2006 IEEE Conference on
Systems, Man, and Cybernetics
October 8-11, 2006, Taipei, Taiwan

Human-Robot Collaborative Learning System for Inspection

Kartoun Uri, *Student Member, IEEE*, Stern Helman, *Member, IEEE*, and Edan Yael, *Member, IEEE*

***Abstract*—This paper presents a collaborative reinforcement learning algorithm, $CQ(\lambda)$, designed to accelerate learning by integrating a human operator into the learning process. The $CQ(\lambda)$-learning algorithm enables collaboration of knowledge between the robot and a human; the human, responsible for remotely monitoring the robot, suggests solutions when intervention is required. Based on its learning performance, the robot switches between fully autonomous operation, and the integration of human commands. The $CQ(\lambda)$-learning algorithm was tested on a Motoman UP-6 fixed-arm robot required to empty the contents of a suspicious bag. Experimental results of comparing the $CQ(\lambda)$ with the standard $Q(\lambda)$, indicated the superiority of the $CQ(\lambda)$ while achieving an improvement of 21.25% in the average reward.**

I. INTRODUCTION

TELEOPERATION is used when a task has to be performed in a hostile, unsafe, inaccessible or remote environment [1]. [2] suggest two components of a human-robot system when the robot is remotely located; (i) autonomy mode - an artificial intelligence or computer control of a robot that allows it to act, for a time, without human intervention, and (ii) Human-Robotic Interfaces (HRI) - software installed at the human's location allowing him to perceive the world, the robot states, and send instructions to the robot. One of the main issues in task-oriented HRI is achieving the right mixture of human and robot autonomy [3]. [4], indicates the importance of HRI in meeting operators' requirements: "an understanding of the human decision process should be incorporated into the design of human-robotic interfaces in order to support the process humans' employ."

[5], describe a HRI that supports both adjustable autonomy and hierarchical task selection. With adjustable autonomy, a computer switches among several control modes ranging from full supervision to full autonomy. With hierarchical task selection, the interface allows an operator to easily solve a high-level task autonomously or else to guide a robot through a sequence of lower-level subtasks that may or may not involve autonomous control. [6], 2005 define sliding scale autonomy as the ability to create new levels of autonomy between existing, pre-programmed autonomy levels. The suggested sliding scale autonomy system shows the ability to dynamically combine human and robot inputs, using a small set of variables such as user and robot speeds, speed limitations, and obstacle avoidance.

Reinforcement learning (RL) is regarded as learning through direct experimentation [7], [8]. It does not assume the existence of a teacher providing training examples. Instead, teaching derives from experience. The learner acts on the process to receive signals (reinforcements) from it, indications about how well it is performing the required task. These signals are usually associated with some dramatic condition - e.g., accomplishing a subtask (reward) or complete failure (punishment). The learning agent learns the associations between observed states and chosen actions that lead to rewards or punishments, i.e., it learns how to assign credit to past actions and states by correctly estimating costs associated with these events [9].

In [10], a collaborative process enabling a robotic learner to acquire concepts and skills from human examples is presented. During the teaching process, the robot must perform tasks based on human instructions. The robot executes its tasks by incorporating feedback until its hypothesis space is converged. Using a Q-learning approach, the robot learns a button pushing task. In [11], a variable autonomy approach is used. User commands serve as training inputs for the robot learning component, which optimizes the autonomous control for its task. This is achieved by employing user commands for modifying the robot's reward function. Using the potential of learning from reinforcement and human rewards illustrates the changes in user reward and Q-value functions accordingly [12], [13]. The task was to learn to optimally navigate to a specific target in a two-dimensional world with obstacles.

$Q(\lambda)$-learning requires no human intervention; the agent is placed in an unknown environment and explores it independently with the objective of finding an optimal policy. A disadvantage of this approach is the large amount of required interaction with the environment until an effective policy is determined. One example for alleviating this problem includes guiding an agent using rules suggesting trajectories of successful runs through the environment [14], [15]. [15] suggest a RL-based framework denoted as "relocation". At any time during training an agent can request to be placed in any state of the environment. The "relocation" approach assumes a cost per relocation, and seeks to limit the number of relocations. The approach requires minimal human involvement and consists of two agent conditions: (i) "in trouble" - taking actions that turn out to be a poor choice, even though it learns from the negative experience, would cause to a waste of time-steps in a part of the state-space that is unlikely to be visited during optimal behavior, and (ii)

Manuscript received March 15, 2006. This work was partially supported by the Paul Ivanier Center for Robotics Research and Production Management, Ben-Gurion University of the Negev and by the Rabbi W. Gunther Plaut Chair in Manufacturing Engineering.

"bored" - if the Q-values are updated by tiny amounts, the agent is not learning anything new in the current part of the environment. In this condition it is forced to relocate with greatest probability when updating a particular Q-value does not change it.

[16] present a cooperative RL algorithm of multi-agent systems denoted as the "leader-following Q-learning algorithm." The algorithm is based on a Markov or stochastic game, in which there are multiple stages and each stage is a static Stackelberg game. [17], investigates the problem of multiple RL agents attempting to learn the value function of a particular task in parallel for the *n*-armed bandit task. A parallel reinforcement learning solution is suggested to overcome the problem of statistic overwhelming by an agent's information that is correspondingly has a larger accumulated experience than the other agents. Experiments on a group of four foraging mobile robots learning to map robots' conditions to behaviors was conducted by Matarić [18]. The learning algorithm of the robots consists of reward functions that combine individual conditions of a robot (such as, "grasped a puck", "dropped puck away from home") and collaborative conditions; how close the robots are to each other. Individually, each robot learns to select the behavior with the highest value for each condition, to find and take home the most pucks. Evaluation of groups of three and four robots found that interference was a detriment; in general, the more robots were learning at the same time, the longer it took for each individual to converge. Additionally, [18] found that while measuring the "percent of the correct policy the robots learned in 15 minutes, averaged over twenty trials," the use of heterogeneous reward functions results in better performance but also suffers from the credit assignment problem.

The usual method for bomb squad personnel is to blow up a suspicious bag and any explosives contained therein. However, if the bag contains chemical, biological or radiological canisters, this method can lead to disastrous results. Furthermore, the "blow-up" method also destroys important clues such as fingerprints, type of explosive, detonators and other signatures of use in subsequent forensic analysis. Learning the extraction of a bag contents by a robot acquiring knowledge from human advice is the subject addressed here. The learning task described in this paper is to observe the position of a plastic bag located on a platform, grasp it with a robot manipulator and shake out its contents on a collection container in minimum time.

Although Q-learning and its variation $Q(\lambda)$ have been used in many robotic fields (e.g., [19]-[24]), accelerating the learning process is important. This paper presents a new algorithm, referred to as $CQ(\lambda)$, that accelerates learning. The $CQ(\lambda)$ algorithm is a collaborative algorithm that integrates the experience of several agents. In this paper, we describe experiments of applying the $CQ(\lambda)$ algorithm on a system that integrates two agents - a robot and a human working cooperatively to achieve a common goal. The system has no *a priori* knowledge regarding to efficient lifting and shaking policies of the bag and it learns this knowledge from experience and from human guidance. Although other alternatives are available for solving the proposed security problem, such as cutting open the bag, or sliding out the inspected objects, the application was selected to serve as a test-bed for testing the $CQ(\lambda)$ algorithm. Section II presents the new $CQ(\lambda)$ algorithm. The test-bed learning application is described in section III followed by experimental results in section IV. Concluding remarks follow in section V.

II. $CQ(\lambda)$ -Learning

A. $CQ(\lambda)$ -Learning for Multiple Agents

The basic assumption in reinforcement learning studies is that any state s_{t+1} made by the agent must be a function only of its last state and action: $s_{t+1} = f(s_t, a_t)$ where $s_t \in S$ and $a_t \in A$ are the state and time at step t, respectively [9]. In Q-learning, the system estimates the optimal action-value function directly and then uses it to derive a control policy using the local greedy strategy [25]. It is stated in [23] that "Q-learning can learn a policy without any prior knowledge of the reward structure or a transition model. Q-learning is thus referred to as a model-free approach." It does not require mapping from actions to states and it can calculate the Q values directly from the elementary rewards observed. Q is the system's estimate of the optimal action-value function [26]. It is based on the action value measurement $Q(s_t, a_t)$, defined in (1):

$$Q(s_t, a_t) = E[r(s_t, a_t) + \gamma V^*(s_{t+1})] = r(s_t, a_t) + \gamma \sum_{s_{t+1} \in S} P(s_{t+1} \mid s_t, a_t) V^*(s_{t+1})], \quad (1)$$

which represents the expected discounted cost for taking action a_t when visiting state s_t and following an optimal policy thereafter. From this definition and as a consequence of Bellman's optimality principle [27], (2) is derived:

$$Q(s_t, a_t) = r(s_t, a_t) + \gamma \sum_{s_{t+1} \in S} P(s_{t+1} \mid s_t, a_t) \max_a Q(s_{t+1}, a_t). \quad (2)$$

The essence of Q-learning is that these characteristics (maximum operator inside the expectation term and policy independence) allow an iterative process for calculating an optimal action. The first step of the algorithm is to initialize the system's action-value function, Q. Since no prior knowledge is available, the initial values can be arbitrary (e.g., uniformly zero). Next, the system's initial control policy, P, is established. This is achieved by assigning to P the action that locally maximizes the action-value. At time-step t, the agent visits state $s_t \in S$ and selects an action

$a_t \in A$, receives from the process the reinforcement $r(s_t, a_t) \in R$ and observes the next state s_{t+1}. Then it updates the action value $Q(s_t, a_t)$ according to (3) which describes a Q-learning one step:

$$Q_{t+1}(s_t, a_t) = (1-\alpha)Q_t(s_t, a_t) + \alpha[\, r(s_t, a_t) + \gamma \hat{V}_t(s_{t+1})], \quad (3)$$

where $\hat{V}_t(s_{t+1}) = \max_{a_t \in A}[Q_t(s_{t+1}, a_t)]$ is the current estimate of the optimal expected cost $V^*(s_{t+1})$ and α is the learning rate which controls how much weight is given to the reward just experienced, as opposed to the old Q estimate. The process repeats until a stopping criterion is met (e.g., robot emptied the bag from contents). The greedy action $\hat{V}_t(s_{t+1}) = \arg\max_{a_t \in A}[Q_t(s_{t+1}, a_t)]$ is the best the agent performs when in state s_t. For the initial stages of the learning process, however, it uses randomized actions that encourage exploration of the state-space. Under some reasonable conditions [28] this is guaranteed to converge to the optimal Q-function [26].

A generalization of Q-learning, represented by $Q(\lambda)$ [25], [29] uses eligibility traces, $e(s_t, a_t)$: the one-step Q-learning is a particular case with $\lambda = 0$ [30]. The Q-learning algorithm learns quite slowly because only one time-step is traced for each action [10]. To boost learning convergence, a multi-step tracing mechanism, the eligibility trace, is used in which the Q values of a sequence of actions can be updated simultaneously according to the respective lengths of the eligibility traces [19].

"The convergence of $Q(\lambda)$ is not assured anymore for $\lambda > 0$, but experience shows that learning is faster" [30]. Several action selection policies are described in the literature where the greedy policy (e.g., [31]) is to choose the best action. Other policies (e.g., "softmax" or "ε-greedy" [32]) are stochastic, and based on choosing a suboptimal policy to explore the state-action space.

The proposed $CQ(\lambda)$ learning algorithm (Fig. 1) has the objective to accelerate learning in a system composed of several similar learning processes. Differently from [18], where the learning algorithms of multiple robots consist of reward functions that combine individual conditions of a robot, the $CQ(\lambda)$ learning algorithm is based on a state-action value of an agent or learning process updated according to the maximal value within all other state-action values existing in the learning system (4); collaboration is in taking the maximum of action values, i.e., the Q-value, across all learners at each update step [33]. Similar to the "leader-following Q-learning algorithm" in the joint policy approach described in [16], the $CQ(\lambda)$ learning algorithm enables collaboration of knowledge between several agents (the human and the robot). Unlike [12] and [13], the $CQ(\lambda)$ algorithm does not allow human rewards inserted directly to its Q-value functions. $CQ(\lambda)$ rewards are achieved by interaction of learning agents with the environment.

Initialize $Q_i(s,a) = 0$ and set eligibility trace $e_i(s,a) = 0$ for all (s,a) $i \in \{1, 2, \ldots N\}$ where N is the number of learning processes (*e.g.*, robot, human).
Repeat (for each learning process):
  Repeat (for each learning episode):
    Set initial state s_{i_t} and pick initial action a_{i_t}
    Repeat (for each step of episode):
      Take action a_{i_t}, observe reward r_{i_t} and the next state $s_{i_{t+1}}$
      Choose $a_{i_{t+1}}$ for $s_{i_{t+1}}$ using a certain policy (*e.g.*, softmax)
      $\delta_{i_t} \leftarrow r_{i_t} + \gamma_i Q_i(s_{i_{t+1}}, a_{i_t}^*) - Q_i(s_{i_t}, a_{i_t})$
      $e_{i_t}(s_{i_t}, a_{i_t}) \leftarrow e_{i_t}(s_{i_t}, a_{i_t}) + 1$
      For all (s_{i_t}, a_{i_t}):
        $Q_i(s_{i_{t+1}}, a_{i_{t+1}}) \leftarrow \max_{i \in N}[Q(s_{i_t}, a_{i_t}) + \alpha_{i_t} \delta_{i_t} e_i(s_{i_t}, a_{i_t})]$
      If $a_{i_{t+1}} = a_{i_t}^*$, then $e_{i_t}(s_{i_t}, a_{i_t}) \leftarrow \gamma\lambda e_{i_t}(s_{i_t}, a_{i_t})$
        else $e_{i_t}(s_{i_t}, a_{i_t}) \leftarrow 0$
    $s_{i_t} \leftarrow s_{i_{t+1}}$; $a_{i_t} \leftarrow a_{i_{t+1}}$
until a stopping condition

Fig. 1. $CQ(\lambda)$-learning algorithm

$$Q_i(s_{i_{t+1}}, a_{i_{t+1}}) \leftarrow \max_{i \in N}[Q(s_{i_t}, a_{i_t}) + \alpha_{i_t} \delta_{i_t} e_i(s_{i_t}, a_{i_t})] \quad (4)$$

In (4) δ_{i_t} is the temporal difference error that specifies how different the new value is from the old prediction and $e_i(s_{i_t}, a_{i_t})$ is the eligibility trace that specifies how much a state-action pair should be updated at each time-step. When a state-action pair is first visited, its eligibility is set to one. Then at each subsequent time-step it is reduced by a factor $\gamma\lambda$. When it is subsequently visited, its eligibility trace is increased by one [34].

B. $CQ(\lambda)$-Learning for Human-Robot Systems

When only two learning agents are involved such as a robot and human (Fig. 2), the robot learning function acquires state-action values achieved from policies suggested by a human operator (HO). In this case, robot learning performance measure is defined as Λ, a minimum acceptable performance threshold in which above the human is called. The measure Λ is compared with the average number of rewarded policies, L_{ave}, over the last N recent learning trials considered (5):

$$L_{ave} = \left(\sum_{i=t-N}^{t} (\mathrm{N}_i) \right) / N, \quad (5)$$

where t is the current learning trial, $i = t-N, t-N+1, t-N+2, \ldots t$, and $N_i \in \{0,1\}$ notifies whether a policy was successful ($N_i = 1$ - at least one item fell from the bag) or failed ($N_i = 0$ - no items fell from the bag) for the i^{th} trial. Based on this learning performance threshold, the robot switches between fully autonomous operation and the integration of human commands.

Two levels of collaboration are defined: (i) autonomous - the robot decides which actions to take, acting autonomously

according to its $Q(\lambda)$ learning function, and (ii) semi-autonomous - HO suggests actions remotely and the robot combines this knowledge, i.e., $CQ(\lambda)$ -learning is being performed. Human-robot collaboration is unnecessary as long as the robot learns policies autonomously, and adapts to new states. The HO is required to intervene and suggest alternative shaking parameters for shaking policies if the robot reports that its learning performance is low (6).

$$L_{ave} = \left(\sum_{i=t-N}^{t} (\mathrm{N}_i) \right) / N > \Lambda \,, \tag{6}$$

The robot learning performance threshold, Λ, a pre-defined minimum acceptable performance measure in which above the human is called is compared with L_{ave}, the average number of rewarded policies over the last N recent learning policies performed, i.e., robot switches its learning level from autonomous (self-shaking performing) to semi-autonomous (acquiring human knowledge) based on its learning performance (see example in Fig. 3).

III. Collaborative Learning Experiment

A. Task Definition

The system is defined by $\Sigma = [R, O, E]$ where R is a robot, O an object, and E an environment contains a platform on which the inspected bag is manipulated. T is a task performed on O, using R, within the environment E. For the task of emptying the contents of a suspicious plastic bag, learning task, T, is to observe the position of the bag, located on a platform, grasp it with a robot manipulator and shake out its contents in minimum time. It is assumed that the number of items in the bag is known in advance.

Robot states denoted as $s_t \in S$ (Table I) are its gripper location in a three-dimensional grid (Fig. 4). The performance of the task T is a function of a set of actions, $a_t \in A$, for each physical state of Σ.

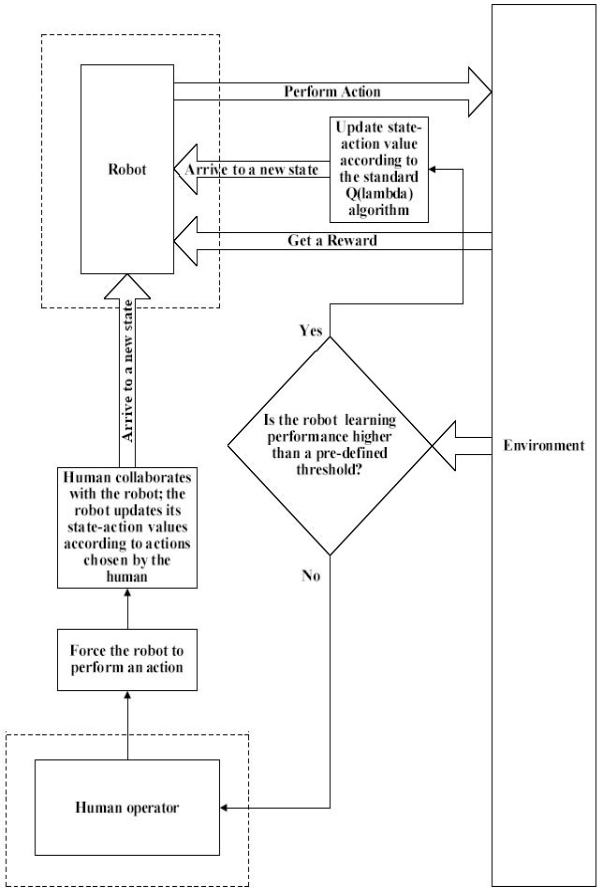

Fig. 2. Robot and a human operator $CQ(\lambda)$ -learning

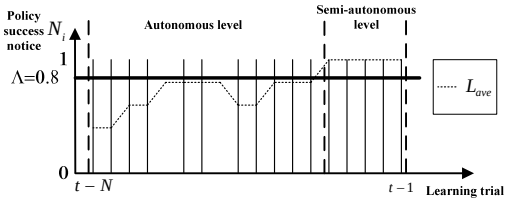

Fig. 3. Example of learning performances

TABLE I
States Description of the Three-Dimensional Grid

State(s)	Description	Number of states
$s_{(Center)_t}$	state center	1
$s_{(X_{3-})_t}$, $s_{(X_{2-})_t}$, $s_{(X_{1-})_t}$, $s_{(X_{1+})_t}$, $s_{(X_{2+})_t}$, $s_{(X_{3+})_t}$	states where the robot can move over its X axis	6
$s_{(Y_{3-})_t}$, $s_{(Y_{2-})_t}$, $s_{(Y_{1-})_t}$, $s_{(Y_{1+})_t}$, $s_{(Y_{2+})_t}$, $s_{(Y_{3+})_t}$	states where the robot can move over its Y axis	6
$s_{(Z_{3-})_t}$, $s_{(Z_{2-})_t}$, $s_{(Z_{1-})_t}$, $s_{(Z_{1+})_t}$, $s_{(Z_{2+})_t}$, $s_{(Z_{3+})_t}$	states where the robot can move over its Z axis	6

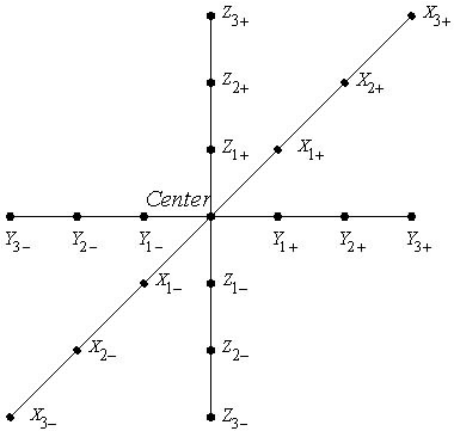

Fig. 4. Robot state-space

An action, a_t, consists of a robot movement over its X, Y or Z axes. The robot starts a shaking policy from the $s_{(Center)_t}$ state. From $s_{(Center)_t}$ it can move to any of the other 18 states. The distance (denoted as "the amplitude") between any two close states is set *a priori* to performing a shaking policy (e.g., distances between $s_{(Z_{3-})_t}$ and $s_{(Z_{2-})_t}$ or between $s_{(Center)_t}$ and $s_{(Z_{1+})_t}$ are 30 mm). From a robot state other than $s_{(Center)_t}$, an action performed by the robot is limited to be performed symmetrically or the robot can move back to $s_{(Center)_t}$ (e.g., from state $s_{(X_{2-})_t}$ the robot can move only to $s_{(Center)_t}$ or to $s_{(X_{2+})_t}$). In addition to the dependency of an action on the predefined amplitude, two speed values are defined, (1000 and 1500 mm / s). This doubles the number of possible actions the robot can take resulting in 108 possible actions.

Let a policy P be a set of 100 state-action pairs, $\{s_t, a_t\}$ and the performance measure be $Z(P_i)$. Performance measure, $Z(P_i)$ $i \in \{1, 2\}$, includes: (i) $Z(P_1)$ - average time to complete (or partially complete) emptying the contents of a bag, and (ii) $Z(P_2)$ - human intervention rate, i.e., a measure that represents the percentage of human interventions out of the total number of learning trails. Human operator (HO) collaboration is triggered when robot learning is slow. $Z(P_2)$ summarizes HO collaboration (the lower it is, and the more autonomous the robot is).

Robot learning experience is achieved through direct experience with the environment according to rewards based on the number of items fell from a bag after performing a shaking policy. Its value is linearly depends on the number of falling items; the robot gets a numerical value of 20 for every item fell. If no items fall, the robot is "punished" by getting no reward.

Experimental setup contains a Motoman UP-6 fixed-arm robot overheads an inspection surface (Fig. 5a). A dedicated gripper was designed enabling a smooth slipping for grasping a bag. At the beginning of each learning trial performed, the robot grasps and lifts a bag containing five wooden cubes (Fig. 5b), then it performs a shaking policy.

(a) Robot and inspection surface (b) Plastic bag and cubes

Fig. 5. Experimental setup

The UP-6 robot has no *a priori* knowledge in its initial learning stages; thus, in addition to interacting with the environment and getting rewards / punishments, policy adjustments are provided by the HO through an interface (Fig. 6).

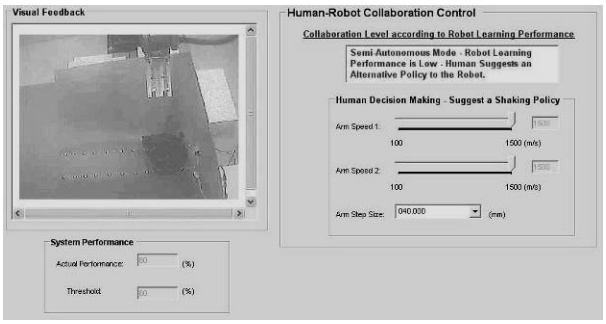

Fig. 6. Human interface

The interface views and controls consists of: (i) real-time visual feedback captured from a web-camera located over the robotic scene; (ii) system learning performance reporting; (iii) system mode reporting - autonomous or semi-autonomous, and (iv) human decision making control - when asked to intervene, the HO can determine robot shaking amplitude and speeds. The robot learning function acquires the suggested parameters and performs a new shaking policy.

B. Experiments

Two experiments were performed. In the *first* experiment $CQ(\lambda)$ -learning is employed. 75 learning trials were separated into three stages: (i) *training* - during the first ten runs the robot performs shaking policies autonomously. The initial shaking parameters were set to amplitude of 30 mm, and to speeds of 1000 and 1500 mm / s; (ii) *collaboration* - in this stage consists of 40 shaking policies human intervention triggering was allowed, based on the system learning performance. In this stage the human can adjust shaking

policies parameters at the range of 10 to 50 mm for the amplitude and speeds in the range of 100 to 1500 mm / s, and (iii) *testing* - for measuring the efficiency of the human collaboration, the robot performed 25 policies using the original shaking parameters defined in the training stage. No human intervention was allowed in stages (i) and (iii).

To compare the $CQ(\lambda)$ with the $Q(\lambda)$ -learning algorithm, a *second* experiment was set. The experiment consisted of 25 learning trials where the system learned according to the standard $Q(\lambda)$ -learning algorithm with no human intervention.

For both experiments, the first trial consists of a random shaking policy over its X , Y or Z axes. The system acquired rewards achieved by interaction with the environment while $\gamma = 0.9$, $\lambda = 0.5$, and $\alpha = 0.05$ were set. To balance between exploration and exploitation (e.g., [35]), ε -greedy action selection with $\varepsilon = 0.1$ was used.

IV. Experimental Results

For the *first* experiment, results of measuring $z(P_1)$, the average time to complete (or partially complete) emptying the contents of a bag for the *training* stage was 15.62 s (STD: 2.41). For the *collaboration* stage, the average $z(P_1)$ was measured as 10.46 (STD: 11.82). At the *testing* stage, where the same shaking parameters as in the *training* stage were set, i.e., amplitude of 30 mm, and speeds of 1000 and 1500 mm / s, average $z(P_1)$ was measured as 12.55 (STD: 3.99).

From a reward perspective, ranging at the scale of 0 to 100, at the *training* stage, in which only four out of the ten policies revealed positive rewards, the average reward achieved was 32 (STD: 47.33). At the *collaboration* stage, the average reward was 94.5 (STD: 22.18), and for the *testing* stage the average reward achieved was 93.6 (STD: 21.8).

Human intervention rate, $z(P_2)$, measured during the *collaboration* mode was 32.5% whereas all system requests for collaboration occurred continuously during the first runs on this stage.

A significant improvement for the *collaboration* stage was achieved in comparison with the *training* stage while measuring $z(P_1)$ (33% with high average reward of 94.5). Importantly, at the *testing* stage where the robot functioned autonomously, with no human intervention, an improvement of 20% was achieved with high average reward of 93.6.

For the *second* experiment, the average result of measuring $z(P_1)$ was 11.17 s (STD: 3.17) and the average reward achieved was 50.4 (STD: 48.69).

While comparing the results achieved at the *first* experiment ($CQ(\lambda)$ -learning) with the second experiment ($Q(\lambda)$ -learning) over 25 learning trials, indicated no significant difference in $z(P_1)$, whereas for the $CQ(\lambda)$ -learning, an improvement of 21.25% in the average reward was achieved.

Intuitionally, vertical shaking would be best, but experiments determine that policies of shaking most of the time over the Y axis (Fig. 4) with small number of actions over the X axis were the most effective. This policy caused the bag sometimes to become entangled, also due to the fact that most of the plastic bag weight is concentrated most of the time in one place. Possible explanation might be the type of the plastic bag knot; pulling it side ways makes it lose faster. Further, hypothetically, if a human would require shaking the bag, it could have seen visually the servo feedback to determine the optimal time to pull it up in a horizontal strategy, an ability that a robot does not have.

V. Conclusions

The proposed RL-based human-robot learning collaboration decision-making method is targeted for a complex system. The robot makes a decision whether to learn the task autonomously or ask for human intervention. The approach is aimed to integrate user instructions into an adaptive and flexible control framework and to adjust control policies on-line. To achieve this, user commands at different levels of abstraction are integrated into an autonomous learning system. Based on its learning performance, the robot switches between fully autonomous operation and the integration of human commands.

The $CQ(\lambda)$ learning algorithm, accelerates learning in systems composed of several learning agents or systems designed for human-robot interaction overcoming the main criticism of the reinforcement learning approach, i.e., long training periods.

Future work will include the design and implementation of an extended collaboration that include human intervention using refined or rough intervention policies, robot autonomy, and pure human control. Additionally, robot autonomy will be expanded. One approach might be to provide an external support resource such as another robot for assisting opening a bag in the grasping stage.

References

[1] J. Bukchin, R. Luquer, and A. Shtub, "Learning in tele-operations", *IIE Trans.*, 2002, vol. 34, no. 3, pp. 245-252.

[2] J. W. Crandall, M. A. Goodrich, D. R. Olsen, and C. W. Nielsen, "Validating Human-Robot Interaction Schemes in Multi-Tasking Environments," *IEEE Trans. on Systems, Man, and Cybernetics Part A: Systems and Humans, Special Issue on Human-Robot Interaction,* July 2005, vol. 35, no. 4, pp.438-449.

[3] A. M. Steinfeld, T. W. Fong, D. Kaber, M. Lewis, J. Scholtz, A. Schultz, and M. Goodrich, "Common Metrics for Human-Robot Interaction," *Human-Robot Interaction Conf.*, ACM, March, 2006.

[4] J. A. Adams, "Critical Considerations for Human-Robot Interface Development," *AAAI Fall Sym.: Human Robot Interaction Technical Report FS-02-03*, Nov. 2002, pp.1-8.

[5] M. T. Rosenstein, A. H. Fagg, S. Ou, and R. A. Grupen, "User intentions funneled through a human-robot interface," *Proc. of the 10th Int. Conf. on Intelligent User Interfaces,* 2005, 257-259.

[6] H. A. Yanco, M. Baker, R. Casey, A. Chanler, M. Desai, D. Hestand, B. Keyes, and P. Thoren, "Improving human-robot interaction for remote robot operation," Robot Competition and

Exhibition Abstract, *National Conf. on Artificial Intelligence (AAAI-05)*, July 2005.

[7] L. P. Kaelbling, M. L. Littman, and A. W. Moore, "Reinforcement learning: a survey," *Journal of Artificial Intelligence Research*, 1996, vol. 4, pp. 237-285.

[8] W. D. Smart, *Making Reinforcement Learning Work on Real Robots*, Ph.D. Dissertation, Brown University, 2002.

[9] C. Ribeiro, "Reinforcement learning agents," *Artificial Intelligence Review*, 2002, vol. 17, no. 3, pp. 223-250.

[10] A. Lockerd and C. Breazeal, "Tutelage and socially guided robot learning," *Proc. of IEEE/RSJ Int. Conf. on Intelligent Robots and Systems*, 2004, Sendai, Japan.

[11] Y. Wang, M. Huber, V. N. Papudesi, and D. J. Cook, "User-guided reinforcement learning of robot assistive tasks for an intelligent environment," *Proc. of the IEEE/RJS Int. Conf. on Intelligent Robots and Systems*, 2003.

[12] V. N. Papudesi and M. Huber, "Learning from reinforcement and advice using composite reward functions," *Proc. of the 16th Int. FLAIRS Conf.*, 2003, pp. 361-365, St. Augustine, FL.

[13] V. N. Papudesi, Y. Wang, M. Huber, and D. J. Cook, "Integrating user commands and autonomous task performance in a reinforcement learning framework," *AAAI Spring Sym. on Human Interaction with Autonomous Systems in Complex Environments*, 2003, Stanford University, CA.

[14] K. Driessens and S. Džeroski, "Integrating guidance into relational reinforcement learning," *Machine Learning*, 2004, vol. 57, pp. 271-304.

[15] L. Mihalkova and R. Mooney, "Using active relocation to aid reinforcement," *Proc. of the 19th Int. FLAIRS Conf.*, May 2006, Melbourne Beach, Florida, to be published.

[16] D. Gu and H. Hu, "Fuzzy multi-agent cooperative Q-learning," *Proc. of IEEE Int. Conf. on Information Acquisition*, 2005, Hong Kong, China, pp. 193-197.

[17] R. M. Kretchmar, "Parallel reinforcement learning," *The 6th World Conf. on Systemics, Cybernetics, and Informatics*, 2002.

[18] M. J. Matarić, "Reinforcement learning in the multi-robot domain," *Autonomous Robots*, Kluwer Academic Publishers, 1997, vol. 4, pp. 73-83.

[19] W. Zhu and S. Levinson, "Vision-based reinforcement learning for robot navigation," *Proc. of the Int. Joint Conf. on Neural Networks*, 2001, vol. 2, pp. 1025-1030, Washington DC.

[20] P. Kui-Hong, J. Jun, and K. Jong-Hwan, "Stabilization of biped robot based on two mode Q-learning," *Proc. of the 2nd Int. Conf. on Autonomous Robots and Agents*, 2004, New Zealand.

[21] A. F. Massoud and L. Caro, "Fuzzy neural network implementation of $Q(\lambda)$ for mobile robots," *WSEAS Trans. on Systems*, 2004, vol. 3, no. 1.

[22] Y. Dahmani and A. Benyettou, "Seek of an optimal way by Q-learning," *Journal of Computer Science*, 2005, vol. 1, no. 1, pp. 28-30.

[23] R. Broadbent and T. Peterson, "Robot learning in partially observable, noisy, continuous worlds," *Proc. of the 2005 IEEE Intl. Conf. on Robotics and Automation*, 2005, Barcelona, Spain.

[24] T. Martínez-Marín and T. Duckett, "Fast reinforcement learning for vision-guided mobile robots," *Proc. of the 2005 IEEE Int. Conf. on Robotics and Automation*, 2005, Barcelona, Spain.

[25] C. J. C. H. Watkins, *Learning from Delayed Rewards*, Ph.D. Dissertation, Cambridge University, 1989.

[26] W. D. Smart and L. Kaelbling, "Practical reinforcement learning in continuous spaces," *Proc. of the 17th Int. Conf. on Machine Learning*, 2002.

[27] R. Bellman and R. Kalaba, *Dynamic Programming and Modern Control Theory*, NY: Academic Press Inc., 1965.

[28] C. J. C. H. Watkins and P. Dayan, "Q-learning," *Machine Learning*, 1992, vol. 8, pp. 279-292.

[29] J. Peng and R. Williams, "Incremental multi-step Q-learning," *Machine Learning*, 1996, vol. 22, no. 1-3, pp. 283-290.

[30] P. Y. Glorennec, "Reinforcement Learning: an overview," *European Sym. on Intelligent Techniques*, Aachen, Germany, 2000.

[31] S. Natarajan and P. Tadepalli, "Dynamic preferences in multi-criteria reinforcement learning," *Proc. of the 22nd Int. Conf. on Machine Learning (ICML 2005)*, Bonn, Germany, 2005.

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

[33] U. Kartoun, H. Stern, Y. Edan, C. Feied, J. Handler, M. Smith, and M. Gillam, "Collaborative $Q(\lambda)$ Reinforcement Learning Algorithm - A Promising Robot Learning Framework," *IASTED Int. Conf. on Robotics and Applications*, 2005, U.S.A.

[34] A. K. MackWorth, D. Poole, and R. G. Goebel, *Computational Intelligence: A Logical Approach*, Oxford University Press, 1998.

[35] M. Guo, Y. Liu, and J. Malec, "A new Q-learning algorithm based on the metropolis criterion," *IEEE Trans. on Systems, Man, and Cybernetics - Part B: Cybernetics*, 2004, vol. 34, no. 5, pp. 2140-2143.