September 22, 2006

Contact: Uri Kartoun (kartoun@yahoo.com)

Visit: http://www.compactech.com/kartoun

This document contains the following papers:

- Kartoun U., Stern H., Edan Y. 2006. Human-Robot Collaborative Learning System for Inspection. *IEEE International Conference on Systems, Man, and Cybernetics*. Oct. 8 - Oct. 11, 2006, Taipei, Taiwan. Finalist (5 papers) for the Best Student Paper Competition.

- Kartoun U., Stern H., Edan Y., Feied C., Handler J., Smith M. and Gillam M. 2006. Vision-Based Autonomous Robot Self-Docking and Recharging. *ISORA 2006 11th International Symposium on Robotics and Applications (Robotics Track) at World Automation Congress (WAC 2006)*. July 24-27, Budapest, Hungary.

- Kartoun U., Stern H. and Edan Y. 2006. Bag Classification Using Support Vector Machines. Applied Soft Computing Technologies: The Challenge of Complexity Series: Advances in Soft Computing, Springer Berlin / Heidelberg, ISBN: 978-3-540-31649-7, pp. 665-674.

- Stern H., Kartoun U., Shmilovici A., An Expert System for Surveillance Picture Understanding, *Proceedings of NATO-ASI, Data Fusion for Situation Monitoring, Incident Detection, Alert and Response Management*, August 18-29, Yerevan, Armenia. - to be published in a collective volume in NATO Science series, IOS Press, 2005, 14p. (a derivative of conference paper no. 22).

- Kartoun U., Stern H., Edan Y., Feied C., Handler J., Smith M. and Gillam M. 2005. Collaborative Q(lambda) Reinforcement Learning Algorithm - A Promising Robot Learning Framework. *IASTED International Conference on Robotics and Applications (RA 2005)*. October 31 - November 2, Cambridge, U.S.A.

- Edan Y., Kartoun U. and Stern H. 2004. Cooperative Human-Robot Learning System using a Virtual Reality Telerobotic Interface. *Conference on Advances in Internet Technologies and Applications*, Purdue University, West Lafayette, Indiana, U.S.A. (Also submitted to the 'Advances in Soft Computing Series' Journal, Springer Verlag).

- Kartoun U., Stern H. and Edan Y. 2004. Virtual Reality Telerobotic System. *e-ENGDET 2004 4th International Conference on e-Engineering and Digital Enterprise Technology*, Leeds Metropolitan University Yorkshire, U.K.

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.

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.

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)

"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}) :
        $Q_i(s_{i_{t+1}}, a_{i_{t+1}}) \leftarrow \max_{i \in N}[Q(s_{i_t}, a_{i_t}) + \alpha_{i_t}\delta_{i_t}e_{i_t}(s_{i_t}, a_{i_t})]$
      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.

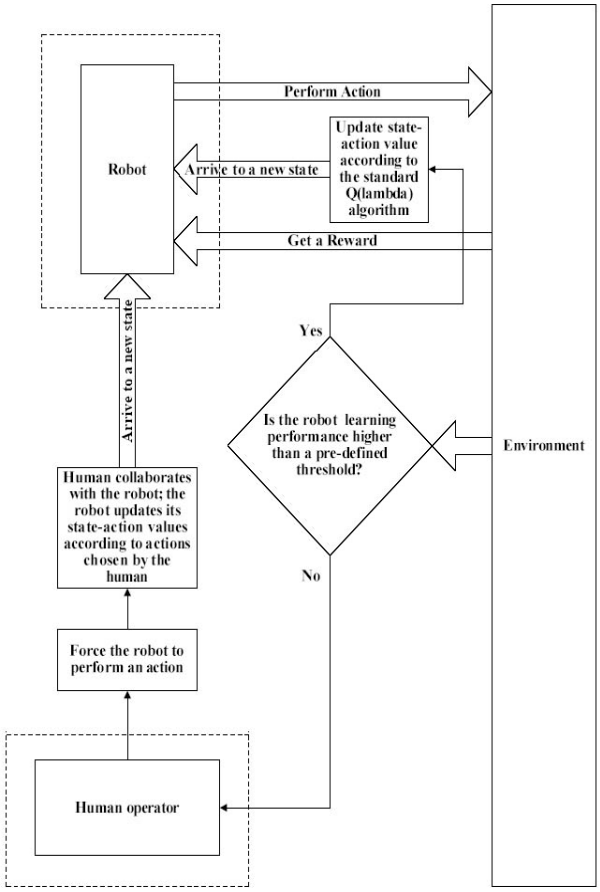


Fig. 2. Robot and a human operator $CQ(\lambda)$ -learning

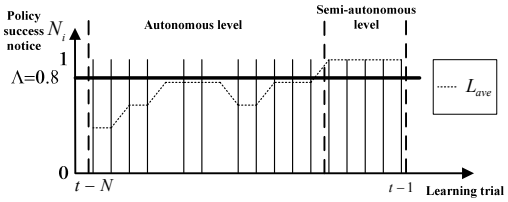


Fig. 3. Example of learning performances

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

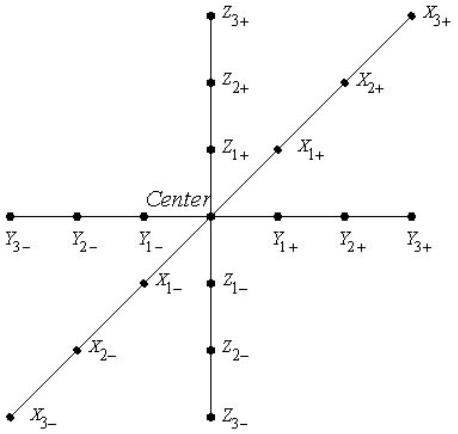


Fig. 4. Robot state-space

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).

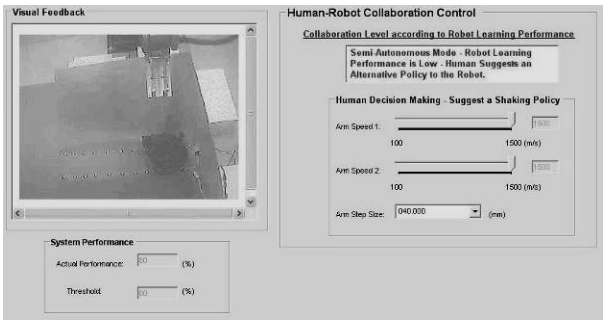


Fig. 6. Human interface

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.

Vision-Based Autonomous Robot Self-Docking and Recharging

Uri Kartoun, Ben-Gurion University of the Negev, Israel, kartoun@bgu.ac.il
Helman Stern, Ben-Gurion University of the Negev, Israel, helman@bgu.ac.il
Yael Edan, Ben-Gurion University of the Negev, Israel, yael@bgu.ac.il
Craig Feied, Washington Hospital Center, U.S.A., cfeied@ncemi.org
Jonathan Handler, Washington Hospital Center, U.S.A., jah505@northwestern.edu
Mark Smith, Washington Hospital Center, U.S.A., msmith@ncemi.org
Michael Gillam, Washington Hospital Center, U.S.A., gillam@gmail.com

ABSTRACT

This paper presents a method for autonomous recharging of a mobile robot, a necessity for achieving long-term robotic activity without human intervention. A recharging station is designed consisting of a stationary docking station and a docking mechanism mounted to an ER-1 Evolution Robotics robot. The docking station and docking mechanism serve as a dual-power source, providing a mechanical and electrical connection between the recharging system of the robot and a laptop placed on it. Docking strategy algorithms use vision based navigation. The result is a significantly low-cost, high-entrance angle tolerant system. Iterative improvements to the system, to resist environmental perturbations and implement obstacle avoidance, ultimately resulted in a docking success rate of 100 percent over 50 trials.

KEYWORDS: Robotic mechanisms, robot control, autonomous navigation, machine vision

1. INTRODUCTION

Mobile robots are being designed to interact increasingly with human environments, working with and around humans on a daily basis. To be considered of any use, these robots must exhibit some form of self-sustainability. In addition to being robust in their physical design as well as control methodology, such robots must be capable of long-term autonomy. Energy is of great concern, and without it the robot will become immobilized and useless [1].

A custom recharging station designed for ActivMedia Pioneer robots is described in [2]. The station emits an infrared signal that is detected by a ring of seven infrared receivers located at the back of the robot. The robot docks into the recharging station by backing up so that the charging pins make contact with the recharging station. Aside from the Pioneer-based recharging system, there are other efforts under way to develop recharging capabilities to achieve long-term autonomous mobile robot control. Examples include [3] and [4], which describe the same recharging system with increased functionality. 94.26 percent docking success is addressed in [4], which describes the results of repetitive docking over the course of a week using a robot similar to the Pioneer. Commercially available robots examples include: (i) the Helpmate robot [5] having been installed in hospitals worldwide, and (ii) a cleaning machine equipped with a Siemens Corporation navigation system [6]. Chips, an autonomous robotic technology XR4000-based was implemented on three robots in museums providing tours and covering travel distance of hundreds of kilometers [7]. In order to achieve true self-reliance, any of the three robots must be able to recharge themselves when necessary. This was accomplished by using a simple 3D fiducial, aligned with an electrical outlet that provided both translational and rotational position feedback. Using this marker, the robots had demonstrated reliable positioning to an accuracy of 1.5 mm using visual position servoing. The entire docking process, including moving over a

distance of four meters to the outlet and then fine-servoing for the insertion process, took less than three minutes.

97 percent success rate for docking within average time of 30.5 seconds using a Pioneer robot was achieved by [1, 8]. The docking station is a freestanding passive device with two degrees of freedom (DOF), providing the necessary compliance for multiple docking conditions. The robot docking mechanism is a passive one DOF assembly attached to the back of the robot. A contact switch is connected to the underside of the docking mechanism, controlling the power connection to the robot's batteries. An IR-LED detector is mounted to the top deck of the robot directed toward its back.

Our focus here is on the development of long-term autonomous capabilities that enable an ER-1 robot to recharge itself without assistance. We developed a recharging station that allows the robot to recharge autonomously. The recharging station's physical design is described in two parts; (i) the docking station, and (ii) the robot-laptop docking mechanism. Our recharging and docking strategy is presented, including an explanation of the tasks developed to control the docking navigation procedures. The system can be easily adapted to different indoor environments, as well as multiple robots.

The physical design is presented in section 2. Recharging and docking strategy describing the navigation control and vision algorithms is presented in section 3. In section 4 experiments and results are described. The paper ends in section 5 with conclusions and some directions for future work.

2. PHYSICAL DESIGN

Our system is based around an ER-1 Evolution Robotics mobile robot. This robot is a three-wheeled platform equipped with an IBM ThinkPad laptop and a Logitech QuickCam Pro 4000 color camera supporting autonomous control capabilities. The ER-1 rechargeable power module equipped with a 12V, 5.4A battery can keep it running for 2.5 hours. In addition power should be supplied to the laptop battery.

The design presented consists of two parts: a docking station (Figure 1(a)) and a robot-laptop docking mechanism (Figure 1(b)). Design factors include a large tolerance window for docking to increase the probability of docking success over a wide range of robot entry angles. The robot may enter the docking station with a high probability of success within a total entry angle of 45°. Minimal modification to the ER-1 robot is desired, reducing any interference with current robot capabilities.

(a) Docking station *(b) Robot-laptop docking mechanism*

Figure 1. System physical design

2.1 Docking Station

The docking station (Figure 1(a)) is a stationary fixture that provides a connection point for the robot-laptop's docking mechanism described in section 2.2. The ER-1 charger as well as the laptop power supplier are connected to the docking station providing the necessary power. The docking station is LEGO-based and built of cubes assembled into two surfaces. Four aluminum surfaces interlace into the structure. The upper two panels are connected to a 16VDC adapter for recharging the robot laptop. The two lower panels are connected to a 12VDC power charger for recharging the ER-1 battery.

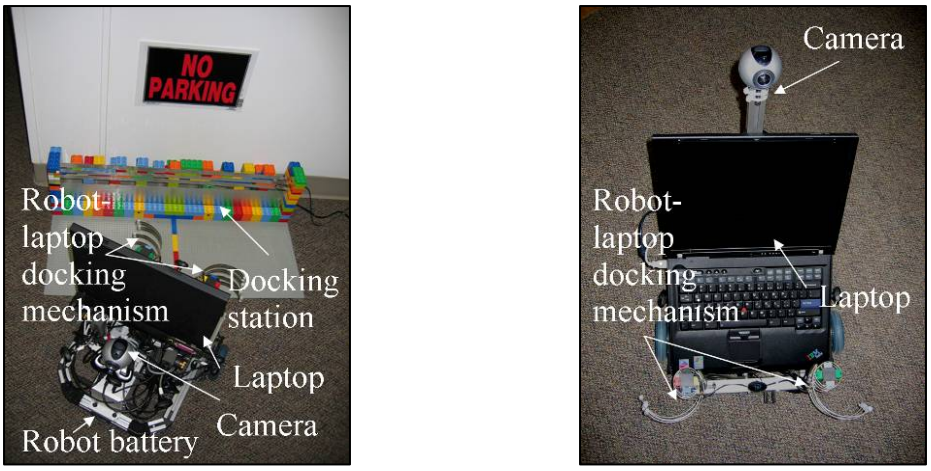
2.2 Robot-Laptop Docking Mechanism

The robot-laptop docking mechanism (Figure 1(b)) is mounted to the ER-1 robot. The mechanism consists of two identical four flexible spring structures located on the left and on the right of the back of the robot. The upper two springs of each structure are connected to the laptop and the lower two are connected to the robot battery. Encounter of any one of the two structures with the docking station results in a successful docking and beginning of recharging.

3. RECHARGING AND DOCKING STRATEGY

3.1 Navigation Control

The system was tested using two robotic configurations, each one was tested for a navigation task in a different environment: (i) "Docking Room I" - the robot-laptop docking mechanism was mounted opposite to the robot motion while the camera was facing toward (Figure 2(a)), and (ii) "Docking Room II" - both the robot-laptop docking mechanism and the camera were mounted toward the robot motion (Figure 2(b)).



(a) Robot-laptop docking mechanism mounted opposite to the robot motion

(b) Robot-laptop docking mechanism mounted toward the robot motion

Figure 2. Overall views over the docking system and the ER-1 robot

The control is built using several tasks: (i) the *Drive-Toward-Object* task makes the ER-1 to track an image while moving and correcting its location to drive toward this image; (ii) the *Turn-To-Object* task operates in three modes: (a) search; (b) hover, and (c) track. In this mode the robot tries to turn toward the target to face it directly; (iii) the *Obstacle-Avoidance* inhibits the *Drive-Toward-Object* task in order to directly control the motors and stop the robot when the range between the robot and an object (*e.g.*, human) is sufficiently small, and (iv) the *Proceed-And-Scan* task triggers the robot to move short distance forward (*e.g.*, 5 cm) and scan its surroundings (*e.g.*, rotate 5° to the left and to the right) repeatedly. This task repeats itself when the robot can not identify any landmarks of an image till the target is found and assures that in cases of low

target visibility or camera trembling due to nonuniform driving the robot arrives to its destination. Flowchart describing the navigation strategy is shown in Figure 3.

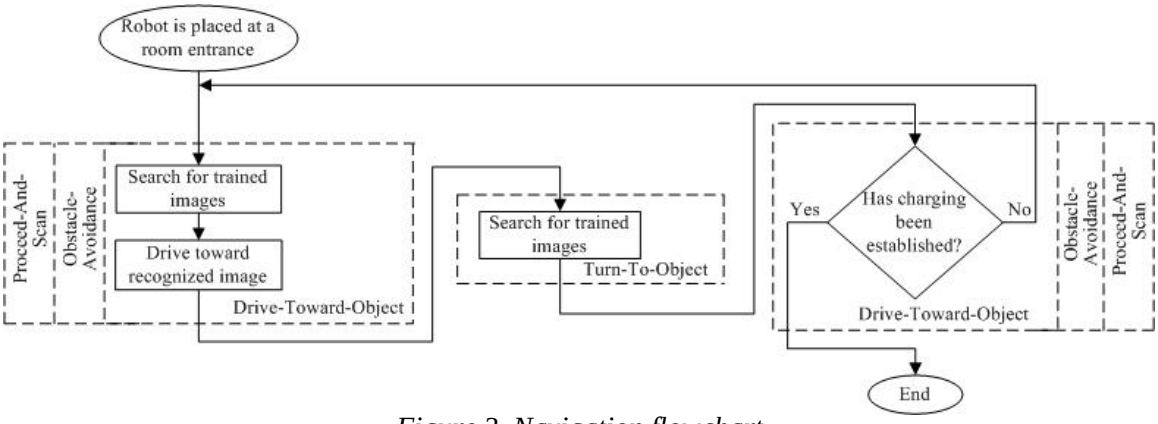


Figure 3. Navigation flowchart

3.2 Object Recognition

The object recognition algorithm is based on extracting salient features from an image or an object [9, 10]. Local descriptors [*e.g.*, 11] are commonly employed in a number of real-world applications [*e.g.*, 12] because they can be computed in real-time, resistant to partial occlusions, and are relatively insensitive to changes in viewpoint.

It is stated in [13] that "Hessian-Laplace regions [11, 14] are invariant to rotation and scale changes. Points are localized in space at the local maxima of the Hessian determinant and in scale at the local maxima of the Laplacian-of-Gaussians. This detector is similar to the approach described in [10] which localizes points at local scale-space maxima of the difference-of-Gaussian (DoG). Both approaches detect the same blob-like structures. However, Hessian-Laplace obtains higher localization accuracy in scalespace, as DoG also responds to edges and detection is unstable in this case. The scale selection accuracy is also higher than in the case of the Harris-Laplace detector".

The main strength of the object recognition algorithm used here [9, 10] lies in its ability to provide reliable recognition in realistic environments where lighting conditions change dramatically. It is stated in [9] that "the time it takes to process and match each training or recognition image is about 0.8 seconds on a Pentium III 600MHz computer. About half of the time is devoted to image acquisition and feature detection, while the remaining time is used for all aspects of matching and model formation". The algorithm consists of two stages: (i) training - accomplished by capturing an image of an object or a scene, and (ii) recognition of local features for each image the robot encounters. A small subset of those features and their interrelation identifies a pose and a distance of the image.

Each feature is uniquely described by the texture of a small window of pixels around it [10]. The model of an image consists of the coordinates of all these features along with each feature's texture description. When the algorithm attempts to recognize objects in a new image, it first finds features in the new image. It then tries to associate features in the new image with all the features in the database of models. This matching is based on the similarity of the feature texture. If eight or more features in the new image have good matches to the same database model, that potential image match is refined. The refinement process involves the computation of an affine transform between the new image and the database model, so that the relative position of the features is preserved through the transformation. The algorithm outputs all object matches for which the optimized affine transform results in a small root-mean-square pixel error between the features found in the new image, and the corresponding affine-transformed features of the original model.

As stated in [9], the similarity transform between two images gives the mapping of a model point $[x\ y]$ to an image point $[u\ v]$ in terms of an image scaling, s , an image rotation, θ, and an image translation, $[t_x\ t_y]$:

$$\begin{bmatrix} u \\ v \end{bmatrix} = \begin{bmatrix} s\cos\theta & -s\sin\theta \\ s\sin\theta & s\cos\theta \end{bmatrix} \begin{bmatrix} x \\ y \end{bmatrix} + \begin{bmatrix} t_x \\ t_y \end{bmatrix} \quad (1)$$

which can be written as a linear form collecting the unknown similarity transform parameters into a vector:

$$\begin{bmatrix} x & -y & 1 & 0 \\ y & x & 0 & 1 \\ & & \cdots & \end{bmatrix} \begin{bmatrix} s\cos\theta \\ s\sin\theta \\ t_x \\ t_y \end{bmatrix} = \begin{bmatrix} u \\ v \\ \vdots \end{bmatrix}. \quad (2)$$

This linear system can be written as

$$\mathrm{Ax} = \mathrm{b} \quad (3)$$

The least-squares solution for the parameters x are determined by solving the corresponding normal equations:

$$x = [A^T A]^{-1} A^T b \quad (4)$$

which minimizes the sum of squares of the distances from the projected model locations to the corresponding image locations.

By using the solution of x the average error, e between each projected model feature and image feature can be estimated:

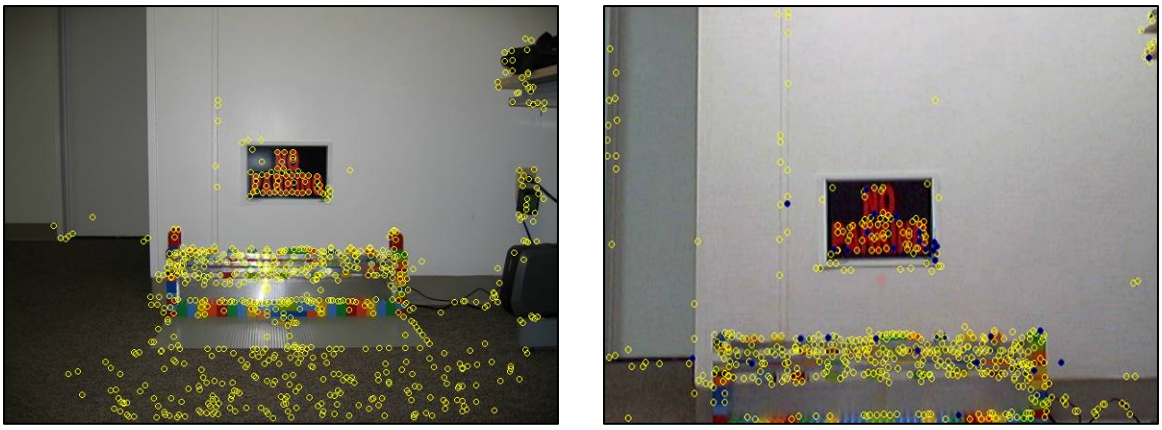
$$e = \sqrt{\frac{2 \parallel A\mathrm{x} - \mathrm{b} \parallel^2}{r - 4}} \quad (5)$$

where r is the number of rows in matrix A. As each new training image arrives, it is matched to the previous model views. The result is that training images that are closely matched by a similarity transform are clustered into model views (number of views is determined dynamically) that combine their features for increased robustness. Otherwise, the training images form new views in which features are linked to their neighbors. The result is that additional training images continue to contribute to the robustness of the system by modeling feature variation without leading to a continuous increase in the number of view models [9].

4. EXPERIMENTS AND RESULTS

Test conditions were developed, consisting of a closed environment for the robot to maneuver and two recharging tasks. When the ER-1 is placed at anyone of “Docking Room I” or “Docking Room II” entrances (Figure 5) it automatically starts its navigation. Performance is based on the success or failure of the docking and recharging operation.

The object recognition algorithms were integrated with the navigation control strategies. The image databases were trained using three different images for each one of the robot-laptop mechanism configurations applied in the docking rooms (Figure 2). An example for features extracted from the docking station image is shown in Figure 4. Figure 4(a) shows the corresponding 724 features (white spheres) overlaid extracted from the docking station image. Figure 4(b) shows the corresponding features overlaid (white spheres) as well as the set of matched features (dark spheres) extracted from an image captured by the camera when the robot moves toward the docking station (28 matched features and 540 features in total). It is noted the height of the robot is taken into account when the images are collected; the height of the camera placed on the robot was set to 65 cm above the floor and was constant during the experiments, both during training and navigation. If it is desired to place the camera in a different height, the training stage should be repeated.

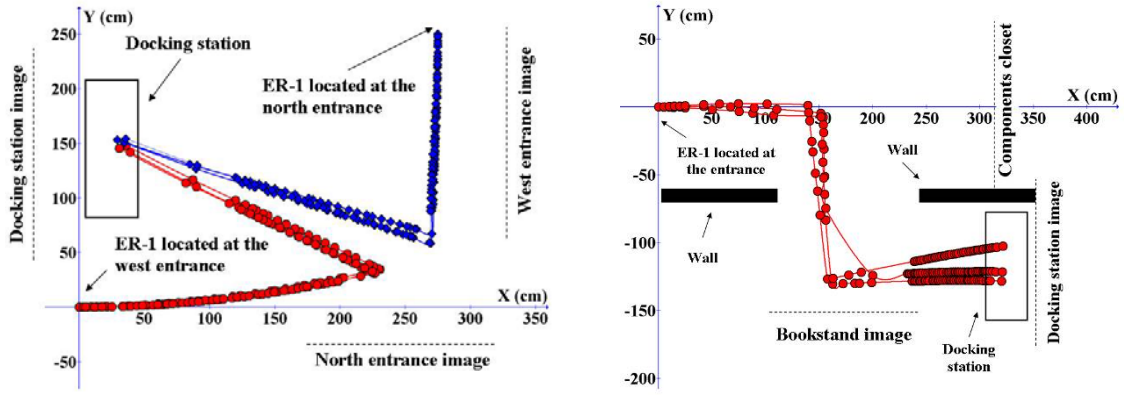


(a) Features extracted from the original docking station image

(b) Features extracted when the ER-1 moves toward the docking station image

Figure 4. Example for features extracted from the docking station image

To determine the performance capabilities of our system 50 trials were conducted for each one of the docking rooms. During "Docking Room I" experiments, in 25 of the trials the robot was placed at the west entrance and in the other 25 trials the robot was placed at the north entrance. Figure 5(a) describes "Docking Room I" structure and shows odometry results of six sample trials (at three of the trials noted as spheres, the robot starts the docking procedure from the west entrance and at the other three trials noted as diamonds it starts from north). Results of the 50 trials showed a 98 percent success rate for mechanical docking, and a 96 percent success rate for the electrical recharging operation. We distinguish between mechanical docking and electrical recharging separately due to the differences involved to complete each operation. Mechanical docking is based on whether or not the robot entered or missed the docking station. The electrical docking indicates whether a recharging procedure has been initiated. Both mechanical and electrical dockings are necessary to allow power to flow to the batteries from the chargers. The one mechanical docking failure in "Docking Room I" during the 50 trials was due the reason that a person stood in the robot's path during the docking procedure and the robot activity was distracted. The two electrical recharging failures were due to unseen dirt and dust that covered the docking station's aluminum surfaces and caused nonconductivity between the contacts.



(a) "Docking Room I"

(b) "Docking Room II"

Figure 5. Top view over the robotic environments and odometry results

In order to improve our dock success rate achieved at the "Docking Room I" experiment, the *Proceed-And-Scan* task was integrated with "Docking Room II" navigation algorithm.

Furthermore, another capability of recognizing whether docking is successful was added to the system; the robot keeps measuring the battery percentage left. An increase of at least three percent in the robot battery level indicates that charging has been established. Figure 5(b) describes "Docking Room II" structure and shows odometry results of three sample trials noted as spheres. Results of 50 trials showed a 100 percent success rate for both mechanical and electrical dockings.

Our tests showed that the recharging and docking control algorithms were capable of positioning the robot with an accuracy of approximately 15° relative to the docking station in both docking room experiments. This falls within the allowable tolerance zone of the docking station. The time associated with the entire docking procedure was measured for each trial. This is the time from which the robot enters the room to the point it successfully docks mechanically and electrical contact is initiated. We determined that in "Docking Room I", the average docking time spanned approximately 57.4 seconds for the west entrance with a standard deviation of 2.4 within an average distance of 400.3 cm and 51.6 seconds for the north entrance with a standard deviation of 3.5 within an average distance of 410 cm. A best case scenario resulted in 55.3 seconds for the west entrance and 46.2 seconds for the north entrance dock time. Our worst case trial in "Docking Room I" experiment where the ER-1 started from the west entrance resulted in a 66.4 seconds dock time due to the reason that a human passed through the robot trajectory and caused it to wait till she left the room. In "Docking Room II", the average docking time spanned approximately 85.23 seconds with a standard deviation of 20.84 within an average distance of 768.2 cm. A best case scenario resulted in 60 seconds. Our worst case trial in this experiment resulted in a 174.52 seconds dock time due to the reason that several curious people gathered around the robot surroundings and blocked its way occasionally.

5. CONCLUSIONS AND FUTURE WORK

We successfully developed a recharging system and navigation control algorithms that have been integrated with an ER-1 robot, thereby providing the capabilities necessary for long-term autonomous experimentation. The docking station concept presented is not limited to any specific robot. Our recharging system was tested resulting in a 100 percent mechanical and electrical success rate for docking, within less than 1.5 min average dock time for each dock. Although the robot met obstacles during its navigation, the algorithm coped with all of them and worked continuously. With comparison with the reviewed docking stations mentioned our system allows considerably high degree of entrance tolerance (45°). Although the docking strategy described in [1] resulted in better average docking times, our simply designed system provided a remarkable percent success of mechanical and electrical docking rate within short dock times while tested in environments with external interruptions (*e.g.*, people). In addition, no double recharging system (i.e., recharging a computer in addition to a robot battery) is discussed in literature. We note that our system achieves high dock rate success in comparison with other systems due to: (i) the unique hardware we installed on the back of the robot, which uses two flexible spring structures as part of its robot-laptop docking mechanism instead of only one, a fact that reduces the probability of mechanical docking failures, and (ii) a design that allows relatively large positioning errors due to its relatively large size. Additional trials are planned to prove the robustness and repeatability of the system over extended periods of time, under various conditions and with different robots.

6. ACKNOWLEDGEMENTS

This work is 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. The assistance of Mr. Robert Irving from the Institute for Medical Informatics is greatly appreciated.

7. REFERNCES

[1] M.C. Silverman, D. Nies, B. Jung, and G.S. Sukhatme, "Staying alive: a docking station for autonomous robot recharging," *IEEE International Conference on Robotics and Automation*, vol. 1, pp. 1050-1055, Washington DC, 2002.

[2] F. Michaud, J. Audet, D. Letourneau, L. Lussier, C. Theberge-Turmel, and S. Caron, "Autonomous robot that uses symbol recognition and artificial emotion to attend the AAAI conference, " *AAAI Robot Competition Technical Report*, 2000.

[3] H. Yasushi and Y. Shin'ichi, "Robust navigation and battery re-charging system for long-term activity of autonomous mobile robot," *Proceedings of the 9th International Conference on Advanced Robotics*, pp. 297-302, 1999.

[4] H. Yasushi and Y. Shin'ichi, "A first stage experiment of long term activity of autonomous mobile robot - result of repetitive base-docking over a week," *Proceedings of The 7th International Symposium on Experimental Robotics (ISER2000)*, pp. 235-244, 2000.

[5] S. King and C. Weiman, "Helpmate autonomous mobile navigation system," *Proceedings of SPIE Conference on Mobile Robots*, vol. 2352, pp. 190-198, Boston, MA, 1990.

[6] H. Endres, W. Feiten and G. Lawitzky, "Field test of a navigation system: autonomous cleaning in supermarkets," *Proceedings of the 1998 IEEE International conference onRobotics and Automation*, pp. 1779-1781, Leuven, Belgium, 1998.

[7] I.R. Nourbakhsh, C. Kunz and T. Willeke, "The mobot museum robot installations: a five year experiment," *Proceedings of the 2003 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS 2003)*, vol. 3, pp. 3636 - 3641, 2003.

[8] M.C. Silverman, D. Nies, B. Jung, and G.S. Sukhatme, "Staying alive longer: autonomous robot recharging put to the test," *Center for Robotics and Embedded Systems, CRES-03-015 Technical Report*, University of Southern California, 2003.

[9] D. Lowe, "Local feature view clustering for 3D object recognition," *Proceedings of the 2001 IEEE/RSJ, International Conference on Computer Vision and Pattern Recognition*, Kauai, Hawaii, U.S.A., pp. 682-688, 2001.

[10] D. Lowe, "Object recognition from local scale-invariant features," *Proceedings of the International Conference on Computer Vision*, pp. 1150-1157, 1999.

[11] D. Lowe, "Distinctive image features from scale-invariant keypoints," *International Journal of Computer Vision*, vol. 2, no. 60, pp. 91-110, 2004.

[12] M. E. Munich, P. Pirjanian, E. Di Bernardo, L.Goncalves, N. Karlsson, and D. Lowe, "Break-through visual pattern recognition for robotics and automation," *Proceedings of the 2005 IEEE International Conference on Robotics and Automation*, Barcelona, Spain, 2005.

[13] K. Mikolajczyk and C. Schmid, "A performance evaluation of local descriptors," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 27, no. 10, pp. 1615-1630, October, 2005.

[14] K. Mikolajczyk and C. Schmid, "Scale and affine invariant interest point detectors," *International Journal of Computer Vision*, vol. 1, no. 60, pp. 63.86, 2004.

Bag Classification Using Support Vector Machines

Uri Kartoun, Helman Stern, Yael Edan

{kartoun|helman|yael}@bgu.ac.il
Department of Industrial Engineering and Management, Ben-Gurion University of the Negev, Be'er-Sheva, Israel

Abstract:

This paper describes the design of multi-category support vector machines (SVMs) for classification of bags. To train and test the SVMs a collection of 120 images of different types of bags were used (backpacks, small shoulder bags, plastic flexible bags, and small briefcases). Tests were conducted to establish the best polynomial and Gaussian RBF (radial basis function) kernels. As it is well known that SVMs are sensitive to the number of features in pattern classification applications, the performance of the SVMs as a function of the number and type of features was also studied. Our goal here, in feature selection is to obtain a smaller set of features that accurately represent the original set. A K-fold cross validation procedure with three subsets was applied to assure reliability. In a kernel optimization experiment using nine popular shape features (area, bounding box ratio, major axis length, minor axis length, eccentricity, equivalent diameter, extent, roundness and convex perimeter), a classification rate of 95% was achieved using a polynomial kernel with degree six, and a classification rate of 90% was achieved using a RBF kernel with 27 sigma. To improve these results a feature selection procedure was performed. Using the optimal feature set, comprised of bounding box ratio, major axis length, extent and roundness, resulted in a classification rate of 96.25% using a polynomial kernel with degree of nine. The collinearity between the features was confirmed using principle component analysis, where a reduction to four components accounted for 99.3% of the variation for each of the bag types.

Keywords: Support vector machines, multi-category classification, feature selection

1 Introduction

SVMs have emerged as very successful pattern recognition methods in recent years [16]. SVMs have yielded superior performance in various applications such as; text categorization [15], and face detection [12], content-based image retrieval [6], and learning image similarity [4].

The motivation here is detection of suspicious bags in a security situation. The usual method for bomb personnel is to blow up a suspicious bag, and any explosives contained therein. However, if the bag contains chemical, biological or radiological canisters, this may lead to disastrous results. Furthermore, the "blow-up" method also destroys important clues such as fingerprints, type of explosive, detonators and other signatures of importance for forensic analysis. Extraction of the bag contents using telerobotics avoids these problems [8]. In a telerobotic system, it is advantageous to automate bag classification which is coupled to robotic tactics such as shaking out of the bags contents. For cooperative human-robot interaction, in situations when autonomous capabilities fail, a human may be called in to distinguish the bag type. Here we focus on the autonomous operation of such a telerobotic system. In the autonomous mode, bags are classified by type using SVMs for the purpose of identifying initial manipulator grasp points. One side of the gripper must slide under the bag, and because of slippage, the grasp may fail. Thus, it is of importance to realize the type of bag, and the location of the opening before the grasp point assessment is made. Moreover, because we are dealing with a soft object here, with an unknown skin texture and unknown objects inside of the bag grasp may fail. Also, for each bag type a different rule set is evoked to determine the robot arm shake trajectories to discharge the contents of the bag for subsequent inspection.

In this paper we report on multi-category support vector classification of bags. The classification approach is presented in Section 2. Image processing operations are described in Section 3. Kernel optimization and optimal feature selection experimental results are described in Section 4. In section 5 an analysis of the results is given. The paper ends with conclusions and some directions for future work.

2 Classification Approach

SVMs belong to the class of maximum margin classifiers [5]. The goal of maximum margin classification in binary SVM classification [16] is to separate the two classes by a hyperplane such that the distance to the support vectors is maximized. SVMs perform pattern recognition between two classes by finding a decision surface that has maximum distance to the closest points in the training set which are termed support vectors. The procedure starts with a training set of points $x_i \in \Re n$, $i = 1, 2, \ldots, N$ where each point x_i belongs to one of two classes identified by the label $y_i \in \{-1,1\}$. This hyperplane is called the optimal separating hyperplane (OSH). The OSH has the form:

$$f(x) = \sum_{i=1}^{N} \alpha_i y_i x_i \cdot x + b. \tag{2.1}$$

The coefficients α_i and b in (2.1) are the solutions of a quadratic programming problem [16]. A data point x is classified by the sign of the right side of (2.1). For multi-category classification, (2.2) is used.

$$d(x) = \frac{\sum_{i=1}^{N} \alpha_i y_i x_i \cdot x + b}{\| \sum_{i=1}^{N} \alpha_i y_i x_i \|}. \tag{2.2}$$

The sign of d is the classification result for x, and $| d |$ is the distance from x to the hyperplane. Intuitively, the farther away a point is from the decision surface, i.e., the larger $| d |$, the more reliable the classification result. The entire construction can be extended to the case of nonlinear separating surfaces. Each point x in the input space is mapped to a point $z = \Phi(x)$ of a higher dimensional space, called the feature space, where the data are separated by a hyperplane. The key property in this construction is that the mapping $\Phi(\cdot)$ is subject to the condition that the dot product of two points in the feature space $\Phi(x) \cdot \Phi(y)$ can be rewritten as a kernel function $K(x, y)$. The decision surface has the equation:

$$f(x) = \sum_{i=1}^{N} \alpha_i y_i K(x, x_i) + b, \tag{2.3}$$

where, the coefficients α_i and b are solutions of a quadratic programming problem. Note, that $f(x)$ does not depend on the dimensionality of the feature space. Kernel functions commonly used for pattern recognition problems are polynomial of degree d (2.4) and gaussian radial basis (2.5).

$$K(x, y) = (1 + x \cdot y)^d, \tag{2.4}$$

$$K(x, y) = e^{-\frac{\|x-y\|^2}{2\sigma^2}}. \tag{2.5}$$

The dual category SVM classification was extended to multi-category classification by [1]. There are two basic strategies for solving q-class problems with SVMs. In the one-vs-all approach, q SVMs are trained. Each of the SVMs separates a single class from all remaining classes [14, 2]. In the pairwise approach (used in this work), $q(q\text{-}1)/2$ machines are trained. Each SVM separates a pair of classes. The pairwise classifiers are arranged in trees, where each tree node represents a SVM. Regarding the training effort, the one-vs-all approach is preferable since only q SVMs have to be trained compared to $q(q\text{-}1)/2$ SVMs in the pairwise approach. The run-time complexity of the two strategies is similar: the one-vs-all and the pairwise approaches require the evaluation of q and q-1 SVMs,

respectively. Results on person recognition indicate similar classification performance for the two strategies [11]. The input to the SVMs is a set of features obtained from the bag image. The image processing feature extraction methods are described in the next section.

3 Image Processing

Image processing starts with a 24-bit color image of the robotic scene that contains a bag located on a platform. Four different bag types are considered: backpacks, small shoulder bags, plastic flexible bags, and small briefcases (Fig. 3.1). Image processing operations used are [7]: conversion to gray-scale, thresholding using the Otsu method [13], removal of noise from the image due to optical lens distortion, adaptation to ambient and external lighting conditions, and segmentation of the bag from the background. Using the MATLAB's Image Processing Toolbox [10], nine popular shape features were extracted from each segmented bag image: Area, Bounding Box Ratio, Major Axis Length, Minor Axis Length, Eccentricity, Equivalent Diameter, Extent, Roundness, Convex Perimeter.

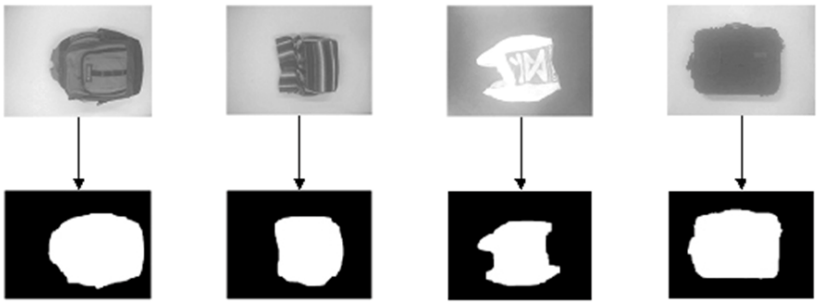


Fig. 3.1. Image processing operations

4 Experimental Results

4.1 Methodology

The heart of the support vector machine design is the kernel selection. Kernels project the data into a high dimensional feature space and thereby increase the

computational power of the linear learning machines [16, 3]. The kernels chosen for the bag classification problem are the most common ones used in support sector machines, i.e., the polynomial and the Gaussian RBF kernels [3]. The kernels were tested using the K-fold cross validation procedure with three subsets on a training set that contained 80 bag images (20 of each of the four classes). For testing, 40 bag images (10 of each class) were used. The SVMs procedure was implemented in the "OSU SVM MATLAB Toolbox" [9].

Two experiments were performed. In the first one, described in Section 4.2, a kernel optimization procedure was conducted, using the nine bag features described in Section 3, for finding the optimal polynomial degrees and RBF sigmas. In the second experiment, described in Section 4.3, an optimal feature selection was conducted for finding the number and combination of features that results in the highest classification rate using kernel optimization.

4.2 Kernel Optimization Experiment for Bag Classification

Classification process was performed for the range of 1-100 degrees / sigmas applying both kernels for finding the optimal polynomial degrees and RBF sigmas using all the nine features mentioned in Section 3.

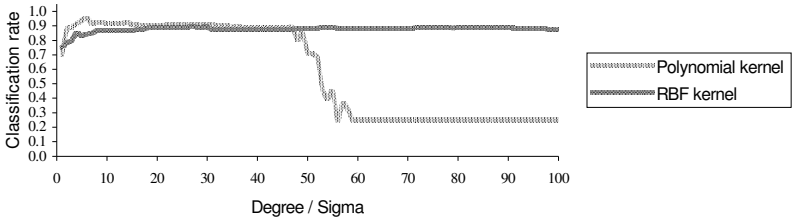


Fig. 4.2.1. Classification rate vs. degree and sigma for four bags classification using nine image features

As can be seen in Fig. 4.2.1 for the nine bag features case, the highest average classification rate between the three subsets achieved was 95% using a polynomial kernel with six degree and 90% using a RBF kernel with 27 sigma. The confusion matrices that summarize the results are shown in Table 4.2.1, where the upper and lower diagonal values in each cell correspond to percent of correct classifications for the polynomial and RBF kernels, respectively.

Table 4.2.1. Confusion matrices for polynomial/RBF kernels for a four bag classification using nine image features

predicted class	true class classification rates [%]			
	backpack	small shoulder bag	plastic flexible bag	small briefcase
backpack	96.7 / 93.3	0 / 0	0 / 0	3.3 / 6.7
small shoulder bag	3.3 / 0	93.4/ 93.3	0 / 0	3.3 / 6.7
plastic flexible bag	0 / 3.3	0 / 3.3	96.7 / 93.4	3.3 / 0
small briefcase	3.3 / 20	3.3 / 0	0 / 0	93.4 / 80

4.3 Optimal Feature Selection

A full enumeration feature selection procedure was performed to choose a set of optimal features to improve the results achieved in Section 4.2. Since there are up to nine features to be used in the classification procedure, there are $2^9 - 1 = 511$ combinations for selection. The classification process was performed for the range of 1-100 degrees / sigmas applying both kernels to find the optimal set of bag image features corresponding to the optimal polynomial degrees and RBF sigmas giving the highest classification results. As can be seen from Fig 4.3.1, the largest average classification rate can be obtained by using only four features for both the polynomial and RBF kernels.

Details of the feature selections and the average classification rates appear in Table 4.3.1. The polynomial kernel's average classification rate of 96.25% was superior to that of the RBF (91.6%). It is noted that for the polynomial kernel, the minimum set of features (among the ties) consists of four features: bounding box ratio, major axis length, extent and roundness. Also, from Table 4.3.1 these features correspond to the optimal polynomial kernel of degree nine.

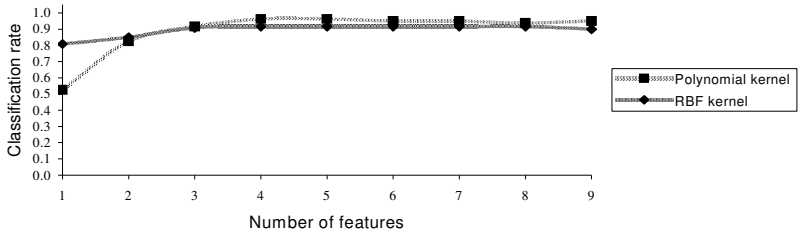


Fig. 4.3.1. Classification rate as a function of number of features using the optimal degrees / sigmas

Table 4.3.1. Feature selection results for the highest classification rates achieved

kernel	polynomial			RBF		
number of features	feature(s)	degree	classification rate [%]	feature(s)	sigma	classification rate [%]
1	G	6	52.5	F	53	80.8
2	G, I	6	82.5	A, G	9	85
3	D, F, G	8	91.6	D, G, H	33	90.8
4	B, C, G, H	9	96.25	B, D, E, G	48	91.6
5	B, E, F, G, H	6	96.25	B, D, E, G, H	39	91.6
6	B, C, D, E, G, H	6	95	B, C, D, E, G, H	60	91.6
7	B, C, D, E, G, H, I	6	95	B, C, D, E, G, H, I	56	91.6
8	A, B, C, D, E, F, H, I	6	93.75	A, C, D, E, F, G, H, I	63	91.6
9	A, B, C, D, E, F, G, H, I	6	95	A, B, C, D, E, F, G, H, I	27	90

A Area, B Bounding Box Ratio, C Major Axis Length, D Minor Axis Length, E Eccentricity, F Equivalent Diameter, G Extent, H Roundness, I Convex Perimeter.

Another view of the results are depicted in Fig. 4.3.2, using the optimal set of features (bounding box ratio, major axis length, extent and roundness).The curve shows that the average classification rate peaks for an optimal polynomial kernel degree nine.

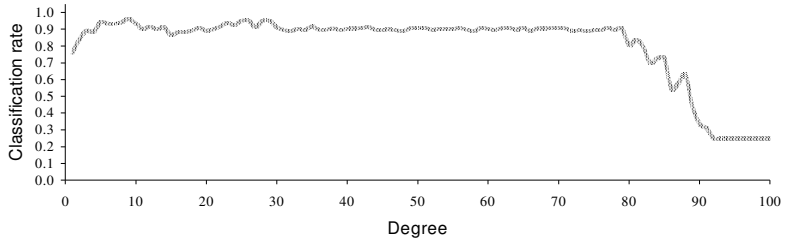


Fig. 4.3.2. Classification rate vs. degree for four-bag classification using optimal features (bounding box ratio, major axis length, extent and roundness)

5 Analysis of the Results

The polynomial kernel's classification rate of 96.25% was superior to that of the RBF (91.6%) using only four out of the nine original features. The small number of features deemed to be optimal was hypothesized to be due to correlation between the features. This is verified by examining the correlation matrix (Table 5.1), which exhibited correlation coefficients as high as 0.99.

Table 5.1. Correlation matrix

features	A	B	C	D	E	F	G	H	I
A	1	0.08	0.96	0.92	0.28	0.99	-0.27	-0.26	0.85
B	0.08	1	0.28	-0.18	0.69	0.1	0.01	-0.74	0.48
C	0.96	0.28	1	0.81	0.49	0.96	-0.3	-0.49	0.96
D	0.92	-0.18	0.81	1	-0.08	0.93	-0.31	0.11	0.61
E	0.28	0.69	0.49	-0.08	1	0.26	0	-0.99	0.69
F	0.99	0.1	0.96	0.93	0.26	1	-0.24	-0.25	0.86
G	-0.27	0.01	-0.3	-0.31	0	-0.24	1	0.02	-0.2
H	-0.26	-0.74	-0.49	0.11	-0.99	-0.25	0.02	1	-0.7
I	0.85	0.48	0.96	0.61	0.69	0.86	-0.2	-0.7	1

A Area, B Bounding Box Ratio, C Major Axis Length, D Minor Axis Length, E Eccentricity, F Equivalent Diameter, G Extent, H Roundness, I Convex Perimeter.

To further confirm the collinearity between the features we performed a principle component analysis (PCA), where a reduction to four components accounted for 99.3% of the variation for each of the bag types. Eigenvalues of the covariance matrix and the cumulative percentage of the total variance in the observations explained by the eigenvectors are shown in Table 5.2.

Table 5.2. Eigenvalues and variance explained matrix

variance explained [%]	62.65	93.46	96.9	99.31	99.79	99.93	99.97	100	100
eigen-value	0.0948	0.056	0.009	0.003	0.0007	0.0003	0.0001	0	0

It is noted, that the reduction from nine to four features was done by a complete enumeration feature selection method. It is a coincidence that a PCA allowed a reduction to four "components", but the PCA was not used to represent the feature reduction through a linear combination of the original features, as in the usual case.

6 Conclusions

Multi-category bag classification for four-bag classes was performed using SVMs. The SVMs procedure was tested using polynomial and RBF kernels. A K-fold cross validation procedure with three subsets was used. In a kernel optimization experiment using nine popular shape features, classification rates of 95% and 90% were achieved using a polynomial kernel of degree six and a RBF kernel with 27 sigma, respectively. The confusion matrices indicate that decreased classification rates may be due to the inability to discriminate between the backpacks and the small briefcases bag images. To improve these results, a full enumeration feature selection procedure for choosing a set of optimal features for describing a bag image was performed. The set of optimal features found was: bounding box ratio, major axis length, extent and roundness. Using these features a classification rate of 96.25% was obtained for a polynomial kernel of degree nine. It was also found that using more than four features resulted in no improvement. The resulting optimal reduction of features from nine to four was hypothesized to be due to correlation between the features. The collinearity between the features was confirmed using principle component analysis, where a reduction to four components accounted for 99.3% of the variation for each of the bag types. Future work includes a comparison of the SVM classifier with that of using a PCA classifier.

Acknowledgments

This project was partially supported by the Ministry of Defense MAFAT Grant No. 1102, and the Paul Ivanier Center for Robotics Research and Production Management, Ben-Gurion University of the Negev. The assistance of Prof. Ehud Menipaz is gratefully acknowledged.

References

[1] Bennett, K.P. and Bredensteiner, E.J. (1999), "Multicategory classification by support vector machines", *Computational Optimizations and Applications*, vol. 12, pp. 53-79.

[2] Cortes, C. and Vapnik, V. (1995), "Support vector networks", *Machine Learning*, vol. 20, pp. 1-25.

[3] Cristianini, N. and Shawe-Taylor, J. (2003), *Support Vector Machines and other Kernel-based Learning Methods*, Cambridge University Press, Cambridge, U.K.

[4] Guo, G.D., Li, S.Z., and Chan, K.L. (2000), "Learning similarity for texture image retrieval", *Proceedings of the European Conference on Computer Vision*, pp. 178-190.

[5] Heisele, B., Ho, P., Wu, J., and Poggio, T. (2003), "Face recognition: component-based versus global approaches", *Elsevier Science Inc.*, New York, U.S.A., vol. 91, pp. 6-21.

[6] Hong, P., Tian, Q., and Huang, T.S. (2000), "Incorporate support vector machines to content-based image retrieval with relevance feedback", *Proceedings of the International Conference on Image Processing*, vol. 3, pp. 750-753.

[7] Kartoun, U. (2003), "A human-robot collaborative learning system using a virtual reality telerobotic interface", *Ph.D. Thesis Proposal, Department of Industrial Engineering and Management at the Ben-Gurion University of the Negev, Israel*.

[8] Kartoun, U., Stern, H., and Edan, Y. (2004), "Virtual reality telerobotic system", *e-ENGDET 2004 4th International Conference on e-Engineering and Digital Enterprise Technology*, Leeds Metropolitan University Yorkshire, U.K.

[9] Ma, J. and Ahalt, S. (2003), OSU SVM Classifier Matlab Toolbox (ver. 3.00).

[10] The MathWorks Inc., (1998), Matlab Software, Image Processing User's Guide.

[11] Nakajima, C., Pontil, M., Heisele, B., and Poggio, T. (2000), "Person recognition in image sequences: the MIT espresso machine system", *IEEE Transactions on Neural Networks*.

[12] Osuna, E., Freund, R., and Girosi, F. (1997), "Training support vector machines: an application to face detection", *Proceedings of Computer Vision and Pattern Recognition*, pp.130-136.

[13] Otsu, N. (1979), "A threshold selection method from gray level histograms", *IEEE Transactions on Systems, Man, and Cybernetics*, vol. 9, pp. 62-66.

[14] Schölkopf, B., Burges, C., and Vapnik, V. (1995), "Extracting support data for a given task", In U. Fayyad and R. Uthurusamy, editors. *Proceedings of the First International Conference on Knowledge Discovery and Data Mining*, AAAI Press, pp. 252-257.

[15] Shanahan, J.G. and Roma, N. (2003), "Boosting support vector machines for text classification through parameter-free threshold relaxation", *Proceedings of the Twelfth International Conference on Information and Knowledge Management*, pp. 247-254.

[16] Vapnik, V. (1998), *Statistical Learning Theory*, JohnWiley and Sons, New York.

An Expert System for Surveillance Picture Understanding

Helman Stern, Uri Kartoun, Armin Shmilovici
Faculty of Engineering, Ben-Gurion University, P.O.Box 653, Be'er-Sheeva 84105, ISRAEL, Fax: +972-8-6472958; Tel: +972-8-6477550, E-mail:(helman, kartoun, armin)@bgumail.bgu.ac.il

Abstract: The last stage of any type of automatic surveillance system is the interpretation of the acquired information from the sensors. This work focuses on the interpretation of motion pictures taken from a surveillance camera, i.e.; image understanding. An expert system is presented which can describe in a natural language like, simple human activity in the field of view of a surveillance camera. The system has three different components: a pre-processing module for image segmentation and feature extraction, an object identification expert system (static model), and an action identification expert system (dynamic temporal model). The system was tested on a video segment of a pedestrian passageway taken by a surveillance camera.

Keywords: image understanding, picture segmentation, fuzzy expert systems, surveillance video

1. INTRODUCTION

With the continuous decline in the price of imaging technology, there is a surge in the use of automatic surveillance systems and closed circuit TV (CCTV). Banks, ATM machines, schools, hospitals, transport walkways employ automatic video recording of their surrounding environments. There appears to be little human inspection (in real-time or otherwise) of these surveillance videos, and thus the system is relegated to a simple deterrence function (mainly for deterrence of possible felonies). However, in many environments it is necessary to understand the contents of the video for subsequent event detection, storage and retrieval. Extraction of the desired events requires a high semantic level of human understanding and requires a prohibitive amount of human processing.

Automatic processing of surveillance videos introduces several practical problems. Recording, storing and managing of large volumes of data is expensive, and cost effective solutions are not yet available for cases where most of the data is useless. Also, in the case that someone will want to

inspect the contents of the video, there would be a great deal of work involved in watching all the recorded segments. There is an imperative need for efficient automated retrieval of desirable information.

Those problems did not escape the surveillance industry: saving of storage space is introduced by time-laps recording (*e.g.*, a frame a second), and by motion-activated recording. While both of those methods are easy to implement, they do not solve the problem of storing the images by a meaningful key that will later facilitate human retrieval of important information (*e.g.*, the face of a felon). Also, the activation sensor might be sensitive to spurious events (*e.g.*, the passage of a pet) which are considered unimportant. The real need is for an effective means by which the content of the data can be automatically characterised, organised, indexed, and retrieved, doing away with the slow, labour-intensive manual search task. Understanding the contents of an image, in the context of its importance to the operator of the surveillance system, is the key to efficient storage and retrieval of video segments.

The problem of automatic image understanding is a difficult one. The problems of modelling and understanding visual behaviours and their semantics are often regarded as computationally ill-defined [14]. In simple terms, the meaning of a behaviour (such as hand-waving) is context dependent. Figuring out the context from the image could be more difficult than identifying the behaviour or the context may not depend on visual information at all.

There are two possible paradigms for that problem [1,2]: the computational feature based semantic analysis – the detection of features based on elaborate computational models, and the human cognitive perception of high level semantics [18], i.e., the subjective interpretation of the user based on some features in the image. Both approaches can be integrated: low level feature detection integrated with interpretation semantics [15,16,17].

With the computational paradigm, it is technically difficult to identify correctly and in a reasonable amount of time, the contents of an image in all possible circumstances (*e.g.*, identify an object from all possible angles). There is a need to develop a model for the features of an image in all possible circumstances.

Human cognitive perception, on the other hand, starts with simple object segmentation, that is to segment projected 2D-plane images (generated from a video sequence) of an arbitrary 3D scene into physically meaningful objects. Image segmentation is potentially simpler than feature identification. Image understanding is related to the relations between the objects in the picture, and the context (*e.g.*, when an object is of importance),

and that in general is very difficult to formulate. Yet, people, even children, can learn to do it with ease.

The problem of image understanding can be facilitated [3,4] if we can restrict the type of objects to be identified, (*e.g.*, humans), the quality of the identification (*e.g.*, contours only), the possible relations between objects (*e.g.*, approaching each other), and the context in which they operate (*e.g.*, a closed passageway).

In this work a prototype of a fuzzy expert system is presented which can describe, in natural language like, simple human activity in the field of view of a surveillance camera. The system has three different components: a pre-processing module for image segmentation and feature extraction, an object identification fuzzy expert system (static model), and an action identification fuzzy expert system (dynamic temporal model). The system was tested on a video segment of a pedestrian passageway taken by a surveillance camera

The rest of this paper is organised as follows: Section 2 describes the surveillance problem and its simplification. Section 3 describes the construction of the fuzzy expert system. Section 4 explains the static and dynamic fuzzy expert systems for object identification and object behaviour, respectively. Section 5 provides the results of applying the system to a pedestrian passageway. Section 6 concludes the paper with some discussion.

2. PROBLEM DEFINITION

Figure 1 presents a typical scene that might be observed under a surveillance system. This scene is a semi-covered passageway between two buildings at Ben-Gurion University. The open door on the lower left leads to a cafeteria, while there is a lecture hall on the right side with large windows. The sunlight enters through the right side of the scene during daytime, so the lighting conditions can change drastically during the day causing variable length shadows to be present in this scene. It was decided to use a single camera whose visual axis points down into the scene. This affords the use of prior knowledge regarding a minimalist modelling, for future real-time processing, and to exploits complimentary qualities of different visual clues, such as relative sizes and motion of objects.

In the scene below, people are the objects of interest, though occasionally other objects appear such as, an electric delivery cart to the cafeteria and birds. It is desired to describe the activities in this scene in terms of the activities of the people.

Figure 1. A typical scene taken from a surveillance camera

The full range of possible human activities that can be described by a natural language is very large indeed. Fortunately, in the context of surveillance, there is interest only in a small subset of these human activities. In this project, *abnormal* incidents were defined as {parcel-near-cafeteria, a running person, a person outside working hours}. Those are indications for abnormal behaviour such as deployment of a parcel bomb, trespassing, violence and theft. The examples outline the identification of normal gross human behaviour. In the context of the scene above, people can be either standing or walking; they could be alone, or in groups of two or more. In either case, the position of the person or group is described relative to the scene.

As it turns out, it is still technically difficult to identify the concept of a "person" in a noisy environment such as that described above, especially with artefacts due to changing lighting conditions, shadows, optical distortions, reflections on large glass windows, and the variability of people motions. Also, there is a limit on the reasonable computational time needed to generate a description. Thus, further simplifications to the problem were made [5,6]:

i) A primitive notion of a "blob" is defined as a set of clustered pixels that have moved between two given images. The blobs in each image were segmented from the background of the image. A blob may or may not be a real object in the scene such as a person. It may instead be background noise that was not removed during the image segmentation operations. This simplification, however, facilitated the necessary image pre-processing operations.

ii) A fuzzy inferencing mechanism is used for perceptual integration of simple visual clues. The theory of fuzzy sets is employed which can use a linguistic variable such as distance, for example. This linguistic variable can take on various terms such as; very close, close, far, etc. This replaces the "crisp" mathematical description of distance such as; 4.15 meters. This facilitated the mathematical models that capture and describe the activities in the scene into a natural like language.

The goal of this project was defined as follows: develop a prototype of a fuzzy expert system that can understand and describe a scene in a natural like language. Fuzzy rules are based on domain expert knowledge to describe a scene, locations of objects of interests, object descriptions and object behaviours.

Given a set of scenes (images) from a video clip {i.e. a set of images I(t), I(t-1), I(t-2), etc. taken at consecutive times}, describe the scene in terms of number of people, and people groups in the scene and actions such as walking toward or away from the camera, standing still, departing from another person, walking with another person, joining a group, etc.

3. FUZZY EXPERT SYSTEMS

Fuzzy set theory [7] and fuzzy expert systems [8] are used to capture expert knowledge that cannot be easily formulated mathematically, or when the mathematics is too complex to solve. Building an expert system starts with interrogating the domain experts (in this case, image processing experts and surveillance staff) and formulating their knowledge in the form of linguistic variables and fuzzy rules [7]. Additional domain knowledge can be included. Also, some knowledge can be derived from statistical analysis of historical information. References [9-11] present applications of fuzzy logic to image processing.

The fuzzy system considered in this paper is comprised of four basic elements [7]: a fuzzifier, a fuzzy rule base, a fuzzy inference engine, and a defuzzifier. We consider multi-input single-output fuzzy systems as elaborate mapping functions: f: $U \subset R_n \rightarrow V \subset R$, where $U = U_1 \times U_2 \times \ldots \times U_n \subset R^n$ is the input space and $V \subset R$ is the output space. A multi-output system can be represented as a group of single-output systems.

A rule is a proposition that implies another proposition. In this paper, the fuzzy rule base consists of a set of linguistic rules in the form of "IF a set of conditions is satisfied THEN a set of consequences is inferred". Assume that there are N rules of the following form:

R_i: IF x_1 is A_{i1} and x_2 is A_{i2} and...and x_n is A_{in} THEN y is C_i, i=1,2,...,N

where x_j ($j=1,2,...,n$) are the input variables to the fuzzy system, y is the output variable of the fuzzy system, and the fuzzy sets A_{ij} in U_j and C_j are linguistic terms characterized by fuzzy membership functions Aij(xj) and Ci(y), respectively. Each rule Ri can be viewed as a fuzzy implication (relation) $A_i = A_{i1} x A_{i2} x ... x A_{in} \rightarrow C_i$, which is a fuzzy set in $U \times V = U_1 \times U_2 \times \ldots \times U_n \times V$ with membership function $R_i(\bar{x}, y) = A_{i1}(x_1) * A_{i2}(x_2) * ... * A_{in}(x_n) * C_i(y)$, and* is the T norm [7], $\bar{x} = (x_1, x_2, ..., x_n) \in U$ and $y \in V$.

4. THE STATIC AND DYNAMIC FUZZY EXPERT SYSTEMS MODELS

In this section the static and dynamic (temporal) expert systems are described. Initially, however, a brief discussion of the pre-processing stage is given although this is not the main focus of the paper. In the Pre-processing stage, a raw image is pre-processed for the identification of blobs. The main function of the static expert system is for object identification. It uses the geometrical attributes of the blobs to make inferences about the objects in the picture. The main function of the dynamic expert system is for action identification. It uses temporal movement attributes of objects to make inferences about the behaviours of the objects in the picture.

4.1 Image Pre-processing

The pre-processing stage starts with a grabbed gray-scale image of a scene. Various image processing operations are used to remove noise from the image due to optical distortions of the lens, adapt to ambient and external lighting conditions, and shareholding to segment out the blobs from the background. The end result is an image of segmented blobs from which features are extracted. Using the Image Processing Toolbox of MATLAB, twelve different geometrical attributes were defined for each blob in the image:

- *Area* - the actual number of pixels in a blob.
- *Convex area* - the number of pixels in the convex area of a blob.
- *Solidity* - the ratio between the above two area measures.
- The *Equivalent Diameter* of a circle with same area.
- *Centroid* - the coordinates of the center of gravity of the blob.
- The coordinates of the *Bounding Box*.
- The *Minor Axis Length*, *Major Axis Length*, and *Eccentricity* for a bounding ellipsoid.

– The *Orientation* (in degrees) to the horizon.
– *Extent* - the proportion of the pixels in the bounding box that are also in the region. Computed as the Area divided by area of the bounding box.

Figure 2 presents a segmentation of the scene shown in figure 1. Table 3 presents further details of the segmentation.

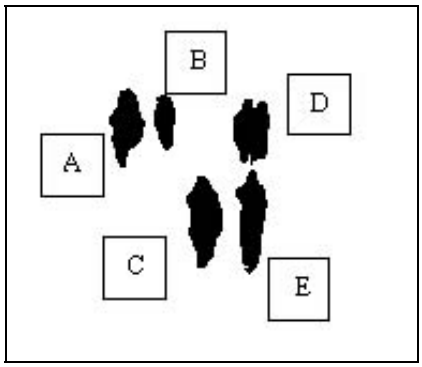


Figure 2. Segmentation of Figure 1 into blobs

Different features are associated with each blob. Static features, such as blob centroid and size, will be used to classify each blob in the scene into separate categories such as: one person; two people; more than two people or a noise blob. These are discussed further in the section on the static model. Dynamic features, such as direction of movement relative to the camera, will be used to classify the activities of each blob in the scene into categories such as: blob moving toward camera, blob moving away from camera, blob is stationary. Other possible descriptions, not investigated here, result from the relational positions between different blobs, such as: blob has merged with another blob; a blob has split into two blobs; a close pair of blobs walk together; two blobs meet and stop; two blobs depart from each other and move in different directions, etc. These are discussed further in the section on the dynamic model.

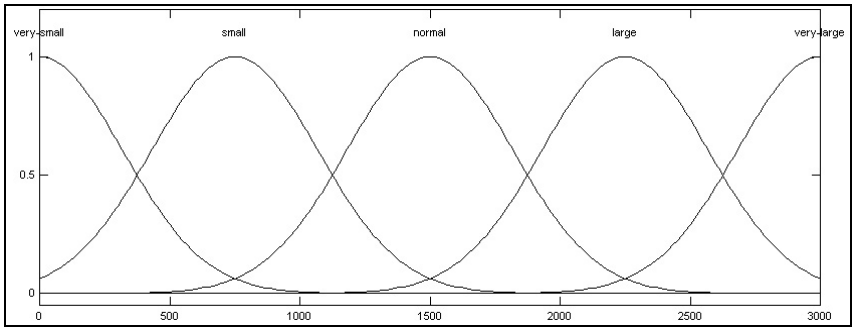
4.2 The Static Expert System Model

For the object identification stage, a fuzzy expert system was built, with three input variables and one output (conclusion) variable. Linguistic variables were defined for each input.

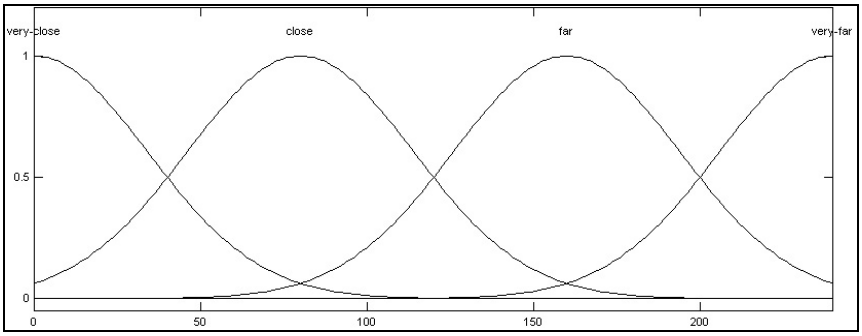
– The area (in pixels) of the blob, defined on the range [0, 3000] pixels, can take on five terms: Area = {*very-small, small, normal, large, very-large*}

– The aspect ratio of the bounding box (the height/width ratio of the smallest rectangle that can contain a blob), defined on the range [0, 3.5], can take on five terms: Ratio = {*very-small, small, normal, large, very-large*}
– The y-coordinates of the center of mass of a blob (a simple estimator for the distance from the camera), defined on the range [0, 250] pixels, can take on four terms: Distance = {*very-far, far, close, very-close*}
– The conclusion about the identity of the blob, defined on the range [0, 30], can take on five terms: Conclusion = {*not-a-person, single, couple, three, many*}

Figures 3(a), 3(b) present the fuzzy sets defined for two of the linguistic variables. Those were selected to be Gaussian membership function from Matlab's fuzzy logic toolbox [13] after some experimentation with the possible ranges. Table 1 presents the rule-base used for blob identification. The seemingly simple rules are logically derived from prior knowledge of the relative position of the camera position and the scene. Also, the rules utilise implicit relational knowledge about the image, such as: "a far object has small area", or "groups have larger area than single person". While it is possible to formulate mathematical functions for these notions, they most likely will be complex because of the stochastic nature of the relationships.



3(a) Membership input function for size of a blob



3(b) Membership input function for distance from camera of a blob

Figure 3. Membership functions for inputs, and for the output conclusion about blob

*Table 1. F*uzzy rules for blob identification

Premise part	Conclusion part
If Area = *very-small*	Then *not-a-person*
If Distance = *close* and Area = *small* and Ratio = *normal*	Then *single-person*
If Distance = *close* and Area = *small* and Ratio = *very-large*	Then *single-person*
If Distance = *close* and Area = *normal* and Ratio = *large*	Then *single-person*
If Distance = *close* and Area = *normal* and Ratio = *very-large*	Then *single-person*
If Distance = *close* and Area = *large* and Ratio = *normal*	Then *single-person*
If Distance = *far* and Area =*very-small* and Ratio = *very-large*	Then *single-person*
If Distance = *far* and Area = *small* and Ratio = *very-large*	Then *a-couple*
If Distance = *far* and Area = *normal* and Ratio = *large*	Then *three-people*

4.3 The Dynamic Expert System Model

For the action identification stage, a second fuzzy expert system is defined with two input variables and two output (conclusion) variables. Linguistic variables are defined to represent the temporal aspects of the blobs.

– The X-movement change - the change of the centroid of a blob in the x-axis. Defined on the range [-5, +5] pixels, can take on five terms: X-movement = {*dramatically-left, slightly-left, almost-no-change; slightly-right; dramatically-right*}.
– The Y-movement change - the change of the centroid of a blob in the y-axis (in relation to the camera), defined on the range [-5, +5] pixels, can take on four terms: Y-movement = {*dramatically-away, slightly-away, almost-no-change; slightly-forward, dramatically-forward*}.

– The conclusion about the object's velocity, defined on the range [-2, +2] units can take on four terms: Velocity = {*standing, slow, fast, running*}.
– The conclusion about the object's direction, defined on the range [0,1] can take on eight terms: Direction = {*right, forward, away-left, away-right, forward-right, forward-left, away, left*}.

Figure 4 presents an example of the membership functions defined for one of the linguistic variables called x-axis change. Table 2 presents part of the rule base used for blob action identification.

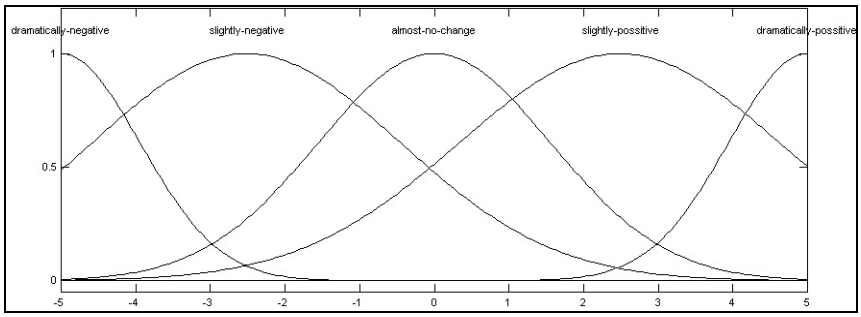


Figure 4. Membership functions for the x-axis change conclusion

*Table 2. F*uzzy rules for blob action identification

Premise part	Conclusion part
If X-movement = *slightly-right* and Y-movement = *almost-no-change*	Then Velocity = *standing* and Direction = *none*
If X-movement = *slightly-right* and Y-movement = *slightly-forward*	Then Velocity = *slow* and Direction = *forward-right*
If X-movement = *slightly-right* and Y-movement = *dramatically-forward*	Then Velocity = *fast* and Direction = *forward-right*
If X-movement = *almost-no-change* and Y-movement = *almost-no-change*	Then Velocity = *standing* and Direction = *none*
If X-movement = *dramatically-left* and Y-movement = *almost-no-change*	Then Velocity = *fast* and Direction = *right*

The fuzzy system was implemented with MATLAB's "fuzzy logic tool box". The operation of the fuzzy system used in this paper was restricted to "Centroid defuzzifier" - that is the center of gravity operator was used to combine the different values resulting from the activation of each fuzzy rule into one crisp result.

Note that this is a prototype system only, so some choices (*e.g.*, the shape of the membership functions) are rather arbitrary. Though further optimisation is possible, one of the typical characteristics of fuzzy expert system is that their predictions are reasonable even with a minimal number of assumptions, and a partial rule-base.

5. APPLICATION OF THE PROCEDURE TO THE UNDERSTANDING HUMAN ACTIVITIES IN A PEDESTRIAN PASSAGEWAY

A video clip of about 12 seconds (25 frames per second) was taken from the surveillance camera at Ben-Gurion University. The video clip was divided into 299 frames. The pre-processing stage was performed using Matlab's Image Processing Toolbox. This stage generated all the blobs and their features.

Several sets of frames were used to test the performance of the fuzzy systems for different scenarios. In this paper, Frames 170 (figure 1) and 180 (figure 5) are used for demonstrating the behaviour of the action identification dynamic fuzzy expert system described in Table 2. Figure 6 is the segmentation of frame 180 (shown in Figure 5) of the video.

Figure 5. Frame no. "180"

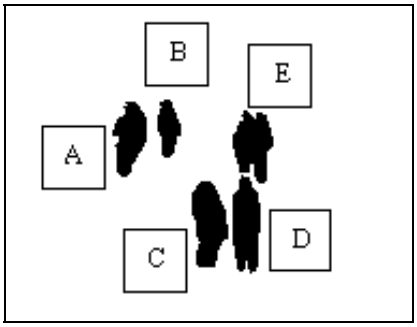


Figure 6. Segmentation of frame no. “180”

Table 3 presents the values of the geometrical attributes; Area, BoundingBox and Centroid for the blobs of frames 170, and 180. The conclusions of the static expert system (of Table 1) regarding the identity of the blobs in figure 2 are presented in bold letters. For example, blob B is correctly identified as a single person. Blob D is correctly identified as a couple.

The last three rows in Table 3 present the input and output variables from the dynamic fuzzy expert system. Based on tracking the outputs of the static expert system the attributes of X,Y movements are determined. The speed and the direction of each blob are identified and shown in the last two rows of the table. For example, blob A was identified as standing. Blob C was identified as moving slowly in the forward-right direction. Thus, the content of the scene could be summarised with five natural like language statements such as; “A single person in location (x,y) is moving slowly in the forward-right direction”. Furthermore, tracing blobs over time, can be used to identify splitting and merging events such as; “one person + two people = three people”.

Table 3. Outputs of fuzzy systems regarding blobs in figures 2, 6

Blob Identification	A	B	C	D	E
Area (170) (fig. 1)	378	195	490	401	458
BoundingBox(170) (fig. 1)	[128.5 36.5 17 37]	[149.5 39.5 10 26]	[164.5 77.5 17 43]	[186.5 41.5 17 31]	[187.5 74.5 15 49]
Centroid (170) (fig. 1)	[136.6 53.9]	[154.7 51.7]	[173.1 98.1]	[195.2 56.5]	[195.1 97.9]
Fuzzy Blob Type Conclusion	couple	Single	single	couple	single
Area (180) (fig. 6)	369	204	489	434	474
BoundingBox(180) (fig. 6)	[129.5 37.5 16 35]	[150.5 36.5 11 27]	[166.5 74.5 17 40]	[185.5 41.5 19 34]	[185.5 71.5 13 46]
Centroid (180)	[137.7 54.1]	[155.6	[174.4	[195.1	[191.9

(fig. 6)		49.7]	94.1]	57.5]	95.0]
Fuzzy Blob Type Conclusion	couple	Single	single	couple	single
X,Y-movements	1.03 0.19	0.92 -1.99	1.35 -3.99	-0.04 1.03	3.17 -2.95
Fuzzy Blob Speed Conclusion	stand	Stand	slow	stand	slow
Fuzzy Blob Direction Conclusion	None	None	forward-right	None	forward-right

6. CONCLUSIONS AND DISCUSSION

An expert system has been developed for high-level understanding of images from a surveillance video. Two fuzzy inference systems were described - static and dynamic. In the implementation of these two systems the static model is used to identify blobs as clusters of people and provide an estimate of the number of people in each blob. The dynamic model uses temporal information between frames to obtain movement information of the blobs, and with little additional effort can place existing blobs into new categories by identifying merge and split activities. Although, correct blob identifications were made for the frames examined; further testing is required. The evaluation of such extended testing requires a performance measure such as; percent of corrected blob types (for the static model), and mean square velocity error (for the dynamic model).

In general, the blob identification system was correct in most cases. Although not implemented here, it is possible to display a “degree of belief” of the expert system regarding the validity of its conclusions. The degree of belief is the value of the membership of the output variables. The larger the membership value, the higher the level of belief. Low degrees of belief will trigger the collection of additional information before presenting the conclusion. Also, further optimisation of the fuzzy system will reduce the probability of false classifications.

Content description is often used for video storage for future retrieval [19,20]. In the context of surveillance, it may be interesting to place the linguistic classification in a database. Especially, it may be interesting to record and update a list of people in the scene. This can act as a “pedestrian log”. The database can then be used to index the images for archival purposes and subsequent retrieval. For example, each blob in Figure 5 will be indexed by its identified class and its identified actions. It is possible to store only the image information in the bounding box of each blob, to obtain image compression, with minimal information loss. Other expert systems can use this image information to make inferences about other types of

activities such as violence, vandalism or theft. It may also be used to study behavioural aspects of crowds or pedestrians. For example, one can query the database for “who gives way to whom”, and for “do large blobs act as attractors for passerby’s”?

7. REFERENCES

[1] Hanjalic, A. and Li-Qun Xu, Video Analysis and Retrieval at Affective Level, submitted to Special Issue of *IEEE Transactions on Multimedia Database*, April 2001.

[2] Xu, Li-Qun, Jian Zhu and Fred Stentiford, Video Summarization and Semantics Editing Tools, in SPIE Proceedings - *Storage and Retrieval for Media Databases* 2001, San Jose, CA, January, 2001.

[3] ICONS: Incident Recognition for Surveillance and Security, DTI/EPSRC LINK Project, July 2000-July 2003, http://www.dcs.qmw.ac.uk/research/vision/projects/ICONS/

[4] VIGOUR - An Integrated Vision System for Research in Human Recognition, The ISCANIT Project, http://www.dcs.qmw.ac.uk/research/ISCANIT.

[5] Dickinson S. J., Christensen H. I., Tsotsos J. K. and Olofsson G. Active Object Recognition Integrating Attention and Viewpoint Control. *Computer Vision and Image Understanding*. Vol. 67, No. 3, September, pp. 239-260, 1997.

[6] Lengagne R., Jean-Philippe T. and Olivier M. From 2D Images to 3D Face Geometry. *Proceedings of IEEE Second International Conference on Automatic Face and Gesture Recognition*, 1996.

[7] Zimmerman H., "*Fuzzy Set Theory*", Kluwer, 1991.

[8] Cox E., "*The Fuzzy Systems Handbook*", AP professionals", Boston, 1994.

[9] Shanahan J., Thomas B., Mirmehdi M., Campbell N., Martin T. and Baldwin J. Road Recognition Using Fuzzy Classifiers. Advanced Computing Research Center University of Bristol, 2000.

[10] Balsi M. and Voci F., Fuzzy Reasoning for the Design of CNN - Based Image Processing Systems. *Proceedings of IEEE International Symposium on Circuits and Systems (ISCAS 2000)*, Geneva, Switzerland, 2000.

[11] Matsakis P., Keller J., Wendling L., Marjamaa J. and Sjahputera O. Linguistic Description of Relative Positions in Images. University of Missouri-Columbia, Columbia, USA., 2000.

[12] Matlab software, Image Processing Toolbox User's Guide. The MathWorks, Inc. www.mathworks.com, 1998.

[13] Matlab software, "fuzzy logic toolbox", The MathWorks Inc., www.mathworks.com, 2000.

[14] Gong S. and Buxton H., Editorial: Understanding visual behaviour. *Image and Vision Computing*, Vol. 20, No. 12, pp. 825-826, 2002.

[15] Xiang T. and Gong S., Discovering Bayesian causality among visual events in a complex outdoor scene. In Proc. IEEE International Conference on Advanced Video- and Signal-based Surveillance, Miami, USA, July 2003.

[16] Galata A., Learning Spatio-temporal models of object Behavior, 2003.

[17] Xiang T. and Gong S., Scene event recognition without tracking. Special issue on visual surveillance of dynamic scenes, *Acta Automatica Sinica* (Chinese Journal of Automation), Chinese Academy of Sciences, Vol. 29, No. 3, pp. 321-331, 2003.

[18] Gong S., Ng J. and Sherrah J., On the semantics of visual behaviour, structured events and trajectories of human action. *Image and Vision Computing*, Vol. 20, No. 12, pp. 873-888, 2002.

[19] Veltkamp, R. C. Burkhardt H., Kriegel H. P. (eds) State-of-the-Art in Content-Based Image and Video Retrieval, Kluwer, 2001.

[20] CogVis, cogvis.informatik.uni-hamburg.de, "Context-Based Image Retrieval: Performance evaluation and Semantic Scene Understanding, Vogel J., annotated biography

COLLABORATIVE Q(λ) REINFORCEMENT LEARNING ALGORITHM - A PROMISING ROBOT LEARNING FRAMEWORK

Uri Kartoun*, Helman Stern*, Yael Edan*, Craig Feied**, Jonathan Handler**, Mark Smith**, Michael Gillam**
*Department of Industrial Engineering and Management, Ben-Gurion University of the Negev
Be'er-Sheeva, 84105, Israel
{kartoun, helman, yael}@bgu.ac.il
**Institute for Medical Informatics, Washington Hospital Center
110 Irving St., Washington DC, NW, 20010, U.S.A.
{cfeied}@ncemi.org

ABSTRACT
This paper presents the design and implementation of a new reinforcement learning (RL) based algorithm. The proposed algorithm, $CQ(\lambda)$ (collaborative $Q(\lambda)$) allows several learning agents to acquire knowledge from each other. Acquiring knowledge learnt by an agent via collaboration with another agent enables acceleration of the entire learning system; therefore, learning can be utilized more efficiently. By developing collaborative learning algorithms, a learning task solution can be achieved significantly faster if performed by a single agent only, namely the number of learning episodes to solve a task is reduced. The proposed algorithm proved to accelerate learning in navigation robotic problem. The $CQ(\lambda)$ algorithm was applied to autonomous mobile robot navigation where several robot agents serve as learning processes. Robots learned to navigate an 11 x 11 world contains obstacles and boundaries choosing the optimum path to reach a target. Simulated experiments based on 50 learning episodes showed an average improvement of 17.02% while measuring the number of learning steps required reaching definite optimality and an average improvement of 32.98% for convergence to near optimality by using two robots compared with the $Q(\lambda)$ algorithm [1, 2].

KEY WORDS
Robot simulation, reinforcement learning, and navigation

1. Introduction

Reinforcement learning (RL) used in this work is learning through direct experimentation [3, 4]. It does not assume the existence of a teacher that provides training examples. Instead, in RL experience is the only teacher. The learner receives signals (reinforcements) from the process by getting an indication about how well it is performing the required task. These signals are usually associated to some dramatic condition - *e.g.*, accomplishment of a subtask (reward) or complete failure (punishment), and the learner's goal is to optimize its behavior based on some performance measure (maximization of a cost function). 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 to these events [5].

Navigation task can be broken down into three parts [6]: (i) localization, the process of figuring out where the robot is; (ii) mapping, the process whereby the robot builds a model of its environment, and (iii) planning, the process of figuring out how the robot can get to other places. Robots are inherently uncertain about the state of their environments. Uncertainty arises from sensor limitations, noise and the fact that real-world environment is unpredictable. Learning to navigate in realistic environments requires novel algorithms for identifying important events and find efficient action policies.

In [7] navigation learning of a miniature mobile robot is described. The robot equipped with vision capabilities learns to navigate a maze using several RL-based algorithms. [8] demonstrate a two mode Q-learning on a humanoid robot at a 17 x 17 maze for improving Q-learning performance. A RL algorithm for accelerating acquisition of new skills by real mobile robot is described in [9]. The algorithm speeds up Q-learning by applying memory-based sweeping [10] and was tested within an image-based visual servoing framework on an ActivMedia PeopleBot mobile robot for a docking task. A solution for robotic docking based on neural and reinforcement is presented in [11]. The solution was achieved partially by training of a value function unit and four motor units via RL. [12] describe a collaborative process enabling a robotic learner to acquire concepts and skills from human examples. During teaching the robot requires to perform tasks based on human instructions.

The robot executes the tasks and by incorporating feedback its hypothesis space is converged. With Q-learning approach, the robot learns a button pushing task.

Although Q-learning and $Q(\lambda)$ were used in many fields of robotics [*e.g.*, 13, 10, 14, 15, 16, 17], the issue of acceleration of learning is still significant. It includes the acceleration of learning toward finding an optimal or close to optimal solution. In order to improve learning, we suggest the new $CQ(\lambda)$ algorithm. The $CQ(\lambda)$ algorithm proved to accelerate learning in a system composed of several similar learning processes.

The paper is organized as follows. In Section two, reinforcement learning theory is described. That includes description of the Q, $Q(\lambda)$ and the new $CQ(\lambda)$ algorithms. Section three describes the learning system. Section four demonstrates simulation experiments and results of applying the $CQ(\lambda)$ algorithm on a navigation problem. Concluding remarks follow in Section five.

2. Reinforcement Learning

The basic assumption on the study of RL 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 $s_t \in S$ is the state at time-step t and $a_t \in A$ is the action taken. Naturally, if the agent can faithfully observe the states of the process which by definition summarize all the relevant information about the process dynamics at a certain instant of time then its observations are Markovian. On the other hand, if the observations made by the agent are not sufficient to summarize all the information about the process, a non-Markovian condition takes place: $s_{t+1} = f(s_t, s_{t-1}, s_{t-2}, ..., a_t, a_{t-1}, a_{t-2}, ...)$ [5].

It is usually more adequate to express the dynamic of the process through a collection of conditional transition probabilities $P(s_{t+1} \in S \mid s_t \in S, a_t \in A)$. Of particular interest is the discounted infinite horizon formulation of the MDP (Markov Decision Process) problem. Given a finite set of possible actions $a \in A$, a finite set of process states $s \in S$, a stationary discrete-time stochastic process, modeled by transition probabilities $P(s_{t+1} \mid s_t, a_t)$ and a finite set of bounded reinforcements $r(s_t, a_t) \in R$, the agent tries to find out a stationary policy of actions $a_t^* = \pi^*(s_t)$ which maximizes the expected cost function (Eq. 1):

$$V^{\pi}(s_i) = \lim_{M \to \infty} E[\sum_{t=0}^{M} \gamma^t r(s_t, \pi(s_t))], \qquad (1)$$

for every state s_i. π indicates the dependency on the followed action policy, via the transition probabilities $P(s_{t+1} \mid s_t, \pi(s_t))$. The discount factor $0 \le \gamma < 1$ forces recent reinforcements to be more important than remote ones. The optimal cost function is presented in Eq. 2 where it is possible that there will be more than a single optimal policy π^* [18].

$$V^*(s_0) = \lim_{M \to \infty} E[\sum_{t=0}^{M} \gamma^t r(s_t, \pi^*(s_t))]. \qquad (2)$$

2.1. *Q*-Learning

The RL algorithm Q-learning [1], is modified here. 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. The advantage of Q-learning is that the update rule is model free as it is a rule that just relates Q values to other Q values. It does not require a 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 [19]. It is based on the action value measurement $Q(s_t, a_t)$, defined in Eq. 3:

$$\begin{aligned} Q(s_t, a_t) &= E[r(s_t, a_t) + \gamma V^*(s_{t+1})] = \\ &= r(s_t, a_t) + \gamma \sum_{s_{t+1} \in S} P(s_{t+1} \mid s_t, a_t) V^*(s_{t+1})], \end{aligned} \qquad (3)$$

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 the Bellman's optimality principle [20], Eq. 4 is derived:

$$\begin{aligned} Q(s_t, a_t) &= r(s_t, a_t) + \\ &\gamma \sum_{s_{t+1} \in S} P(s_{t+1} \mid s_t, a_t) \max_a Q(s_{t+1}, a_t). \end{aligned} \qquad (4)$$

These characteristics (max operator inside the expectation term and policy independence) allow an iterative process for calculating an optimal action policy via action values which is the essence of Q-learning. 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 Eq. 5 which describes a Q-learning one step:

$$Q_{t+1}(s_t, a_t) = (1-\alpha)Q_t(s_t, a_t) + \alpha[\, r(s_t, a_t) + \gamma \hat{V}_t(s_{t+1})], \quad (5)$$

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 stopping criterion is met (*e.g.*, robot reached target). $\alpha = 1$ gives full weight to new experiences. As α decreases, the Q-value is built up based on all experiences, and new unusual experience does not disturb the established Q-value much. The greedy action $\hat{V}_t(s_{t+1}) = \max_{a_t \in A}[Q_t(s_{t+1}, a_t)]$ is the best the agent performs when in state s_t, but for the initial stages of the learning process it uses randomized actions that encourages exploration of the state-space. Under some reasonable conditions [21] this is guaranteed to converge to the optimal Q-function [19].

2.2. CQ(λ)-Learning

$Q(\lambda)$ [1, 2] is a generalization of Q-learning that uses eligibility traces, $e(s_t, a_t)$: the one-step Q-learning is a particular case with $\lambda = 0$. The Q-learning algorithm learns quite slowly because only one time-step is traced for each action. To boost convergence of learning, the 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 [13]. An outline of the multi-step $Q(\lambda)$ learning algorithm [1, 2], which is based on the tableau version in [22], is shown in Fig. 1.

Initialize $Q(s,a) = 0$ and set eligibility trace $e(s,a) = 0$ for all (s,a)
Repeat (for each learning episode):
  Set initial state s_t and pick initial action a_t
  Repeat (for each step of episode):
    Take action a_t, observe reward r_t and the next state s_{t+1}
    Choose a_{t+1} for s_{t+1} with a certain policy (*e.g.*, ε-greedy)
    $a^* \leftarrow \arg\max_{b \in A} Q(s_{t+1}, b)$ (if a_{t+1} ties for the max, then $a^* \leftarrow a_{t+1}$)
    $\delta_t \leftarrow r_t + \gamma Q(s_{t+1}, a^*) - Q(s_t, a_t)$
    $e(s_t, a_t) \leftarrow e(s_t, a_t) + 1$
    For all (s_t, a_t):
      $Q(s_{t+1}, a_{t+1}) \leftarrow Q(s_t, a_t) + \alpha \delta_t e(s_t, a_t)$
      If $a_{t+1} = a^*$, then $e(s_t, a_t) \leftarrow \gamma\lambda e(s_t, a_t)$
        else $e(s_t, a_t) \leftarrow 0$
  $s_t \leftarrow s_{t+1}$; $a_t \leftarrow a_{t+1}$
until a stopping condition

Fig. 1. *Q(λ)* learning algorithm [1, 2]

The new $CQ(\lambda)$ (Fig. 2) algorithm objective is to accelerate learning in a system composed of several similar learning processes. It is based on that a state-action value of an agent or learning process is updated according to the maximal value within all other learning processes state-action values exist in a learning system. Alternatively if only two learning agents are involved such as a robot and human, state-action values known to the human are acquired by the robot learning function.

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_{t+1} for s_{t+1} using a certain policy (*e.g.*, softmax)
      $\delta_{i_t} \leftarrow r_{i_t} + \gamma_i Q_i(s_{i_{t+1}}, a_i^*) - Q_i(s_{i_t}, a_{i_t})$
      $e(s_t, a_t) \leftarrow e(s_t, a_t) + 1$
      For all (s_{i_t}, a_{i_t}):
        $Q_i(s_{i_{t+1}}, a_{i_{t+1}}) \leftarrow \max[Q_{i \in N}(s_{i_t}, a_{i_t}) + \alpha_{i_t} \delta_{i_t} e_i(s_{i_t}, a_{i_t})]$
        If $a_{i_{t+1}} = a_i^*$, then $e_i(s_{i_t}, a_{i_t}) \leftarrow \gamma\lambda e_i(s_{i_t}, a_{i_t})$
          else $e_i(s_{i_t}, a_{i_t}) \leftarrow 0$
    $s_{i_t} \leftarrow s_{i_{t+1}}$; $a_{i_t} \leftarrow a_{i_{t+1}}$
until a stopping condition

Fig. 2. *CQ(λ)*-learning algorithm

where δ_t is the temporal difference (TD) error which specifies how different the new value is from the old prediction and $e(s_t, a_t)$ 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, one is added to its eligibility trace [23].

In [24] it is stated that "the convergence of $Q(\lambda)$ is not assured anymore for $\lambda > 0$, but experiences show that the learning is faster". Several action selection policies are described in literature where the greedy policy [25] is always to choose the best action. Other policies (*e.g.*, "softmax" or "ε-greedy" [22]) are stochastic and based on choosing a suboptimal policy to explore the state-action space. In [22] it is stated that "although ε-greedy action selection is an effective and popular means of balancing exploration and exploitation [*e.g.*, 26], one drawback is that when it explores it chooses equally among all actions. This means that it is as likely to choose the worst-appearing action as it is to choose the next-to-best action. The obvious solution is to vary the action probabilities as a graded function of estimated value". One way to do that is to choose action a_t with probability that depends on the value of $Q(s,a)$. This is known as a "softmax" action selection. A common method is to use a Gibbs or Boltzmann distribution, where the probability of choosing

action a_t in state s_t is proportional to $e^{\beta_t Q(s_t,a_t)}$, i.e., in state s_t the agent chooses action a_t with probability

$$P(a_t \mid s_t) = \frac{e^{\beta_t Q(s_t,a_t)}}{\sum_{a_{t+1}} e^{\beta_t Q(s_t,a_t)}}, \quad (6)$$

where β_t is a positive parameter which specifies how randomly values should be chosen. When β_t is low, the actions are chosen about the same amount each other. As β_t increases, the highest valued actions are more likely to be chosen, and in the limit $\beta_t \to \infty$ the best action is always chosen [23].

In implementing the $CQ(\lambda)$ we adapt the attitude described in [4] for one agent and apply it for the collaborative agent attitude: learning rate of each process $i \in \{1, 2, \ldots N\}$, α_{i_t}, is set relatively high and is reduced adaptively over time. This is done independently for each state-action pair; number of times each state-action pair has been previously updated, $c_{i(s,a)}$, is calculated. The effective learning rate, $\alpha_{i(s,a)_t}$, is then determined from the initial rate by

$$\alpha_{i(s,a)_t} = \frac{\alpha_{i(s,a)_t}}{c_{i(s,a)} + 1}. \quad (7)$$

The principle is that the more and more knowledge about a certain state-action pair is gained, the less it is should be modified in response to any particular experience. Convergence was proved by [21].

3. Learning System

Robots' states include robot locations in a 11 x 11 two dimensional world (Fig. 3). The state s_t is defined by: $s_{(i,j)_t} = \{i, j\}$ where $i \in \{1, 2, \ldots, 11\}$ and $j \in \{1, 2, \ldots, 11\}$. Actions can be taken at each state are: traveling north, west, south and east. An action, a_t, is noted as $a_{(k)_t}$ where $k \in \{1, 2, 3, 4\}$. Rewards defined as $r_{(l)_t}$ where $l \in \{-1, 0, +1\}$. If the robot reaches target, the reward is +1. If it passes through an undesirable area such as an obstacle / boundary, the reward is -1. Otherwise, the reward is 0. Learning episode is a description of one session of reaching the target. Performance measures include: (i) $N_{i_{near-optmal_t}}$ - convergence to near optimality - mean of the last N_t learning step values, and (ii) $N_{i_{optimal_t}}$ - convergence to optimality - number of learning steps required to achieve the shortest path and repeat it infinite number of times where $i \in \{1, 2, \ldots N\}$ and N is the number of learning processes.

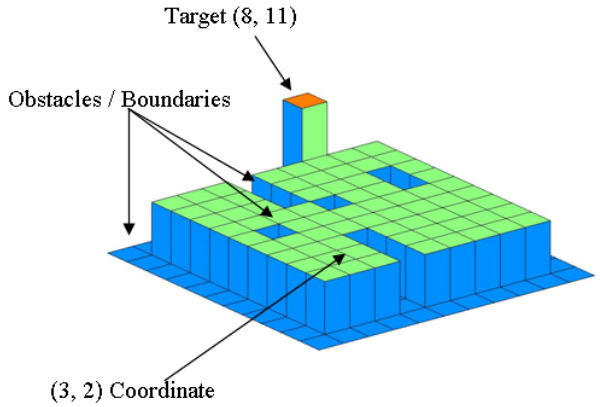


Fig. 3. An 11 x 11 two dimensional world

The world consists of three layers: (i) environmental cells - areas where the robot should navigate in its way to target; (ii) obstacles / boundaries - reduced cells and (iii) target - a higher cell (Fig. 3). Robots can move from a cell to any one of its four close neighbors with the restriction that they can not move out of the world. The task of the learning agents is to learn navigating the world by choosing optimal routes to a known target through environmental learning and knowledge sharing.

4. Experimental Results

The simulated system was evaluated using two experimental model sets and significance levels of mean steps to converge values were examined. In both models robot agents learn the world simultaneously and share knowledge. The first model contains two robot agents where the first learns according to $CQ(\lambda)$ and the second learns according to the $Q(\lambda)$ algorithms. The second model contains three robot agents where the first learns according to $CQ(\lambda)$ and the second and third learn according to the $Q(\lambda)$ algorithms.

At the first experimental model contains two robot agents. The first robotic agent noted as $i = 1$ learns according to the $CQ(\lambda)$ algorithm, i.e., gathers knowledge from the other robot whereas the second agent noted as $i = 2$ learns according to the $Q(\lambda)$ algorithm and does not gain knowledge from the first robot. The following parameters were set: $N = 2$, $N_t = 10$, $\alpha_1 = \alpha_2 = 0.95$ (initial values), $\gamma_1 = \gamma_2 = 0.99$, and $\lambda_1 = \lambda_2 = 0.5$ (Fig. 2). At the second experimental model contains three robot agents, the first robot noted as $i = 1$ learns according to the $CQ(\lambda)$ algorithm, i.e., gathers knowledge from the other two robots whereas the second and third robots noted as $i = 2$

and $i = 3$ learn according to the $Q(\lambda)$ algorithm and do not gain knowledge from the first robot. The model was set with the following parameters: $N = 3$, $N_t = 10$, $\alpha_1 = \alpha_2 = \alpha_3 = 0.95$ (initial values), $\gamma_1 = \gamma_2 = \gamma_3 = 0.99$, and $\lambda_1 = \lambda_2 = \lambda_3 = 0.5$ (Fig. 2).

Experiments based on 50 simulation runs were conducted for both models. An example for state-action values of one simulation run is given in Fig. 4.In both models number of learning episodes was set to 100, i.e., the algorithm stops after an agent navigates from starting point to target 100 times. For evaluating system performance, a specific state with coordinates (3, 2) was chosen. For this state, the optimal route length traveling to the target at coordinates (8, 11) is 14 steps (Fig. 3). Based on the results shown in Table 1, ten hypotheses were evaluated (Table 2).

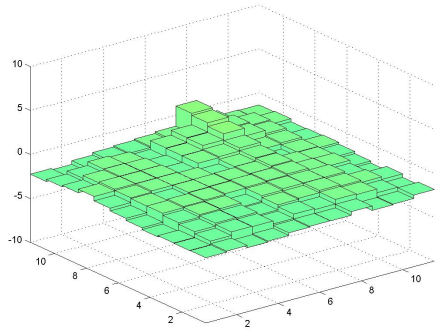


Fig. 4. 11 x 11 world state-action value map after 100 learning episodes

Null hypotheses H_{10} and H_{20} were rejected with P-values equal to $2.58 \cdot 10^{-5}$ and 0 respectively; namely; the mean coefficients of learning agents are not equal. Null hypothesis H_{30} was not rejected with P-value equals to 0.643 which results an equal means of the two learning agents. Null hypotheses H_{40} and H_{50} were rejected with P-values equals to $4.52 \cdot 10^{-6}$ and $6.59 \cdot 10^{-7}$ respectively concluding a difference between the means of the learning agents. The null hypothesis H_{60} was not rejected with P-value equals to 0.824; namely; the means of these two learning agents are equal. Null hypotheses H_{70} and H_{80} were rejected with P-values equals to $1.33 \cdot 10^{-15}$ and 0 respectively; namely; there is a difference between the means of the learning agents. Fig. 5 shows an example of one of the 50 simulation runs using two robots to converge (Table 1).

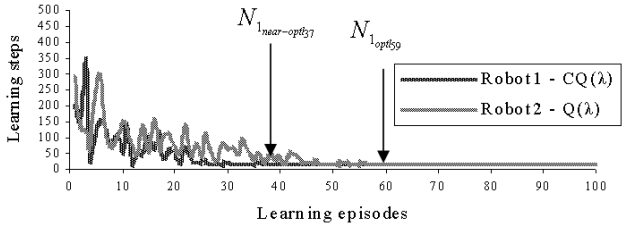


Fig. 5. Two learning agent convergence example of one simulation run

Fig. 6 shows an example of one of the 50 simulation runs using three robots to converge to optimal route (Fig. 3).

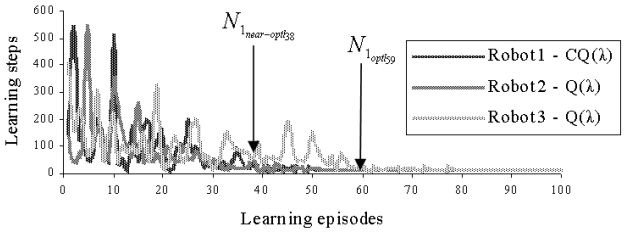


Fig. 6. Three learning agent convergence example of one simulation run

It is clearly seen both from hypotheses H_{10} through H_{80} evaluation and from Figs. 5 and 6 that the robot that uses the $CQ(\lambda)$ algorithm has superior learning performance while converging either to near optimality or to optimality with comparison to robots that use the $Q(\lambda)$ algorithm. It is also seen from Figs. 5 and 6 that initially a large number of steps is required to achieve learning because at that time the agents do not have enough knowledge about the world. As learning proceeds and the agents gain more knowledge number of steps drops dramatically.

For evaluating whether adding a third learning agent improves learning, hypotheses nine and ten were tested (Table 2). Null hypotheses H_{90} and H_{100} were not rejected with P-value equals to 0.81 and 0.436 respectively; namely; the means are equal. This means that an additional learning agent does not improve system learning significantly.

5. Discussion and Conclusions

A new type of a collaborative reinforcement learning algorithm has been developed in traditional navigation task. We demonstrated in simulation an acceleration of learning and the convergence to an optimum using collaboration between several learning agents. We presented the design and implementation via simulation of the $CQ(\lambda)$ algorithm applied on autonomous mobile robots for navigation in an 11 x 11 two dimensional world

contains obstacles and boundaries choosing the optimum path to reach a target. 50 simulation runs showed an average improvement of 17.02% while measuring the number of learning steps required reaching definite optimality and an average improvement of 32.98% for convergence to near optimality by using two robots compared with the $Q(\lambda)$ algorithm. Significant statistical difference was indicated for both convergence to optimality and convergence to near optimality while comparing two robots; the first uses $CQ(\lambda)$ and the second uses $Q(\lambda)$. While using three robots; the first uses the $CQ(\lambda)$ and the second and third use $Q(\lambda)$, we found that there is no statistical significant differences in both convergence to optimality and convergence to near optimality while comparing the $Q(\lambda)$-based robots. We found statistical significant differences in both convergence to optimality and convergence to near optimality while comparing the $CQ(\lambda)$-based robot to the other two. In addition we found that there is no statistical significant differences in both convergence to optimality and convergence to near optimality using $CQ(\lambda)$-based robot learns in a two robots environment or an $CQ(\lambda)$-based robot learns in three robots environment, i.e., we conclude that adding another robot in addition to two collaborative robots does not achieve significant improvement. We demonstrated: (i) superiority of the $CQ(\lambda)$ algorithm over the $Q(\lambda)$ and (ii) proved statistically that three learning processes have no significant advantage over two learning processes, thereby when $CQ(\lambda)$ will be tested on a real robot, only one additional learning process (*e.g.*, another robot or a human) will be applied.

Basically there is always a gap between simulated and real robot tests; the sensors and actuators are never ideal and are usually very difficult to model accurately and the operation environment is usually dynamic, etc. Future work includes a verification of the $CQ(\lambda)$ algorithm with real robots for: (i) learning the efficient lifting and shaking policies for emptying the contents of suspicious bags using a fixed arm robot [27, 28, 29, 30], and (ii) learning a navigation task using an Evolution Robotics ER-1 robot [31] when a human serves as an additional learning process.

6. Acknowledgements

This project was partially supported by the Paul Ivanier Center for Robotics Research and Production Management, Ben-Gurion University of the Negev.

References:

[1] C.J.C.H. Watkins, Learning from delayed rewards, *Ph.D. Dissertation*, Cambridge University, 1989.

[2] J. Peng, & R. Williams, Incremental multi-step Q-learning, *Machine Learning, 22*(1-3), 1996, 283-290.

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

[4] W.D. Smart, Making reinforcement learning work on real robots, *Ph.D. Dissertation*, Brown University, 2002.

[5] C. Ribeiro, Reinforcement learning agents, *Artificial Intelligence Review, 17*(3), 2002. 223-250.

[6] A. Howard, Probabilistic navigation: coping with uncertainty in robot navigation tasks, *Ph.D. Dissertation*, Department of Computer Science and Software Engineering, University of Melbourne, Australia, 1999.

[7] B. Bhanu, P. Leang, C. Cowden, Y. Lin, & M. Patterson, Real-time robot learning. *Proc. of the 2001 IEEE International Conf. on Robotics and Automation*, Seoul, Korea, 2001, 491-498.

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

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

[10] C. Touzet, *Q-learning for robots, the handbook of brain theory and neural networks*, (Cambridge, MA: M. Arbib editor, MIT Press, 2003). 934-937.

[11] C. Weber, S. Wermter, & A. Zochios, Robot docking with neural vision and reinforcement, *Knowledge-based systems, 17*(2-4), 2004. 165-72.

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

[13] W. Zhu, & S. Levinson, Vision-based reinforcement learning for robot navigation, *Proc. of the international joint conf. on neural network, 2*, Washington DC., 2001, 1025-30.

[14] A.F. Massoud, & L. Caro, Fuzzy neural network implementation of Q(λ) for mobile robots. *WSEAS Transaction on Systems, 3*(1), 2004.

[15] E. Menegatti, G. Cicirelli, C. Simionato, A. Distante, & E. Pagello. Reinforcement learning based omnidirectional vision agent for mobile robot navigation, *Workshop Robotica del IX Convegno della Associazione Italiana Intelligenza Artificiale (AI*IA04)*, 2004.

[16] Y. Dahmani, & A. Benyettou, Seek of an optimal way by Q-learning, *Journal of Computer Science, 1*(1), 2005, 28-30.

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

[18] S.M. Ross, *Introduction to stochastic dynamic programming* (New York: Academic Press, 1983).

[19] W.D. Smart, & L. Kaelbling, Practical reinforcement learning in continuous spaces, *Proc. of the 17th International Conf. on Machine Learning*, 2002.

[20] R. Bellman, & R. Kalaba, *Dynamic programming and modern control theory* (New York: Academic Press Inc., 1965).
[21] C.J.C.H. Watkins, & P. Dayan, Q-learning. *Machine Learning, 8*, 1992, 279-292.
[22] R.S. Sutton, & A.G. Barto, *Reinforcement learning: an introduction* (Cambridge, MA: MIT Press, 1998).
[23] A.K. MackWorth, D. Poole, & R.G. Goebel, *Computational intelligence: a logical approach* (Oxford University Press, 1998).
[24] P.Y. Glorennec. Reinforcement Learning: an overview. *European Symposium on Intelligent Techniques*, Aachen, Germany, 2000.
[25] S. Nason, & J.E. Laird, Soar-RL: integrating reinforcement learning with soar, *Proc. of the International Conf. on Cognitive Modeling*, 2004, 51-59.
[26] M. Guo, Y. Liu, & J. Malec, A new Q-learning algorithm based on the metropolis criterion, *IEEE Transactions on Systems, Man, and Cybernetics - Part B: Cybernetics, 34*(5), 2004, 2140-2143.
[27] Y. Edan, U. Kartoun, & H. Stern. Cooperative human-robot learning system using a virtual reality telerobotic interface, *Conf. on Advances in Internet Technologies and Applications*, Purdue University, West Lafayette, Indiana, U.S.A., 2004.
[28] U. Kartoun, H. Stern, & Y. Edan, Virtual reality telerobotic system. *Proc. of the 4th International Conf. on e-Engineering and Digital Enterprise Technology*, Leeds Metropolitan University, Yorkshire, U.K., 2004, 445-454.
[29] U. Kartoun, H. Stern, & Y. Edan, Bag classification using support vector machines. *The 9th On-line World Conf. on Soft Computing in Industrial Applications*, 2004.
[30] http://www.motoman.co.uk/UP20.htm/
[31] http://www.evolution.com/er1/

Table 1. Simulation results

	Experimental model set I		Experimental model set II		
	Robot1	Robot2	Robot1	Robot2	Robot3
Learning strategy	$CQ(\lambda)$	$Q(\lambda)$	$CQ(\lambda)$	$Q(\lambda)$	$Q(\lambda)$
Mean / standard deviation of steps to converge to optimality	56.8 / 13.6	68.4 / 14.0	55.2 / 14.1	68.9 / 13.9	69.4 / 14.4
Mean / standard deviation of steps to converge to near optimality	37.7 / 7.2	56.2 / 10.1	36.8 / 6.5	56.2 / 15.6	56.6 / 11.2

Table 2. Evaluation hypotheses

Evaluation within experimental model set I learning agents	H_{10} : There is no difference between convergence to optimality of the two learning agents. H_{11} : There is a significant difference between the two learning agents.
	H_{20} : There is no difference between convergence to near optimality of the two learning agents. H_{21} : There is a significant difference between the two learning agents.
Evaluation within experimental model set II learning agents	H_{30} : There is no difference between convergence to optimality of $Q(\lambda)$ learning (Robot2) and $Q(\lambda)$ learning (Robot3). H_{31} : There is a significant difference between the two learning agents.
	H_{40} : There is no difference between convergence to optimality of $CQ(\lambda)$ learning (Robot1) and $Q(\lambda)$ learning (Robot2). H_{41} : There is a significant difference between the two learning agents.
	H_{50} : There is no difference between convergence to optimality of $CQ(\lambda)$ learning (Robot1) and $Q(\lambda)$ learning (Robot3). H_{51} : There is a significant difference between the two learning agents.
	H_{60} : There is no difference between convergence to near optimality of $Q(\lambda)$ learning (Robot2) and $Q(\lambda)$ learning (Robot3). H_{61} : There is a significant difference between the two learning agents.
	H_{70} : There is no difference between convergence to near optimality of $CQ(\lambda)$ learning (Robot1) and $Q(\lambda)$ learning (Robot2). H_{71} : There is a significant difference between the two learning agents.
	H_{80} : There is no difference between convergence to near optimality of $CQ(\lambda)$ learning (Robot1) and $Q(\lambda)$ learning (Robot3). H_{81} : There is a significant difference between the two learning agents.
Evaluation between experimental sets I and II	H_{90} : There is no difference between convergence to optimality of using two or three $CQ(\lambda)$ robots. H_{91} : There is a significant difference between the two learning models.
	H_{100} : There is no difference between convergence to optimality of using two or three $CQ(\lambda)$ robots. H_{101} : There is a significant difference between the two learning models.

Cooperative Human-Robot Learning System using a Virtual Reality Telerobotic Interface

Yael Edan, Uri Kartoun, Helman Stern

Abstract— A cooperative human-robot learning system for remote robotic operations using a virtual reality (VR) interface is presented. The paper describes the overall system architecture, and the VR telerobotic system interface. Initial tests using on-line control through the VR interface for the task of shaking out the contents of a plastic bag are presented.
The system employs several state-action policies. System states are defined by: type of bag, status of robot, and environment. Actions are defined by initial grasping point, lift and shake trajectory. A policy is a set of state-action pairs to perform a robotic task. The system starts with knowledge of the individual operators of the robot arm, such as opening and closing the gripper, but it has no policy for deciding when these operators are not appropriate, nor does it have knowledge about the special properties of the bags. An optimal policy is defined as the best action for a given state that is learned from experience and human guidance. A policy is found to be beneficial if a bag was grabbed successfully and all its contents extracted.

Index Terms— telerobotics, human-robot collaboration, robot learning

1. INTRODUCTION

A telerobot is defined as a robot controlled at a distance by a human operator (HO). Telerobotic devices are typically developed for situations or environments that are too dangerous, uncomfortable, limiting, repetitive, or costly for humans to perform [Durlach and Mavor, 1995]. Applications include: underwater [e.g., Hsu et al., 1999], space [e.g., Hirzinger et al., 1993], and medical [e.g., Kwon et al., 1999; Sorid and Moore, 2000].

One of the difficulties associated with teleoperation is that the HO is remote from the robot; hence, feedback may be insufficient to correct control decisions. Therefore, the HO usually acts as a supervisor, defining the goals, and current plans, getting back information about accomplishments, and problemes; while the robot executes the task based on information received from the HO plus its own artificial sensing and intelligence [Earnshaw et al., 1994].

Autonomous robots are designed to build physical systems that can accomplish useful tasks without human intervention [Manesis et al., 2002]. To accomplish a given task, the robot must collect sensory information concerning its external dynamic environment, make decisions, and take actions. Full autonomy is usually complicated to achieve in unstructured environments [Hagras and Sobh, 2002].

For telerobotic control, decision-making can be performed by a combination of knowledge based autonomous procedures, sensor based autonomous procedures, learning procedures and HO procedures. Learning enables the improvement of autonomous operation by allowing the system to acquire and adapt their behavior to new unknown conditions. Learning robots have resulted in improved performance and are especially suitable for changing and dynamic environments [Aycard and Washington, 2000; Asoh et al., 2001; Bhanu et al., 2001; Carreras et al., 2002]. However, their design is complicated and costly.

Humans can easily adapt to unpredictability task environments due to their superior intelligence and perceptual abilities. Introducing a HO into the system can help improve its performance and simplify the robotic system making the telerobotic system a viable option [Rastogi et al., 1995]. Robot performance can be improved by taking advantage of human skills (e.g., perception, cognition) and benefiting from human advice and expertise. To do this, robots must function as active partners instead of as passive tools. They should have more freedom of action, be able to initiate interaction with humans, instead of merely waiting for human commands [Fong et al., 2001]. Several systems exist in which a human and a robot work as partners, collaborating to perform tasks and to achieve common goals [Scárdua et al., 2000; Asoh et al., 2001]. Instead of a supervisor dictating to a subordinate, the human and the robot engage in dialogue to exchange ideas, to ask questions, and to resolve differences [Fong et al., 2001]. In particular, if a robot is not treated as a tool, but rather as a partner, by incorporating learning, the system can continuously improve.

Interface design has a significant effect on the way people operate a robot [Preece et al., 1994]. Examples for using graphical models to allow users to control robots off-line through the Web [Hirukawa. and Hara, 2000] can be found in the RobotToy research [Sorid and Moore, 2000], the KhepOnTheWeb project [Michel et al., 1997], the WITS (Web Interface for Telescience) project [Backes et al., 1999], the Tele-Garden project [Goldberg et al., 1995], the University of Western Australia's Telerobot experiment [Taylor and Trevelyan, 1995] and the PumaPaint project [Stein, 2000].

Virtual reality (VR) is a high-end human-computer interface allowing user interaction with simulated environments in real-time and through multiple sensorial channels [Burdea, 1999]. The increased interaction enables operators to perform tasks on remote real worlds, computer-generated

• *All authors are with the Dept. of Industrial Engineering and Management, Ben-Gurion University of the Negev, Beer Sheva, Israel 84105. E-mails: yael, kartoun, helman@bgumail.bgu.ac.il*

Manuscript received April 30, 2004.

worlds or any combination of both resulting in improved performance [Hine et al., 1995; Burdea, 1999; Lee et al., 2000; Heguy et al., 2001].

The objective of this paper is to present the concept of a human-robot learning system for remote robotic operations using a VR interface. Problem definition and notation are presented in the next section. The overall system architecture and VR telerobotic system interface is described in sections 3 and 4, respectively. To test the system an advanced VR telerobotic bag shaking system is proposed. The task is defined to observe the position of an unknown suspicious bag located on a platform, classify it, grasp it with a robot manipulator and shake out its contents on a table or into a nearby container. Initial tests using on-line control through the VR interface for shaking out the contents of a plastic bag are presented in section 5. Section 6 concludes the paper.

2. PROBLEM DEFINITION AND NOTATION

The system is defined by $\Sigma = [R, O, E]$ where R is a robot, O an object, and E an environment. Let F be a set of features or patterns representing the physical state of Σ. Let ψ be a mapping from Σ to F. T is a task performed on O, using R, within the environment E to meet a goal G. The performance of the task T is a function of a set of actions, A, for each physical state of Σ. The state of the system, represented by an element of F, is denoted as S. Let a policy P be a set of state-action pairs, {S, A}. Let the performance measure be Z(F, P).

2.1 Goal

The goal, G, is to classify a bag correctly, grab it successfully and empty its contents onto a table or into a collection container in minimum time.

2.2 Task

The task, T, is to observe the position of an unknown bag (e.g., plastic bag, briefcase, backpack, or suitcase) located on a platform, grasp it with a robot manipulator and shake out its contents on a table or into a nearby container. It is assumed that all clasps, zippers and locks have already been opened by another robotic operation. The system is trained first for identifying several bag classes, but it has no a-priori knowledge regarding to efficient grasping and shaking policies. The system learns this knowledge from experience and from human guidance.

2.3 System

Robot

Several states will be defined for the robot, R.

Robot states, S_R, include:

{home, idle, performing a task}.

Object

The system will contain different types of bags (e.g., plastic bag, briefcase, backpack, suitcase). Different geometrical attributes will be defined for each bag, O, to identify the bag. The state of the object So is defined by:

$S_{bag\text{-}class}$ = {plastic bag, briefcase, backpack, suitcase, not recognized}.

$S_{bag\text{-}condition}$ = {open, close, orientation).

$S_{bag\text{-}contents}$ = {how many}.

Environment

The robotic environment, E, contains a platform on which the inspected bag is manipulated, light sources, and extraneous objects such as undesirable human hand.

Environmental states, S_E, include:

$S_{obstructions}$ = {how many, positions}

$S_{illumination}$ = {light, day, night}

2.4 Features

Let F be a set of features or patterns representing the state of Σ. F may include bag classifications, robot position and orientation and environmental conditions.

2.5 Mapping

Let ψ be visual mapping function which obtains a digital image I of the system Σ and extracts a set of representative features denoted as F.

2.6 Actions

Actions, A, are command instructions such as grasping, shaking, etc.

2.7 Policy

Carrying out the task involves a policy P. Given F of $\Sigma = [R, O, E]$, a policy $P = \{(F, A)\}$ is a set of state-action pairs.

2.8 Performance measures

The performance measures, Z(F, P), include:

- Classification accuracy.
- Whether a bag was grabbed successfully or not.
- Quality - how empty the bag is.
- Time to completely empty the contents of a bag.
- Number of learning iterations to complete the task.
- Human intervention rate.
- Abort task rate.

3. METHODOLOGY

3.1 System architecture

The system architecture (Fig. 1) consists of state-action classifiers that receive inputs from a vision system, a robotic system and a VR interface.

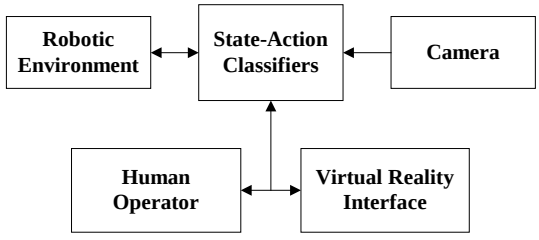


Fig. 1. System architecture

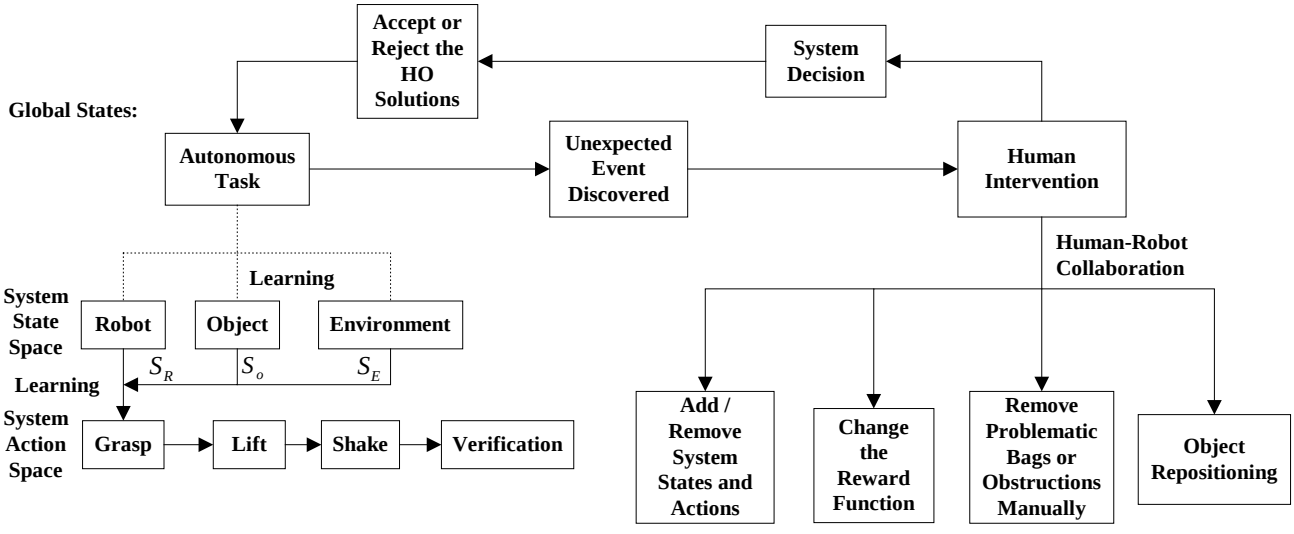


Fig. 2. System operation description

When the state-action classifiers have little or no knowledge about how to proceed on a task, they can try to obtain that knowledge through advice from the human operator (HO). This is achieved through human-robot collaboration. In this manner the HO can affect and change parameters of the learning algorithms on-line through the VR interface (e.g., by suggesting new actions such as shaking plans, intervening in case of misclassification).

In the proposed system, the search space in which to discover a successful solution may be quite large. To make this search tractable, the system should accept advice from the HO through its search. This requires the ability to identify when knowledge is needed, as well as, to provide the necessary problem-solving context for the HO so that supplying the knowledge is easy. It must also be possible for the system to proceed without advice when the HO is unavailable. If guidance is available, the system will utilize its own strategies together with advice to quickly construct a solution.

3.2 System stages

The overall system operation is described in Fig. 2. The system operates through the following stages. Each stage is evaluated using the defined performance measures and results in success or failure. Human-intervention is enabled based on the performance measures.

A. Classification:

Description: Determine the state of the system (S) using visual and possibly tactile sensors. This may involve positioning a robot on board camera ("hand-in-eye") to view the object position and the surrounding environment. It may also involve touching the object with a tactile sensor to assess its composition (soft cloth, plastic, hard plastic, etc.). Classification is performed by image processing [Kartoun, 2003].

Success: A bag was classified correctly.

Failure: Required for avoiding the object repositioning stage.

Human Intervention: Required for setup - put manually various bags on the robot workspace. If failure occurs, HO gives correct classification.

Performance measure: Classification accuracy.

B. Grasping:

Description: The robot grasps the object.

Success: A bag was grasped successfully.

Failure: A bag was not grasped optimally or was not grasped at all.

Human Intervention: HO gives a correct set of possible grasping points.

Performance measure: Whether a bag was grasped successfully or not.

C. Repositioning:

Description: Re-arranging the position of the object to prepare it for easier grasping.

Success: A bag was grasped successfully.

Failure: A bag was not grasped at all.

Human Intervention: HO repeats this stage until the object is grasped successfully.

Performance measure: Whether a bag was grasped successfully or not.

D. Lift and shake:

Description: The robot lifts the object above the table or above a collection container and shakes out its contents.

Success: Always successful.

Human Intervention: Not required.

Performance measure: Time to completely empty the contents of a bag.

E. Verification:

Description: The system tries to verify if all the contents have been extracted.

Success: 1) If the number of items fell from a bag is higher than a pre-determined threshold for a shaking policy and 2) time to empty the contents of a bag is lower than a predetermined threshold.

Failure: 1) Not all items fell out; 2) time to empty the contents is too slow and 3) abort task rate is too high.

Human Intervention: 1) Suggest new grasping points through VR interface and 2) suggest new lifting and shaking policies through VR interface.
Performance measures: 1) Quality - how empty the bag is; 2) time to completely empty the contents of a bag and 3) abort task rate.

3.3 Learning

The reinforcement learning algorithm [Kaelbling et al., 1996], Q-learning [Watkins, 1992] will be employed. 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. The advantage of Q-learning is that the update rule is policy free as it is a rule that just relates Q values to other Q values. It does not require a 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. 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(S) the action that locally maximizes the action-value. That is, P(S)←A, such that Q(S, A) = max Q(S, A) where ties are broken arbitrarily. The robot then enters a cycle of acting and policy updating.
First, the robot senses the current state, S = {S_R, S_O, S_E} where S_R is the state of the robot, S_O is the state of the object and S_E is the state of the environment. It then selects an action A to perform next. An action consists of a set of coordinates of possible grasping points, lift trajectories (e.g., manipulate the robotic arm 60cm above the collection container and with 30 degrees left) and shaking trajectories (e.g., manipulate the robotic arm 10cm horizontally and 15cm vertically in 3m/s speed for 5 cycles). Most of the time, this action will be the action specified by the system's policy P(S), but occasionally the system will choose a random action (choosing an action at random is a particularly simple mechanism for exploring the environment). Exploration is necessary to guarantee that the system will eventually learn an optimal policy. The system performs the selected action and notes the immediate reward r and the resulting state S'. The reward function is probabilistic and depends on the number of items falling out of the bag during the lifting and shaking operations. The action-value estimate for the state-action pair {S, A} is then updated.

4. VIRTUAL REALITY TELEROBOTIC INTERFACE

4.1 Physical experimental setup

In this work, a five degrees of freedom (DOF) articulated robot is controlled through a VR interface from a remote site for performing various tasks. The articulated robot is a "CRS-A255" robot system that consists of robot arm and controller (Fig. 3). The proposed VR telerobotic system contains a human operator (HO), VR web-based control interface, Internet access method, a remote server, a robot and its controller, and visual sensory feedback.

Fig. 3. Experimental setup

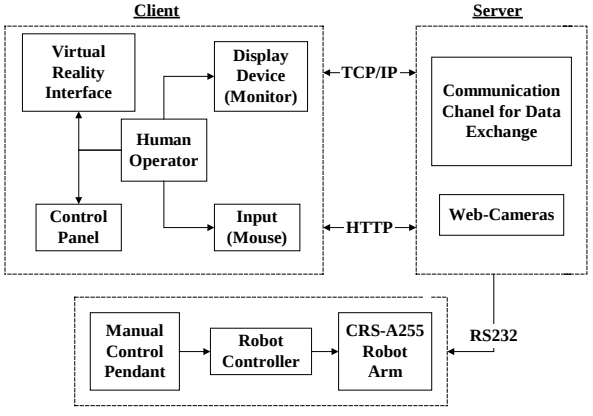


Fig. 4. Client-server communication architecture

The system is a client-server application (Fig. 4). The server contains a frame grabber connected to a camera mounted over the workspace. Robot control software is also located in the server. Two additional web-cameras are mounted over the workspace for visual feedback of the scene to the HO (the client). From the client site, the HO can take control over the robot through a VR interface.

4.2 User and control interface

A control interface was developed to enable HO interaction with the robot (Fig. 5). The interface includes a graphical model of the "CRS-A255" robot, two camera views of the real robot (overall and close views), a checkerboard on a table, and a world coordinate diagram that shows the x, y and z directions in the 3D scene.
The system has six different operational stages controlled through predefined control panels [Kartoun et al., 2004]. These are: changing the real robots speed, showing a 3D-grid that contains spatial locations which the robot gripper moves to when selected, selecting the viewing aspect of the VR model, planning shaking policies, planning off-line paths for transferring to the real robot, and on-line simultaneous control (in real-time) of the VR real robots.

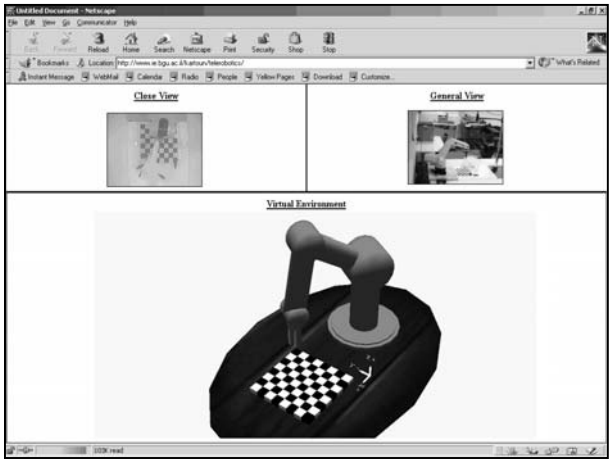


Fig. 5. Web-based interface (camera views, and VR model)

4.3 VR model

The VR model was developed using "3D-Studio-Max" and "Alice" softwares [Kartoun, 2003]. "Alice", rapid prototyping software for creating interactive computer graphics applications was chosen to be the VR software [Conway et al., 1993]. It is designed to enable rapid development and prototyping of interactive graphics applications and uses "Python" as the language for writing its scripts.

4.4 Communication

The HO communicates with the server, connected to a robotic arm through a web-browser. Commands sent from the VR client are transmitted through TCP/IP to the server that extracts them and updates the real robot.

4.5 System calibration

The inverse kinematics (IK) equations were solved using a closed form analytical solution [Lander, 1998; Craig, 1989; McKerrow, 1991]. It has the benefit of being an exact solution and very fast to calculate. For the VR robot, an IK algorithm was implemented to determine the joint angles required to reach an end point location by supplying the (x, y, z) coordinates. A transformation matrix, providing a 1 to 1 mapping between the VR and real robots is estimated from corresponding pairs of 32 intersection points on the checkerboard appearing in the VR and the real environments. The estimate is based on the method of aligning a pair of shapes, which uses a least-squares approximation [Cootes et al., 1992]. Given the calibrated transformation matrix, an experiment to determine the transformation error was performed. The experiment starts with controlling the VR robot arm through the VR interface. The coordinates of 32 points (x_{vr} , y_{vr}), selected from checkerboard intersections, were set as test points. These points were inserted into the inverse kinematic equations to obtain (x_r , y_r) which were sent to the real robot. A pen inserted into the real robot's gripper marked its controlled position. The coordinates of the robot's pen on the checkerboard intersection points ($x_{r,m}$, $y_{r,m}$) in the real environment were measured manually by a ruler. The average transformation error between the robot's pen positions, and the desired points in the real checkerboard was found to be 3mm [Kartoun et al., 2004].

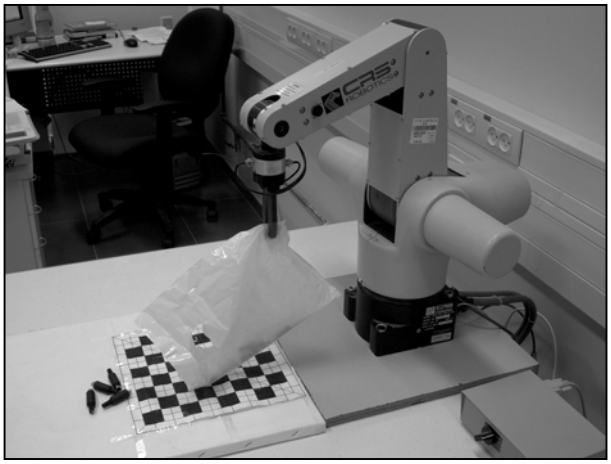
5. System Test Using VR On-Line Control

Initial system testing was conducted using on-line control through the VR interface for the task of shaking out the contents of a plastic bag. The HO views the images of the robotic workspace in the client browser, and commands the robot by selecting points in the 3D VR scene. Using the calibrated mapping of points in the VR scene to the real workspace, the robot is controlled to perform the task. A view of the experiment to empty the contents of a plastic bag onto the platform is shown in Fig. 6. It is assumed that the bag contains ten identical electronic components known in advance. An inexperienced operator performed ten identical experiments, and performance times (the amount of time it takes to extract all the objects from the bag) were recorded. The learning curve of task completion time was reduced quickly reaching standard time (330 seconds) after 5 to 6 trials (Fig. 7).

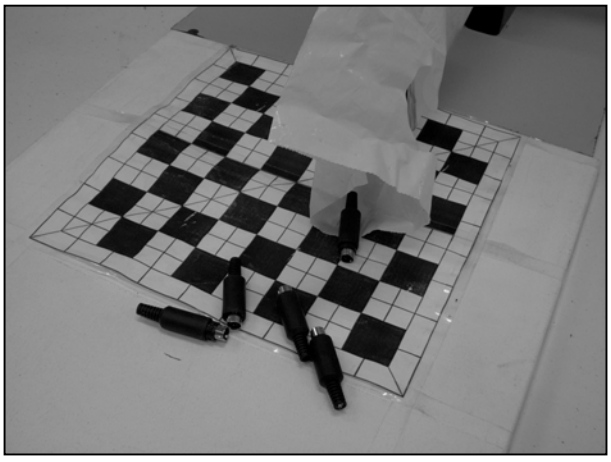
6. Conclusion

Human-robot collaboration is unnecessary as long as a telerobotic system can adapt to new states and unexpected events. However, when an autonomous system fails, incorporating learning using human operator (HO) supervision and interventions can achieve improved performance.

In this work we have described the design, implementation and testing of a real-time VR-telerobotic web-based system. Visual feedback is inserted into the human interface via two independent web-cameras. A transformation matrix is used to map the VR scene into the real one. The system allows a HO to: (a) perform off-line path planning by manipulating an object in a VR robotic scene, (b) perform on-line control by indirectly controlling the real robot through manipulation of its VR representation in real-time. Initial testing using on-line control indicated rapid learning, reaching standard time (330 seconds) within 5 to 6 trials. Future research will include integration of learning as presented in the above so as to improve performance. A cooperative human-robot learning system for remote robotic operations using this VR interface is underway.



(a) Overall view



(b) Close view

Fig. 6. “CRS-A255” robot, plastic bag, platform and components

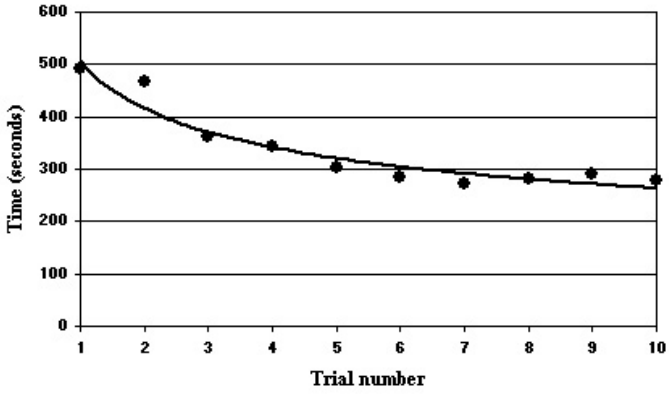


Fig. 7. Learning curve experimental results

REFERENCES

[1] H. Asoh, N. Vlassis, Y. Motomura, F. Asano, I. Hara, S. Hayamizu , K. Ito, T. Kurita, T. Matsui, R. Bunschoten and J.A. Kröse Ben. 2001. Jijo-2: An Office Robot that Communicates and Learns. IEEE Intelligent Systems. Vol. 16. Num 5. pp. 46-55.

[2] O. Aycard and R. Washington. 2000. State Identification for Planetary Rovers: Learning and Recognition. Proceedings of the IEEE International Conference on Robotics and Automation. pp. 1163-1168.

[3] P. G. Backes, K. S. Tso, and G. K. Tharp. 1999. The Web Interface for Telescience, Presence, MIT Press. Vol. 8. pp. 531-539.

[4] B. Bhanu, P. Leang, C. Cowden, Y. Lin, and M. Patterson. 2001. Real-Time Robot Learning. International Conference on Robotics and Automation. Seoul. Korea.

[5] G. Burdea. 1999. Invited Review: The Synergy between Virtual Reality and Robotics. IEEE Transactions on Robotics and Automation. Vol. 15. Num. 3. pp. 400-410.

[6] M. Carreras, P. Ridao, J. Batlle and T. Nicosevici. 2002. Efficient Learning of Reactive Robot Behaviors with a Neural-Q Learning Approach. IEEE International Conference on Automation, Quality and Testing. Romania.

[7] M. J. Conway, R. Pausch, R. Gossweiler and T. Burnette. 1993. Alice: A Rapid Prototyping System for Building Virtual Environments. University of Virginia, School of Engineering.

[8] T. F. Cootes, C. J. Taylor, D. H. Cooper and J. Graham. 1992. Training Models of Shape from Sets of Examples. Proceedings of the British Machine Vision Conference. pp 9-18.

[9] J. J. Craig. 1989. Introduction to Robotics: Mechanics and Control. Addison-Wesley.

[10] I. N. Durlach and S.N. Mavor. 1995. Virtual Reality: Scientific and Technological Challenges. National Academy Press. Washington DC.

[11] R. A. Earnshaw, M. A. Gigante and H. Jones. 1994. Virtual Reality Systems. Academic Press Limited.

[12] T. W. Fong, C. Thorpe and C. Baur. 2001. Collaboration, Dialogue, and Human-Robot Interaction. Proceedings of the 10th International Symposium of Robotics Research, Lorne, Victoria, Australia, Springer-Verlag.

[13] K. Goldberg, J. Santarromana, G. Bekey, S. Gentner, R. Morris, J. Wiegley and E. Berger. 1995. The Telegarden. Proceedings of ACM SIGGRAPH.

[14] H. Hagras and T. Sobh. 2002. Intelligent Learning and Control of Autonomous Robotic Agents Operating in Unstructured Environments. Information Science Journal. Elsevier Science Inc. Vol. 145. Issue 1-2. pp. 1-12.

[15] O. Heguy, N. Rodriguez, H. Luga, J. P. Jessel and Y. Duthen. 2001. Virtual Environment for Cooperative Assistance in Teleoperation. The 9th International Conference in Central Europe on Computer Graphics, Visualization and Computer Vision.

[16] B. Hine, P. Hontalas, T.W. Fong, L. Piguet, E. Nygren and A. Kline. 1995. VEVI: A Virtual Environment Teleoperations Interface for Planetary Exploration. SAE 25th International Conference on Environmental Systems.

[17] H. Hirukawa and I. Hara. 2000. The Web's Top Robotics. IEEE Robotics and Automation Magazine. Vol. 7. Num. 2.

[18] G. Hirzinger, B. Brunner, J. Dietrich and J. Heindl. 1993. Sensor-Based Space Robotics-ROTEX and its Telerobotic Features. IEEE Transactions on Robotics and Automation. Vol. 9. Num. 5. pp. 649-663.

[19] L. Hsu, R. Costa, F. Lizarralde and J. Soares. 1999. Passive Arm Based Dynamic Positioning System for Remotely Operated Underwater Vehicles. IEEE International Conference on Robotics and Automation. Vol. 1. pp. 407-412.

[20] L. P. Kaelbling, M.L. Littman and A.W. Moore. 1996. Reinforcement Learning: A Survey. Journal of Artificial Intelligence. Vol. 4. pp. 237-285.

[21] U. Kartoun. 2003. A Human-Robot Collaborative Learning System Using a Virtual Reality Telerobotic Interface. Ph.D. proposal, Dept. of Industrial Engineering and Management, Ben-Gurion University of the Negev, Beer Sheva 84105, ISRAEL.

[22] U. Kartoun, Y. Edan and H. Stern. 2004. Virtual Reality Telerobotic System. e-ENGDET 4th International Conference on e-Engineering and Digital Enterprise Technology, U.K.

[23] D. Kwon, K. Y. Woo and H. S. Cho. 1999. Haptic Control of the Master Hand Controller for a Microsurgical Telerobot System. IEEE International Conference on Robotics and Automation. Vol. 3. pp. 1722-1727.

[24] J. Lander. 1998. Oh My God, I Inverted Kine. Game Developer Magazine. Vol. 9. pp. 9-14.

[25] K. H. Lee, H. S. Tan and T. J. Lie. 2000. Virtual Operator Interface for Telerobot Control. Proceedings of the Sixth International Conference on Modern Industrial Training. Beijing. pp. 250-258.

[26] S. A. Manesis, G. N. Davrazos and N. T. Koussoulas. 2002. Controller Design for Off-tracking Elimination in Multi-articulated vehicles. 15th IFAC World Congress, Spain.

[27] P. J. McKerrow. 1991. Introduction to Robotics, Addison-Wesley.

[28] O. Michel, P. Saucy and F. Mondada. 1997. KhepOnTheWeb: an Experimental Demonstrator in Telerobotics and Virtual Reality, Proceedings of the International Conference on Virtual Systems and Multimedia, IEEE VSMM'97. pp. 90-98.

[29] J. Preece, Y. Rogers., H. Sharp, D. Benyon, S. Holland and T. Carey 1994. Human-Computer Interaction. Adison-Welsey Publishing Company.

[30] A. Rastogi, P. Milgram and J. J. Grodski. 1995. Augmented Telerobotic Control: a Visual Interface for Unstructured Environments. University of Toronto and Defense and Civil Institute of Environmental Medicine.

[31] L. Scárdua, A. H. Reali-Costa and J. J. da Cruz. 2000. Learning to Behave by Environment Reinforcement. RoboCup-99: Robot Soccer World Cup III. Lecture Notes in Artificial Intelligence. Berlin. Vol. 1856. pp. 439-449.

[32] D. Sorid and S. K. Moore. 2000. The Virtual Surgeon. IEEE SPECTRUM. pp. 26-39.

[33] M. Stein. 2000. Interactive Internet Artistry, Painting on the World Wide Web with the PumaPaint Project. IEEE Robotics and Automation Magazine. Vol. 7. Num. 1. pp. 28-32.

[34] K. Tayor and J. Trevelyan. 1995. A Telerobot on the World Wide Web. Proceedings of the National Conference of the Australian Robot Association.

[35] C. J. C. H.Watkins and P. Dayan. 1992. Q-learning. Machine Learning. Vol. 8. pp. 279-292.

Virtual Reality Telerobotic System

URI KARTOUN, HELMAN STERN, YAEL EDAN
Department of Industrial Engineering and Management, Ben-Gurion University of the Negev, Be'er-Sheeva, ISRAEL

ABSTRACT

This paper describes a telerobotic system operated through a virtual reality (VR) interface. A least squares method is used to find the transformation mapping, from the virtual to real environments. Results revealed an average transformation error of 3mm. The system was tested for the task of planning minimum time shaking trajectories to discharge the contents of a suspicious package onto a workstation platform. Performance times to carry out the task directly through the VR interface showed rapid learning, reaching standard time (288 seconds) within 7 to 8 trials - exhibiting a learning rate of 0.79.

1. INTRODUCTION

Teleoperation is used when a task has to be performed in a hostile, unsafe, inaccessible or remote environment (1). A telerobot is defined as a robot controlled at a distance by a human operator (HO) regardless of the degree of robot autonomy. Telerobotic devices are typically developed for situations or environments that are too dangerous, uncomfortable, limiting, repetitive, or costly for humans to perform (2). Some applications include: underwater (3), space (4), resource industry (5) and medical (6)(7).

Examples for using graphical models to allow users to control robots off-line and practice control techniques can be found in the RobotToy research (8), the KhepOnTheWeb project (9) and the WITS (Web Interface for Telescience) project (10). NASA developed WITS for controlling remote vehicles on planets such as Mars and Saturn. In the Tele-Garden project (11), users can tend a garden that contains live plants through a graphical representation of the environment. The University of Western Australia's Telerobot experiment (12) provides Internet control of an industrial ASEA IRB-6 robot arm. The PumaPaint project (13) is a website allowing users to control a PUMA-760 robot equipped with a parallel-fingered gripper, to perform painting tasks on an easel.

The virtual reality telerobotic system, described in this paper allows a HO to: (a) perform off-line path planning by manipulating an object in a VR robotic scene, and (b) perform on-line control by indirectly controlling the real robot through manipulation of its VR representation in real-time. The available commands for control are the manipulator coordinates (x, y, z) and open / close gripper. When a command is chosen, the VR model is initially updated. After conversion, the joint angles are sent to the real robot for execution.

To demonstrate the utility of the system, we focus on the task of bag shaking. 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. Extraction of the bags contents using telerobotics, which avoids these problems, is the subject addressed here. In the on-line control mode suggested here, the user controls the robot to perform the task or develops a plan off-line before downloading it to the robot's controller for execution. For off-line bag shaking, the HO chooses a sequence of spatial locations (including inter point speeds) in the VR model to define a shaking trajectory. The trajectory is then downloaded to the real robot for execution.

In this paper we report on experiments for on-line control. The system architecture is presented in Section 2. Before carrying out the on-line control experiments it is necessary to calibrate the VR-telerobotic system. Calibration experiments are described in section 3. In section 4 we report on user experiments, using on-line control for the bag lifting and shaking task. The paper ends with conclusions and directions for future work in Section 5.

2. SYSTEM ARCHITECTURE

The proposed virtual reality (VR) telerobotic system contains a human operator (HO), VR web-based control interface, Internet access method, a remote server, a robot and its controller, and visual sensory feedback (Fig. 1).

2.1 User and control interface

The HO requires interacting with the remote robot through a control interface (Fig 2). The interface developed includes a five degree of freedom articulated robotic arm model of the "CRS A255" robot (14), two views of the real robot, a checkerboard on a table, and a world coordinate diagram that shows the x, y and z directions in the 3D scene. In addition, overall and close views of the robot site are displayed on-line.

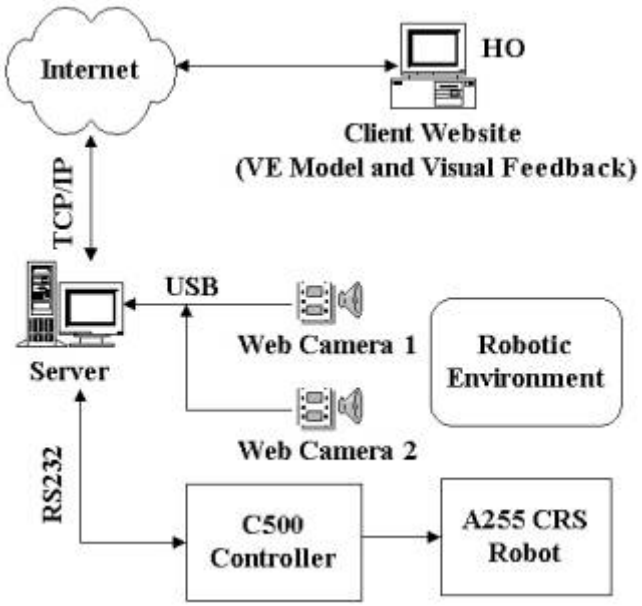


Figure 1. System architecture layout

The system has six different operational stages controlled through predefined control panels. These are: changing the real robots speed, showing a 3D-grid (Fig. 3) that contains spatial locations which the robot gripper moves to when selected, selecting the viewing aspect of the VR model, planning shaking policies, planning off-line paths for downloading to the real robot, and on-line simultaneous control (in real-time) of the VR and real robots.

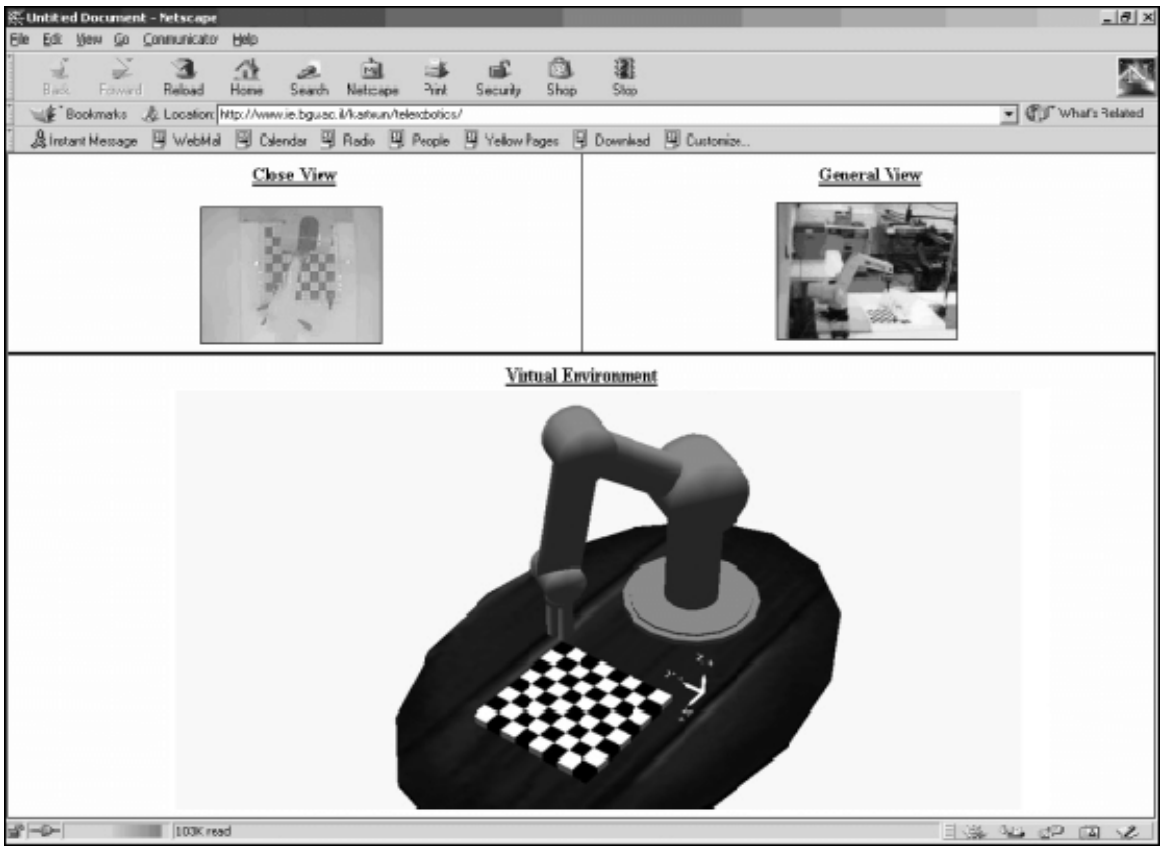


Figure 2. Web-based interface (camera views, and virtual environment)

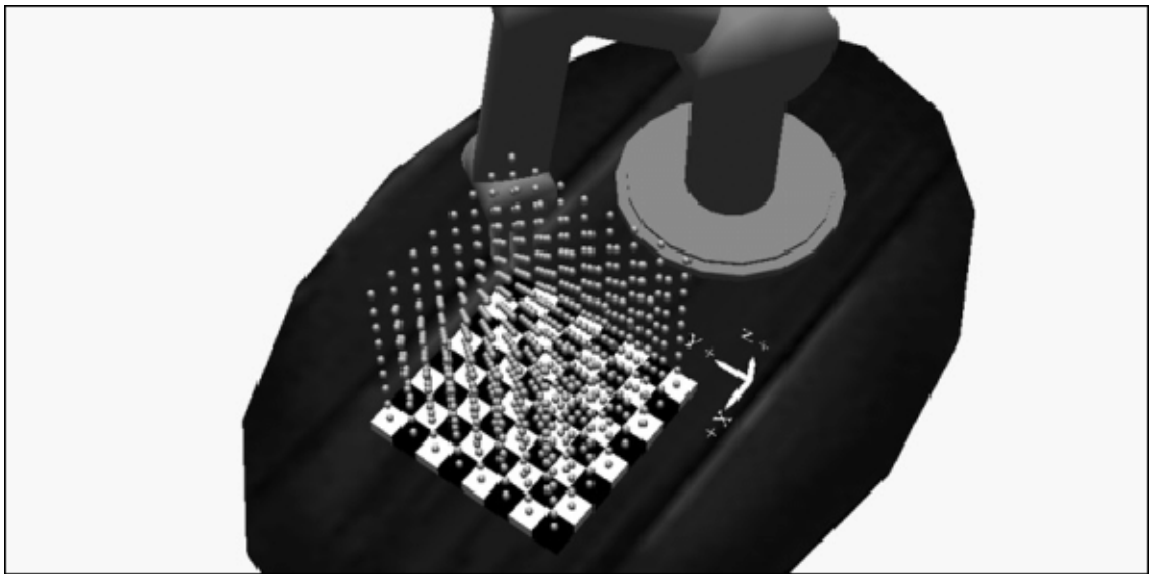


Figure 3. 3D-grid

2.2 Virtual environment

The virtual environment (VE) model was built and developed using "3D-Studio-Max" (15) and "Alice" (16) softwares. "Alice", a rapid prototyping software for creating interactive computer graphics applications (17)(18)(19)(20)(21), was chosen to be the VR software. It is designed to enable rapid development and prototyping of interactive graphics applications and uses "Python" (22)(23) as the language for writing its scripts.

2.3 Communication

The HO communicates with the server, connected to a robotic arm (Fig. 4) through a web-browser. Commands sent from the VE client are transmitted through TCP/IP to the server that extracts them and updates the real robot.

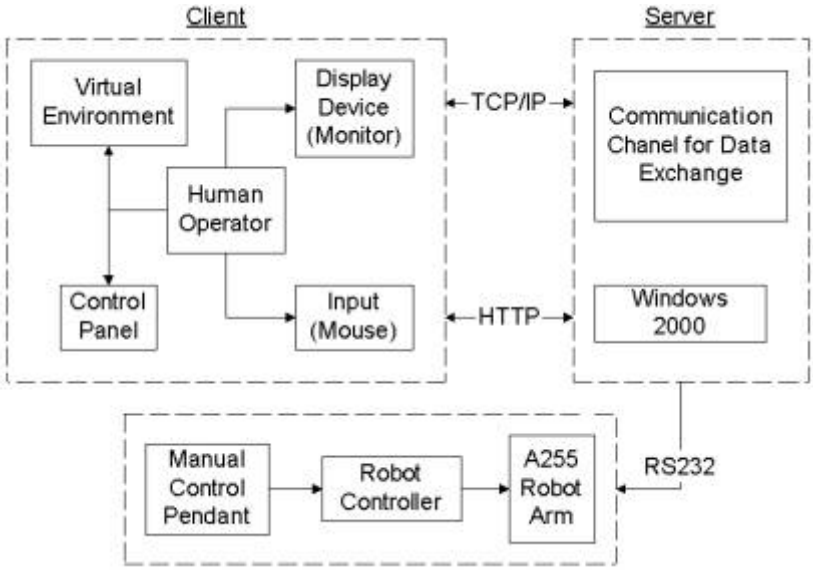


Figure 4. Client-server communication architecture

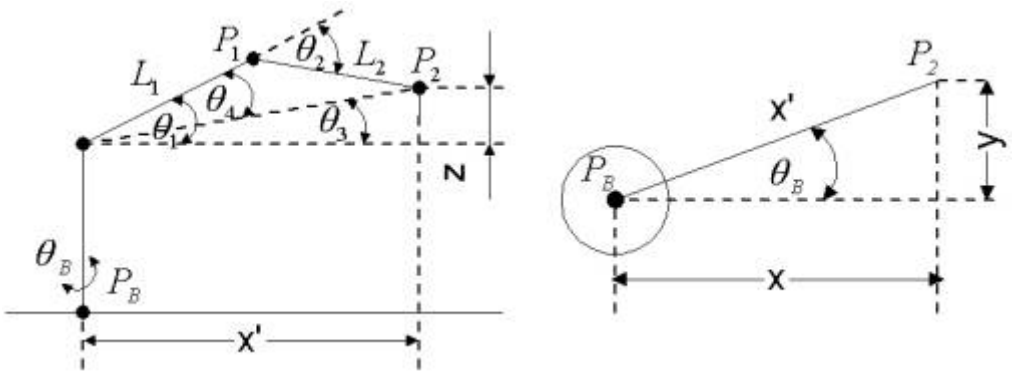
2.4 Sensory feedback, robot, and controller

The remote server is connected to the robot controller, and two USB web-cameras (Fig. 1), which constantly send updated images to the client for observation of the environment. Images are in 24 bit colour and of size 240X180 pixels which appear as close-up and overall views (Fig. 2) in the client browser interface. The "A255" robot system consists of robot arm and controller. The robot arm is equipped with a special gripper capable of sliding under bags.

3. SYSTEM CALIBRATION

3.1 Kinematics

The inverse kinematics (IK) equations were solved using a closed form analytical solution (24)(25)(26). It has the benefit of being an exact solution and very fast to calculate. For the VR robot, an IK algorithm was implemented to determine the joint angles required to reach an end point location by supplying the (x, y, z) coordinates. The side and top views of the VR robotic chain are shown in Fig. 5. The distance x' is obtained from the projection of the shoulder and the elbow links onto the X-Y plane. The x and the y values are the horizontal and vertical values of the robot gripper position relative to the robot base-point, P_B. The z coordinate is taken vertical to the platform plane and measured from the base of L_1.



(a) Side view **(b) Top view**

Figure 5. "A255" robotic chain

Using trigonometric identities yields:

$$\theta_1 = \arctan\left(\frac{z(L_1 + L_2\cos(\theta_2)) + yL_2\sin(\theta_2)}{zL_2\sin(\theta_2) - y(L_1 + L_2\cos(\theta_2))}\right), \quad [1]$$

$$\theta_2 = \arccos\left(\frac{\mathrm{y}^2 + z^2 - L^2{}_1 - L^2{}_2}{2L_1L_2}\right), \quad [2]$$

$$\theta_3 = \arctan(z/x'), \quad [3]$$

$$\theta_4 = \arctan\left(\frac{L_2\sin(\theta_2)}{L_1 + L_2\cos(\theta_2)}\right), \quad [4]$$

$$\theta_B = \arctan(y / x), \quad [5]$$

$$x' = \sqrt{x^2 + y^2}. \quad [6]$$

Now, given any end point position (x, y, z), the joint angles required to reach it are determined by θ_B, θ_1, and θ_2 using [1] through [6].

3.2 Transformation matrix

A transformation matrix, providing a 1 to 1 mapping between the VR and real robots is estimated from corresponding pairs of 32 intersection points on the checkerboard appearing in the VR and the real environments. The estimate is based on the method of aligning a pair of shapes, which uses a least-squares approximation (27). Given two similar shapes, x_1 (the VR environment) and x_2 (the real environment), a rotation θ, a scale s, and a translation (t_x, t_y) is calculated. The transformation equation using the 3x3 transformation matrix is shown in equation [7]. The points ($x_{vr(i,j)}$, $y_{vr(i,j)}$) and ($x_{r(i,j)}$, $y_{r(i,j)}$) are in the VR scene and in the real environments, respectively.

$$\begin{pmatrix} s\cos\theta & -s\sin\theta & t_x \\ s\sin\theta & s\cos\theta & t_y \\ 0 & 0 & 1 \end{pmatrix} \begin{pmatrix} x_{vr(i,j)} \\ y_{vr(i,j)} \\ 1 \end{pmatrix} = \begin{pmatrix} x_{r(i,j)} \\ y_{r(i,j)} \\ 1 \end{pmatrix} \quad [7]$$

3.3 Transformation accuracy

Given the calibrated transformation matrix an experiment to determine the transformation error was performed. The experiment (Fig. 6) starts with controlling the VR robot arm through the VR interface. The coordinates of 32 points (x_{vr} , y_{vr}), selected from checkerboard intersections, were set as test points. These points were inserted into [7] to obtain (x_r , y_r) which were sent to the real robot.

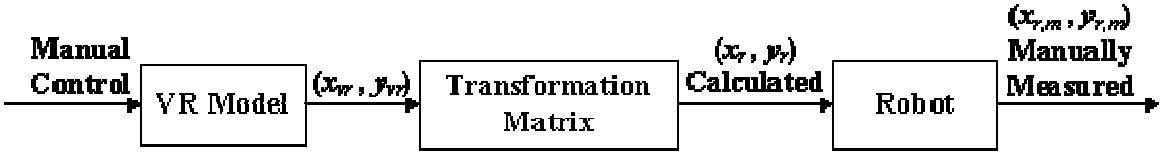


Figure 6. Calibration experiment flow chart

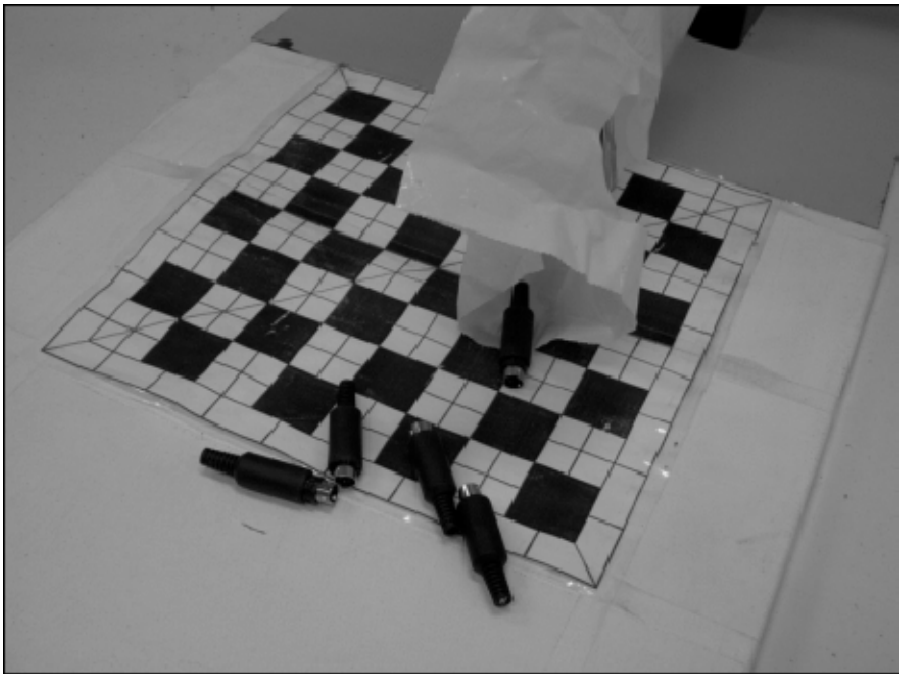
A pen inserted into the real robot's gripper marked its controlled position. The coordinates of the robot's pen on the checkerboard intersection points ($x_{r,m}$, $y_{r,m}$) in the real environment were measured manually by a ruler. The average transformation error between the robot's pen positions, and the desired points in the real checkerboard was found to be 3mm.

4. SYSTEM TEST USING VR ON-LINE CONTROL

The system was tested using on-line control through the VR interface for the task of shaking out the contents of a plastic bag. The HO views the images of the robotic workspace in the client browser, and commands the robot by selecting points in the 3D VR scene. Using the

calibrated mapping of points in the VR scene to the real workspace, the robot is controlled to carry out the task. A view of the experiment to empty the contents of a plastic bag onto the platform is shown in Fig. 7. It is assumed that the bag contains ten identical electronic components known in advance. Five inexperienced operators performed ten identical experiments, and performance times (the amount of time it takes to extract all the objects from the bag) were recorded. The learning curve of task completion time was reduced quickly reaching standard time (288 seconds) after 7 to 8 trials (Fig. 8).

(a) Overall view



(b) Close view

Figure 7. "A255" robot, plastic bag, platform and electronic components

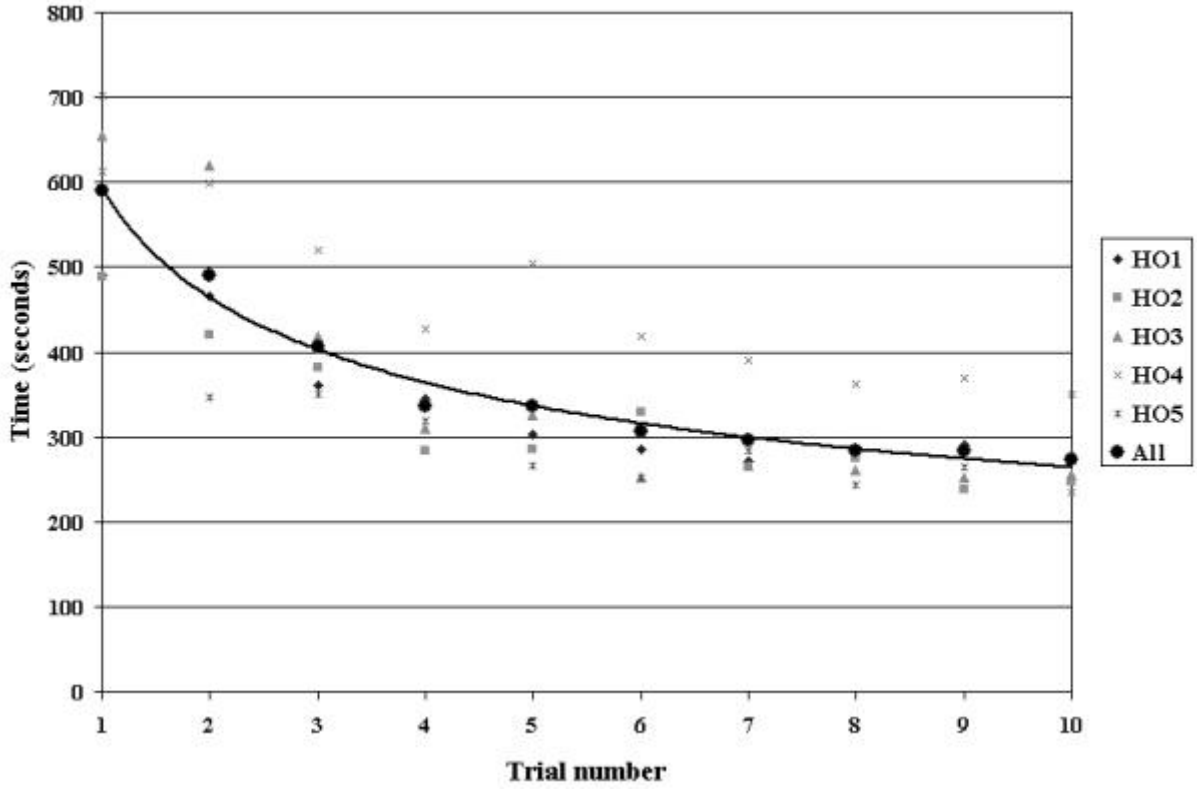


Figure 8. Learning curve experimental results

5. CONCLUSIONS AND FUTURE WORK

This paper describes the design, implementation and testing of a real-time VR-telerobotic web based system. Visual feedback arrived to the human interface via two independent web-cameras and a transformation matrix mapped the VE to the real one was calculated to calibrate the system. The system allows a human operator to: (a) perform off-line path planning by manipulating an object in a VR robotic scene, (b) perform on-line control by indirectly controlling the real robot through manipulation of its VR representation in real-time. A least squares method is used to find the orientation, rotation and translation of 1 to 1 mapping between the virtual and real environments. Results revealed an average transformation error of 3mm. The system was tested for the task of planning minimum time shaking trajectories to discharge the contents of a suspicious package onto the workstation platform. Performance times to carry out the task directly through the VR interface showed rapid learning, reaching standard time (288 seconds) within 7 to 8 trials. This fast learning rate was calculated as 0.79. A future extension is to use visual feedback from the cameras for creating a second transformation mapping the real environment to the virtual one. Errors for both transformations will be calculated and the real robot will perform corrections in order to reduce it. Possible extension of the VR graphical display of the robotic environment presented here is to use visual input (*e.g.*, data gloves, movement trackers, hand gestures) and output (*e.g.*, head mounted displays, shutter glasses) devices. Future research will include development of a cooperative human-robot learning system for remote robotic operations using a VR interface (28).

ACKNOWLEDGMENTS

This project was partially supported by the Ministry of Defense MAFAT Grant No. 1102, and the Paul Ivanier Center for Robotics Research and Production Management, Ben-Gurion University of the Negev.

REFERENCES

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

(2) **N. Durlach** and **S. N. Mavor**, Virtual Reality: Scientific and Technological Challenges, *National Academy Press*, Washington D.C., 1995.

(3) **L. Hsu, R. Costa, F. Lizarralde,** and **J. Soares**, "Passive arm based dynamic positioning system for remotely operated underwater vehicles", *IEEE International Conference on Robotics and Automation*, vol. 1, pp. 407-412, 1999.

(4) **G. Hirzinger, B. Brunner, J. Dietrich,** and **J. Heindl**, "Sensor-based space robotics-ROTEX and its telerobotic features", *IEEE Transactions on Robotics and Automation*, vol. 9, no. 5, pp. 649-663, 1993.

(5) **A. A. Goldenberg, J. Wiercienski, P. Kuzan, C. Szymczyk, R.G. Fenton,** and **B. Shaver**, "A remote manipulator for forestry operation", *IEEE Transactions on Robotics and Automation*, vol. 11, no. 2, pp. 185-197, 1995.

(6) **D. Kwon, K. Y. Woo,** and **H. S. Cho**, "Haptic control of the master hand controller for a microsurgical telerobot system", *IEEE International Conference on Robotics and Automation*, pp. 1722-1727, 1999.

(7) **D. Sorid** and **S. K. Moore**, "The virtual surgeon", *IEEE Spectrum*, pp. 26-39, 2000.

(8) http://robotoy.elec.uow.edu.au/

(9) **O. Michel, P. Saucy,** and **F. Mondada**, "KhepOnTheWeb: an experimental demonstrator in telerobotics and virtual reality", *Proceedings of the International Conference on Virtual Systems and Multimedia, IEEE VSMM'97*, pp. 90-98, 1997.

(10) **P. G. Backes, K. S. Tso,** and **G. K. Tharp**, "The web interface for telescience", Presence, *MIT Press*, vol. 8, pp. 531-539, 1999.

(11) **K. Goldberg, J. Santarromana, G. Bekey, S. Gentner, R. Morris, J. Wiegley,** and **E. Berger**, "The Telegarden", *Proceedings of ACM SIGGRAPH*, 1995.

(12) **K. Talyor** and **J. Trevelyan**, "A telerobot on the world wide web", *National Conference of the Australian Robot Association*, 1995.

(13) **M. Stein**, "Interactive internet artistry, painting on the world wide web with the PumaPaint project", *IEEE Robotics and Automation Magazine*, vol. 7, no. 1, pp. 28-32, 2000.

(14) "A255" Robot System, *CRS Robotics Human Scale Solutions*, 1998.

(15) http://www.discreet.com/

(16) http://www.alice.org/

(17) **M. J. Conway, R. Pausch, R. Gossweiler,** and **T. Burnette**, "Alice: a rapid prototyping system for building virtual environments", *Adjunct Proceedings of ACM CHI'94 Human Factors in Computing Systems Conference*, vol. 2, pp. 295-296, 1994.

(18) **R. Pausch, T. Burnette, A. C. Capehart, M. Conway, D. Cosgrove, R. DeLine, J. Durbin, R. Gossweiler, S. Koga,** and **J. White**, "A brief architectural overview of Alice, a rapid prototyping system for virtual reality", *IEEE Computer Graphics and Applications*, 1995.

(19) **M. J. Conway**, "Alice: easy-to-learn 3D scripting for Novices", *Ph.D. Thesis, Faculty of the School of Engineering and Applied Science at the University of Virginia*, 1997.
(20) **S. Cooper, W. Dann,** and **R. Pausch**, "Alice: a 3-D tool for introductory programming concepts", *Proceedings of the Fifth Anual CCSC Northeastern Conference*, pp. 107-116, 2000.
(21) **W. Dann, S. Cooper,** and **R. Pausch**, "Making the connection: programming with animated small world", *Fifth Annual SIGCSE/SIGCUE Conference on Innovation and Technology in Computer Science Education (ITiCSE)*, pp. 41-44, 2000.
(22) http://www.python.org/
(23) **M. Lutz**, Programming Python, *O'Reilly and Associates*, 2001.
(24) **J. Lander**, "Oh my god, I inverted kine", *Game Developer Magazine*, pp. 9-14, 1998.
(25) **J. J. Craig**, Introduction to Robotics: Mechanics and Control, *Addison-Wesley*, 1989.
(26) **P. J. McKerrow**, Introduction to Robotics, *Addison-Wesley*, 1991.
(27) **T. F. Cootes, C. J. Taylor, D. H. Cooper,** and **J. Graham**, "Training models of shape from sets of examples", *Proceedings of the British Machine Vision Conference*, pp 9-18, 1992.
(28) **U. Kartoun**, "A human-robot collaborative learning system using a virtual reality telerobotic interface", *Ph.D. Thesis Proposal, Department of Industrial Engineering and Management at the Ben-Gurion University of the Negev*, Israel, 2003, Available: http://www.compactech.com/kartoun