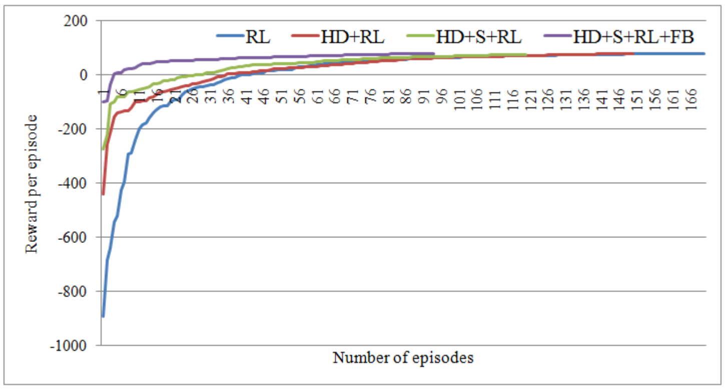


Fig. 4. Human demonstration for picking-up and placing a particular object (top) and the output produced by the Kinect (bottom).

Figure 5 shows the performance of the different experiment's settings, plotting the accumulated reward per episode (top graph) and the time to complete the task per episode (bottom graph).



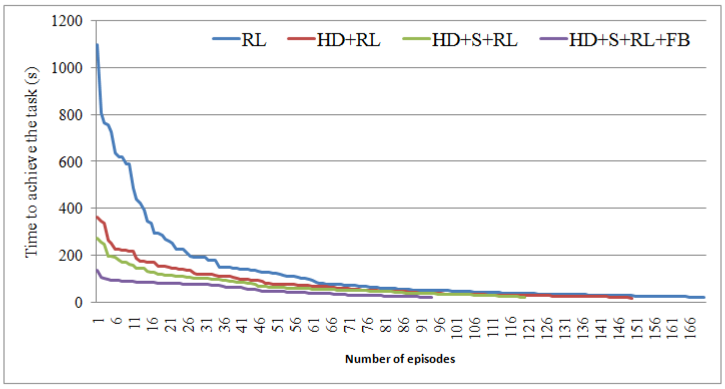


Fig. 5. Performance of the different experimental conditions. The top figure shows the accumulated reward, while the bottom figure shows the total time to complete the task. Where; (i) RL = reinforcement learning, (ii) $HD + RL$ = human demonstration followed by RL, (iii) $HD + S + RL$ = RL with human demonstrations interleave with learning simulation time, and (iv) $HD+S+RL+FB$ = the same as (iii) but including feedback from the user.

The experiments are repeated 12 times and averaged. As can be seen, from the figure, human demonstrations provides a significant *jump start* in the experiments. Also using human demonstration and user's feedback during the learning process significantly reduce the convergence times. It should be noted that during training each episode started from a random initial position and ended in a random (reachable) object position.

Table 4 shows the total training times using a real robot under the different testing conditions. The user roughly spent 5 minutes for the demonstrations and we interleave a few seconds of simulation after each episode.

The algorithm converges even when there is no control on when and how to provide feedback. It should be noted that in these experiments the user was not

Table 2. Total training times with a real robot.

	Time (s)			
	Demos	Sim.	RL	Total Time
RL	0	0	16168.896	16168.896 ($\sim$ 4:30 h)
HD + RL	$\sim$ 300	0	11056.56	11356.56 ($\sim$ 3:10 h)
HD + S + RL	$\sim$ 300	25.628	6729.399	7055.027 ($\sim$ 2 h)
HD + S + RL + FB	$\sim$ 300	19.348	3242.273	3561.621 ($\sim$ 1)

a student from Computer Science or related areas and has never worked with robots or machine learning systems before.

Analyzing the human interventions, we noticed that roughly 60% of the interventions were action commands, while 40% were qualifiers over the performance of the robot. Also, the number of human interventions decreased almost by half when human demonstrations are given beforehand. As future work, we would like to see if this percentages are similar for different users.

We tested the learned policy over 100 random initial and final reachable positions for the objects. The robot was able to successfully complete 88% of the tests (the object is picked-up from its initial position and correctly placed in its target position). The tests were incomplete 7% of the time (the object cannot be taken by the gripper or falls from the gripper during its transfer to its target position, but the robot was able to reach the initial and target position), and the robot did not reach either the initial or target position in 5% of the tests.

These experiments show that with a more abstracted representation the learned policies can be used in other instantiations of the task. and reduces the *correspondence problem*[1]. The system is relatively robust to noisy demonstrations (our speech recognition system roughly misunderstands 20% of the utterances). The incorporation of user's feedback during the learning process provides a powerful technique for more natural human teaching interaction. Finally, interleaving, even short learning simulation time between episodes, is an attractive alternative to accelerate the learning process and also provides a more natural teaching setting.

5 Conclusions and Future Work

Teaching a robot how to perform new tasks will soon become a very relevant topic with the advent of service robots. We want non-expert users to be able to teach robots new tasks in natural ways. In this paper, we propose a learning framework for teaching robots new tasks oriented to non-expert users. This approach uses human demonstrations to focus the search space and accelerates the learning process. Transforms the original log-traces from the sensors of the robot into a relational representation, which accelerates the learning process. And allows human feedback during the learning stages, thus, allowing a more natural

[1] The problem of creating adequate correspondence between the human and robot morphologies.

interaction from the user into the learning loop. This process is relatively robust to noise and errors from the user or speech understanding system, and can be used to correct faulty demonstrations.

As future work, we would like to extend our speech recognition system to provide more natural interactions and to consider different intentions. It is not the same to shout *stop*! to a robot heading towards a staircase than to tell gently the robot to *stop* before receiving a new command. Our current feedback is translated into a numeric value. We would like to explore using a qualitative representation for the reward function. The demonstrations are shown by the user, alternatively we would like to give only general guidelines, like “exit the room”, “go to the end of the aisle”, etc. We would also like to include commands to undo some actions or rewards.

Acknowledgments

This work was supported by grant number **84162** from CONACyT, México.

References

1. B. Argall, B. Browning, and M. Veloso. Learning by demonstration with critique from a human teacher. In *2nd Conf. on Human-Robot Interaction (HRI)*, pages 57–64, 2007.
2. Aude G. Billard, Sylvain Calinon, Ruediger Dillmann, and Sstefan Schaal. *Robot programming by demonstration, in: B. Siciliano, O. Khatib (Eds.), Handbook of Robotics*, chapter 59. Springer, New York, NY, USA, 2008.
3. F. Idia, M. Tabata, and F. Hara. Generating personality character in a face robot through interaction with human. In *Proc. of the 7th IEEE International Workshop on Robot and Human Communication*, pages 481–486, 1998.
4. K. Judah, S. Roy, A. Fern, and T.G. Dietterich. Reinforcement learning via practice and critique advice. In *Proceedings of the Twenty-Fourth AAAI Conference on Artificial Intelligence (AAAI-10)*, 2010.
5. W.B. Knox and P. Stone. Combining manual feedback with subsequent mdp reward signals for reinforcement learning. In *Proc. of the 9th. International Conference on Autonomous Agents and Multiagent Systems (AAMAS 2010)*, 2010.
6. A. Laud. *Theory and Application of Reward Shaping in Reinforcement Learning.* PhD thesis, University of Illinois, 2004.
7. A. León, E.F. Morales, L. Altamirano, and J.R. Ruiz. Teaching a robot to perform task through imitation and on-line feedback. In *Proc. of the 16th. Iberoamerican Congress on Pattern Recognition (CIARP-2011)*, pages 549–556. LNCS-7042, Springer, 2011.
8. T. A. Lockerd, G. Hoffman, and C. Breazeal. Reinforcement learning with human teachers: Understanding how people want to teach robots. In *Proc. of the 15th IEEE International Symposium on Robot and Human Interactive Communication*, pages 352–357, 2006.
9. T.A. Lockerd and C. Breazeal. Reinforcement learning with human teachers: Evidence of feedback and guidance with implications for learning performance. In *21st National Conference on Artificial Intelligence*, 2006.

10. T.A. Lockerd, G. Hoffman, and C. Breazeal. Real-time interactive reinforcement learning for robots. In *Workshop on Human Comprehensible Machine Learning*, 2005.
11. A.Y. Ng, D. Harada, and S. Russell. Policy invariance under reward transformations: Theory and application to reward shaping. In *Proceedings of the Sixteenth International Conference on Machine Learning*, pages 278–287. Morgan Kaufmann, 1999.
12. L. Pineda. *Corpus DIMEx100 (Level T22).* UNAM, 2010.
13. Paul E. Rybski, Kevin Yoon, Jeremy Stolarz, and Manuela M. Veloso. Interactive robot task training through dialog and demonstration. In *In Proceedings of the 2007 ACM/IEEE International Conference on Human-Robot Interaction, Washington D.C*, pages 255–262, 2007.
14. R.S. Sutton and A.G. Barto. *Reinforcement learning: An introduction.* The MIT Pres, Cambridge, MA, London, England, 1998.
15. A.C. Tenorio-Gonzalez, E.F. Morales, and L. Villaseñor Pineda. Dynamic reward shaping: training a robot by voice. In *Proc. of the 12th Ibero-American conference on Advances in artificial intelligence*, IBERAMIA'10, pages 483–492, Berlin, Heidelberg, 2010.
16. A.C. Tenorio-Gonzalez, E.F. Morales, and L. Villaseñor Pineda. Teaching a robot to perform tasks with voice commands. In *Proc. of the 9th Mexican international conference on Advances in artificial intelligence: Part I*, MICAI'10, pages 105–116, Berlin, Heidelberg, 2010. Springer-Verlag.
17. Y. Wang, M. Huber, V. N. Papudesi, and D. J. Cook. User-guided reinforcement learning of robot assistive tasks for an intelligent environment. In *Proceedings of the IEEE/RJS International Conference on Intelligent Robots and Systems (IROS), Las Vegas, NV.* IEEE, 2003.
18. J.H. Zaragoza and E.F. Morales. Relational reinforcement learning with continuous actions by combining behavioural cloning and locally weighted regression. *JILSA*, 2(2):69–79, 2010.
19. Y. Zhang and J. Weng. Action chaining by a developmental robot with a value system. In *Proceedings of the 2nd International Conference on Development and Learning*, pages 53–60. MIT Press, 2002.