Fast Reinforcement Learning for Vision-guided Mobile Robots

Tomás Martínez-Marín
Dept. of Physics, Systems Engineering and Signal Theory
University of Alicante
Alicante, Spain
Email: tomas@dfists.ua.es

Tom Duckett
AASS, Dept. of Technology
Orebro University
SE-70182 Orebro, Sweden
Email: tom.duckett@tech.oru.se

Abstract— **This paper presents a new reinforcement learning algorithm for accelerating acquisition of new skills by real mobile robots, without requiring simulation. It speeds up Q-learning by applying memory-based sweeping and enforcing the "adjoining property", a technique that exploits the natural ordering of sensory state spaces in many robotic applications by only allowing transitions between neighbouring states. The algorithm is tested within an image-based visual servoing framework on a docking task, in which the robot has to position its gripper at a desired configuration relative to an object on a table. In experiments, we compare the performance of the new algorithm with a hand-designed linear controller and a scheme using the linear controller as a bias to further accelerate the learning. By analysis of the controllability and docking time, we show that the biased learner could improve on the performance of the linear controller, while requiring substantially lower training time than unbiased learning (less than 1 hour on the real robot).**

I. Introduction

In this paper we present a new approach for accelerating learning of new skills by mobile robots using reinforcement, in order to eliminate the need for offline simulations as in many current approaches. A new reinforcement learning algorithm is proposed that greatly speeds up Q-learning by applying memory-based sweeping [1] and reducing the number of allowed state transitions by enforcing the "adjoining property" [2], [3], a technique from the field of optimal control. This technique exploits the fact that in many robotic applications, continuous sensory variables (interval numbers) are quantized into discrete states, where the natural ordering of states is preserved (ordinal numbers). While standard Q-learning assumes unordered states (nominal numbers), the adjoining property assumes that the states are ordered and allows only transitions between neighbouring states.

The new algorithm is tested on a docking task for an Activmedia PeopleBot mobile robot, in which the robot has to move towards an object located on a table and then pick up that object with its gripper (Fig. 1). Due to physical constraints, i.e., the very limited grasping capability of the PeopleBot, the robot has to dock itself at a very precise position and orientation next to the table, without hitting the table, so that the object can be reached by the gripper. A solution based on visual servoing [4] is applied, where the robot's pan-tilt camera is used to keep track of the object and the edge of the table while the robot is steered to the desired configuration. An image-based visual servoing method is used, where the control law is computed directly from visual features, without explicit pose estimation.

In our approach, a separate tracking behaviour is used to control the robot's pan-tilt camera, in order to keep the object in the centre of the image at all times while the robot is moving. A minimal number of state variables are extracted from the image (concerning the apparent slope of the table edge) and the position encoders of the camera (concerning the pan and tilt angles). These state variables are then used in the input to the motor controller of the mobile robot.

In the experiments we compare a number of control algorithms, including a hand-designed linear controller, the new reinforcement learning algorithm, and a scheme using the linear controller as a bias to accelerate reinforcement learning. By analysis of the controllability and docking time, we found that the biased learning system could improve on the performance of the linear controller, while requiring much less training than unbiased learning. With this approach, a high performance controller could be acquired in less than 1 hour on the real robot.

A. Related Work

In principle, a robot could learn any task from scratch by reinforcement learning (RL) given enough time [5]. In practice, however, this time is too high for most complex tasks, and the designer must find some method to incorporate prior knowledge into the learning process.

Smart and Kaelbling addressed the practical issues of getting Q-learning to work on a real mobile robot [6]. The tasks investigated were wall following and obstacle avoidance. Learning was carried out in two phases: first, with the control policy being provided by a pre-programmed controller or a human with a joystick, and second, using the learned policy of the robot while RL continues ("fine tuning"). The tasks were also simplified by controlling only a single variable, the rotational velocity of the robot, while the translational velocity was controlled by a hand-coded algorithm.

Gaskett et al. [7] introduced a RL-based approach for training a mobile robot to wander (obstacle avoidance) and pursue a target using real-time vision. This was implemented in a subsumption architecture, such that target pursuit takes over from wandering when a valid target is detected. They

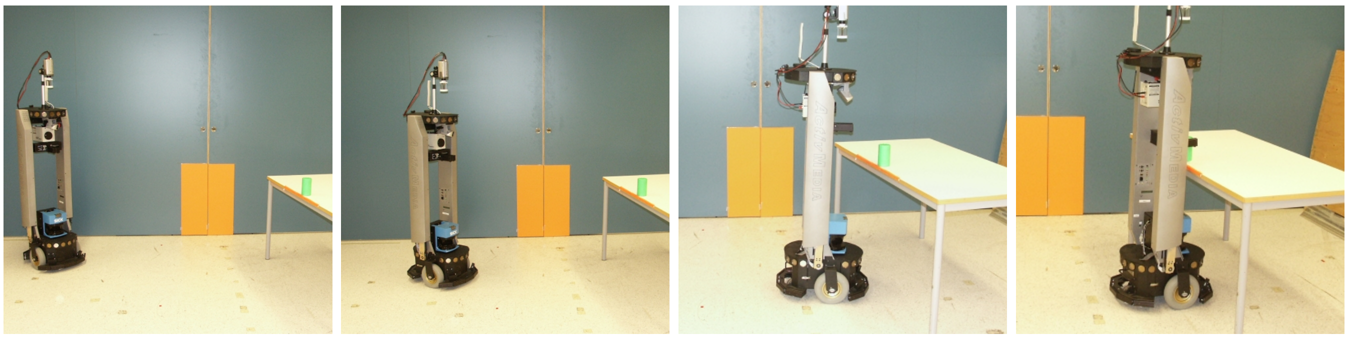

Fig. 1. Sequence of images showing the docking behaviour learned on the real robot.

used Advantage Learning to improve the optimal behaviour of Q-learning and a neural network to map the states to actions. Target pursuit was realised by visual servoing. This is simpler than our robot docking task because it requires servoing to a position but not to a particular orientation.

In work done independently, Weber et al. [8] used a neural network based approach to solve the docking problem on a Peoplebot robot by reinforcement learning. However, the reinforcement learning was done in simulation, limiting the generality of the approach to tasks and environments that can be simulated accurately. In addition, the pan-tilt camera was fixed to see the gripper and the object in the same image (i.e., the pan-tilt mechanism was not exploited). This means that their visual controller is only valid when the robot is near to the table (40–50 cm). Our controller is valid for much longer distances of 3–4 m.

II. Visual Servoing Task

The complete mobile manipulation task can be divided into two phases: docking (approaching the table), and grasping (picking up the object). Due to physical constraints of our robot, the first task is relatively difficult while the second task is relatively easy, provided that docking has been successful. In this section, we therefore concentrate on the solution of the docking problem by visual servoing. We define the 3D state space used for input to the closed loop controller, and the techniques for estimating the state variables on the real robot.

A. The robot

We used an Activmedia PeopleBot, a holonomic mobile robot that is equipped with an array of sensors including sonar, laser, infrared and a pan-tilt camera (see Fig. 1). The mobile robot base has two DOF, corresponding to the translational and rotational velocities. The robot also has a basic gripper with one DOF that can be used for simple grasping operations.

In this work, only the vision sensor was used to estimate the state variables. The pan-tilt unit has two DOF's. The range of the pan angle is ± 100 degrees, and the tilt angle has a range of 0-90 degrees. The position of the camera on the robot is especially suitable for this application, allowing the robot to see both the gripper and the target object in the same image when the robot has completed the docking phase.

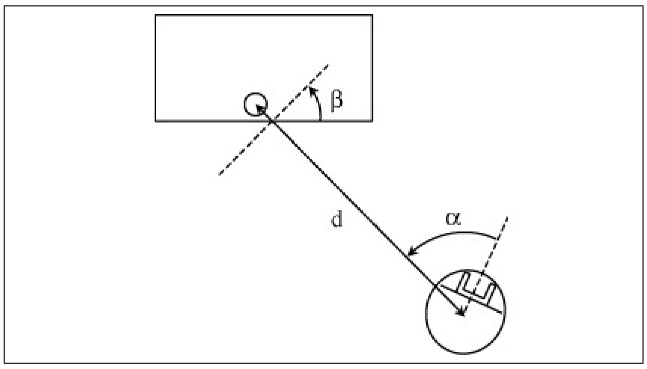

Fig. 2. Representation of the state variables in Cartesian space.

B. The state space variables

In many vision-based manipulation applications, information has to be extracted from the image concerning the position, orientation, size and shape of the object. In our case, it is not necessary to calculate a 3D pose estimate, it is sufficient to use a 2D pose estimate to correct the pan-tilt angles in order to keep the object in the centre of the image. Instead of using the 2D position (relative x, y values in the image) as state variables, we use the pan and tilt camera angles, since these two variables implicitly contain 3D information.

Fig. 2 depicts the three state variables that define a new state space (β, α, d), where β is the angle between the table and the perpendicular direction to segment d, α is the angle between the mobile robot and the perpendicular direction to segment d, and d is the distance between the object and the robot. For the docking task, the goal state in this relative coordinate system is the origin. The state equations of this formulation are:

$$\dot{\beta} = \frac{v_T}{d}\sin\alpha, \tag{1}$$

$$\dot{\alpha} = \frac{v_T}{d}\sin\alpha - v_R, \tag{2}$$

$$\dot{d} = -v_T\cos\alpha. \tag{3}$$

where v_T is the translational velocity and v_R is the rotational velocity of the mobile robot. It is important to note that the state equations are only used in the simulations in order to describe the robot trajectories. In the experiments, the model is built on-line by recording the transitions between states and the immediate rewards.

In this paper, the problem is simplified by fixing the value of v_T. Then, the task is reduced to controlling the orientation of the robot in a two-dimensional state space (β, α) by the action v_R. In our reinforcement learning experiments, we allow only two possible actions in each state (turning left or right at 9 deg/s). In this case the controllability of the system is more limited, since the controller cannot reduce v_T if big changes in the orientation of the robot are required.

C. Estimating the state variables on the real robot

The state variables (β, α, d) can be estimated through the variables $(m, pan, tilt)$ respectively, which can all be obtained from the image sensor (note that only the first two variables are used in current experiments, as discussed above). The variable m corresponds to the slope of the edge of the table in the image (see Fig. 3). The pan and tilt angles are valid if the object is kept in the centre of the image. For reinforcement learning on the real robot, β is estimated using $\arctan(m)$, and α is estimated using pan. The continuous state variables are converted into discrete states (cells) by uniform quantization (see section III).

For the success of the docking behaviour, it is essential to track the object at all times while the robot is moving and picking up the object. The pan-tilt camera has two DOF that allow the independent control of the x-position with the pan motor and the y-position with the tilt motor. For each axis, a Proportional-Derivative (PD) controller is employed in order to keep the object in the centre of the image [9]. The parameters of the controller were adjusted to avoid overshoot if the object or the robot changes its position suddenly.

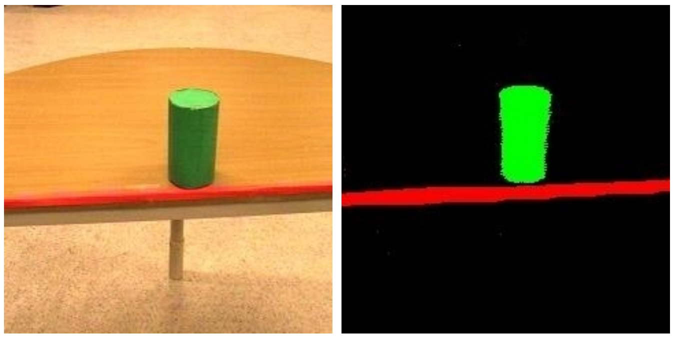
Fig. 3. Left: raw image. Right: image after colour thresholding.

The image processing steps can be described as follows. The regions of the image corresponding to the object and the table edge are extracted by simple colour segmentation in HSV space. In our experiments, we simplified the experimental set-up by using a green-coloured drink can and an orange stripe placed on the edge of the table. However, we are also investigating more advanced image processing in current work, e.g., using a Hough transform to recognise the table edge without needing the orange stripe. The centre of the object (x, y) is determined by taking the median x and y values in the segmented image. The gradient of the edge of the table, m, is calculated by a standard least-squares fitting method.

III. Reinforcement Learning

Reinforcement learning methods only require a scalar reward (or punishment) to learn to map states to actions. In our approach, the learner is given a positive reward if it successfully reaches the goal configuration, or a negative reward if it fails. A small negative reward is also given for every time step, in order to minimize the time taken to reach the goal. The aim is to learn a policy, i.e., a mapping from perceived states of the environment to actions to be taken in those states, that maximizes the accumulated reward in each state [1].

Here we present a novel reinforcement learning algorithm for mobile robots. Its fundamental characteristics are: the reinforcement learning mechanism is implemented through the update equation of *Q-learning*; the *adjoining property* is used to approximate the optimal robot behaviour for continuous state spaces; and the data are processed on-line to build a model that is used to further accelerate the learning by "mental rehearsal" or planning, thus implementing an *indirect reinforcement learning* algorithm.

A. Q-learning

Q-learning is one of the most popular reinforcement learning methods, since it can solve model-free optimization problems and has a simple formulation. The knowledge is saved in a look-up table that contains an estimation of the accumulated reward for reaching the goal in each situation or state. The accumulated reward for each state-action pair $Q(s, a)$ is updated by the one-step equation

$$\Delta Q(s,a) = \alpha \left(r + \gamma max_{a'} Q(s', a') - Q(s,a) \right), \tag{4}$$

where Q is the expected value of performing action a in state s, r is the reward, α is a learning rate which controls convergence and γ is the discount factor. The discount factor makes rewards earned earlier more valuable than those received later. The action a with highest Q value at state s is the best policy up to instant t.

B. The adjoining property

Q-learning was conceived for discrete state and action spaces, where the state space is not necessarily metric. In robotic applications the state space is continuous, so it is mandatory to discretize the state space into cells. The inherent discretization errors can produce a poor approximation to the optimal behaviour in complex nonlinear systems such as mobile robots.

To address this problem we have introduced the adjoining property taken from the CACM technique [2], [3], which is an optimal control technique for nonlinear continuous dynamic systems. This method is based on the Adjoining Cell Mapping (ACM) technique, whose central concept is the creation of a cell mapping where only transitions between adjoining

State variables: 2. (961 states)	x_1: -65 $\leq \beta \leq$ 65°. Cells: 15 (31) x_2: -80 $\leq \alpha \leq$ 80°. Cells: 15 (31)
Objective state:	$(\alpha, \beta) = (0°, 0°)$
Control variables: 1. (2 actions)	u_1: -9 $\leq v_R \leq$ 9°/s. (u_1: -10 $\leq v_R \leq$ 10°/s.)
Sampling time:	T_s: 0.125 sec.
Reward:	r = 300 if goal r = -50 if sink r = -n (T_s) otherwise
Adjoining distance:	D-1

TABLE I

PARAMETERS IN THE RL ALGORITHM ON THE REAL ROBOT. (BRACKETS INDICATE DIFFERENT VALUES USED IN SIMULATION).

cells are allowed [10]. The adjoining property states that the distance D_k between the current cell and the previous cell is equal to some integer value k equal or greater than 1. The distance between two cells $\mathbf{z}$ and $\mathbf{z}'$ is defined as:

$$D_k(\mathbf{z}, \mathbf{z}') = \max_j |z_j - z'_j| = k. \tag{5}$$

In Q-learning the transitions between states are evaluated at fixed sample times, while with our RL controller the transitions have to satisfy the adjoining distance condition in order to be evaluated. By appropriate selection of this distance with respect to the number of cells, it is possible to minimize quantization effects and better approximate the optimal behaviour of the system.

C. Indirect reinforcement learning

The RL algorithm has been implemented as a model-based reinforcement learning method, where all states are backed-up N times by simulation (planning) after each transition that satisfies the adjoining property. Another alternative would be to apply Prioritized Sweeping (PS) [11], where a prioritized queue of state-action pairs is maintained in order to focus the search on transitions with big changes in Q. In our application, where the sample time is 0.125 seconds and the number of states is less than a thousand, prioritization of the state-action pairs would not improve the efficiency of the algorithm.

In direct reinforcement learning (e.g., Q-learning), the back-ups are only made by experimentation, which is suitable when the back-up time for experimentation compared to simulation is not very high. In general, model-based reinforcement learning finds better trajectories and manages changes in the environment (e.g., obstacles) more efficiently than direct reinforcement learning.

The model stored in memory for each state-action pair (s, a) is the estimation of the continuous state x' reached by applying an action a and the average of the reward r. The model is only updated when the adjoining property is satisfied.

```
Initialize $Q(s,a)$ and $Model(s,a)$
    $x \leftarrow$ current state
    $s \leftarrow cell(x)$
    IF    s = sink or goal
    THEN  $reverse(x)$
    ELSE  $a \leftarrow policy(s)$
          Execute action $a$
          Observe resultant state $x'$ and reward $r$
          IF    $D_k$-adjoining$(x, x')$
          THEN  $Model(s,a) \leftarrow x', r$
                FOR all $(s,a)$, repeat N times:
                    $\bar{x}', \bar{r} \leftarrow Model(s,a)$
                    $s' \leftarrow cell(\bar{x}')$
                    Update $Q$ table using Eqn. 4
UNTIL training terminated
```

Fig. 4. Reinforcement Learning Algorithm, a variant of the Dyna-Q algorithm [1].

D. Implementation details

The RL algorithm that implements the concepts described above is presented in Fig. 4. The state is represented in the algorithm by a real valued vector x, which is converted to the discrete state s (integer index) by the function $cell()$. In our experiments, uniform discretization was used with 31 cells per variable in simulation and 15 cells on the real robot (see Table I for full details of the RL parameters). The function $Dk - adjoining()$ is used to determine whether the adjoining property has been satisfied. The index s is used to update the Q-table, and x is used to update the function $model()$. Since the controller uses noisy data from an image sensor, the function $model()$ estimates the state of the system by filtering before storing it. In our experiments an average filter was used.

For the docking behaviour the aim of the controller is to move the robot from any initial position inside the region of interest to the object position through a minimum-time trajectory. A trial finishes when the robot moves outside of the state space (sink cell) or when it enters in the goal. Then, the function $reverse()$ moves the robot backward, using its vision system to keep the object in the centre of the image, until some starting position inside the state space is reached. We use a range of the $tilt$ angle in order to limit the area where the robot is trained ($20 < tilt < 43°$, corresponding to a distance $0.20 < d < 0.95$ m), and to avoid collisions between the robot and table. The function $reverse()$ finishes when $tilt < 20°$.

The function $policy()$ selects an action for each transition of the system. The RL controller selects the actions randomly to explore most of the state space during training. Other alternatives such as an ϵ-greedy or softmax exploration policy do not introduce significant benefits in this application, since they reduce the exploration of peripheral states, thus delaying the growth of the controllability region.

By changing the function $policy()$ it is possible to implement other controllers, e.g., the RL-LC controller described in the next section. In the update rule (Eqn. 4), the learning rate α is variable, falling inversely with the number of transitions and the discount factor is fixed to $\gamma = 1$.

The docking task is symmetric in the state and action spaces, i.e., the actions taken when the robot is to the right of the object are symmetric with respect to the actions on the left side. Each cell has a *mirror* cell that satisfies this property. For each transition, we exploit this symmetry by also updating the model and the Q-factor for the mirror cell.

IV. Other Controllers Investigated

A. Linear controller

The equation of the state space linear controller is given as

$$v_R = \dot{\alpha} = K_1\beta + K_2\alpha. \quad (6)$$

The advantage of this simpler linear controller with respect to optimal controllers is that it does not require a model of the system and environment, and only requires tuning of the (K_1, K_2) parameters to obtain satisfactory performance.

On the real robot, the linear controller was implemented as $V_R = K_m m + K_p pan$, where m and pan are the state variables measured on the PeopleBot. The parameter values used in our experiments were $K_m = 100$ and $Kp = 0.2$.

B. RL-Linear controller

A number of schemes have been proposed to speed up the learning phase of RL methods [12], [6]. Although the RL algorithm presented in this paper learns the optimal policy much faster than other RL methods (e.g., the basic Q-learning algorithm requires a training phase over 10 times longer), it is beneficial to accelerate the learning even further.

The linear controller can act as teacher of the RL controller in the early stages of learning, causing a fast propagation of the Q-factors in the state space. In order to incorporate the information from the linear controller into the RL algorithm, only the function $policy()$ in Fig. 4 was modified. In this case, the function alternately selects between the actions provided by the linear controller and the random exploration of the RL controller for each trial of the learning process.

C. RL' controller

In order to test the improvement of using the adjoining property in the RL controller, a version of the RL controller without the adjoining restriction was also tested.

V. Simulation Results

The controllers described in the previous sections were compared using two measures of performance: the *percentage of controllable cells* and the *average docking time*. A cell is controllable if starting at the centre point of such a cell (initial state), the system evolves reaching some point inside the objective cell (final state). The average docking time indicates the approximation to the optimal behaviour of the system. Both statistics are calculated from a subset of the possible starting states in a rectangular area between 1 and 2 meters from the object, testing the trajectories that reach the goal.

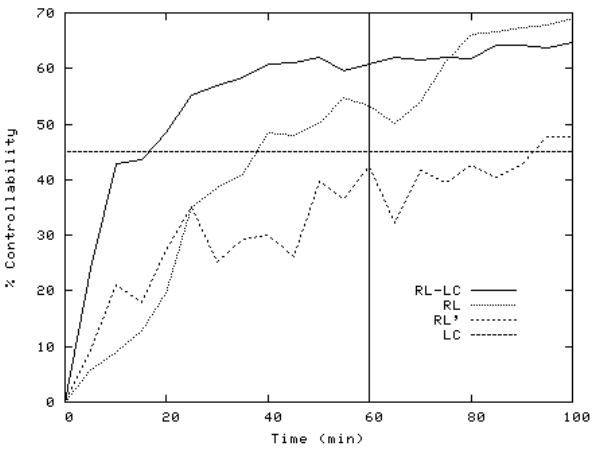

Fig. 5. Percentage of controllable cells over time for the 4 algorithms.

Fig. 5 shows the learning process for RL, RL' and RL-LC controllers using the percentage of controllable cells over time, averaged over 20 runs of the algorithm. The x-axis represents the estimated time that the real robot employs to acquire the current behaviour. In order to test the improvement of using the adjoining property in the RL controller, a version of the RL controller without the adjoining restriction is depicted (RL'). It is observed that the adjoining property provides a greater controllabitity region in less time. Thus, the learning process is faster and more reliable. The results also show that RL improves the performance of the linear controller in both the size of the controllable region and the average time to reach the goal (Table II). Furthermore, the biased learning using the RL-LC controller speeds up the learning phase, obtaining an acceptable behaviour in much less time than the RL controller.

Fig. 6 depicts some trajectories of the robot employing the linear controller (grey) and the biased reinforcement learning controller (black) after training. The translation velocity was fixed at a constant value of $v_T = 0.1$ m/s. As shown in Fig. 6, the trajectories for the RL-LC controller are straighter than for the linear controller, corresponding to a better approximation to the time-optimal behaviour.

VI. Robotic Experiments

The reinforcement learning controllers have been implemented successfully on the real robot (see Fig. 1). When the table is reached at the goal position and orientation, the robot

Controller	Av. Dock. Time(s)	Control. Cells(%)
LC	14.00	44.4
RL	11.84	53.2
RL-LC	11.78	60.8
RL'	11.85	42.3

TABLE II
Average docking time and percentage of controllable cells after 60 minutes.

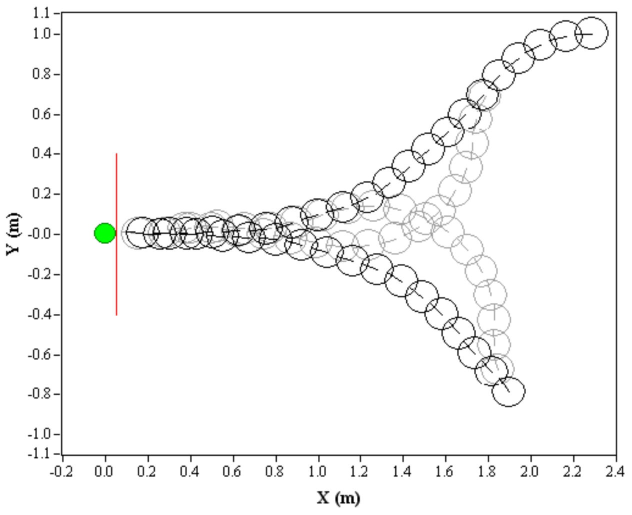

Fig. 6. Some example trajectories using the linear controller (grey) and biased reinforcement learner (black).

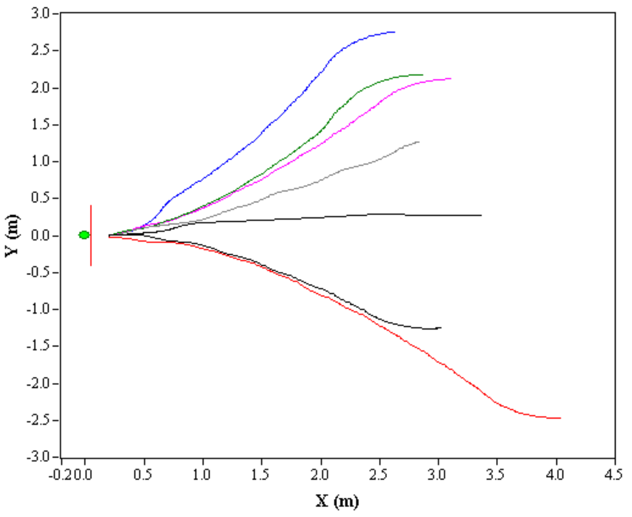

Fig. 7. Some example trajectories using the biased reinforcement learner on the real robot.

picks up the object using a separate hand-coded controller that moves the gripper downwards until its horizontal break beams detect the object. The training time using the RL-LC controller was 40 minutes. Fig. 7 shows some trajectories on the real robot using the RL-LC controller, where both the position and the orientation are smoothly controlled. Furthermore, the robot can reach the object from a long distance (3–4 meters, only limited by the image resolution) since the controller does not have any distance restriction. The RL-LC controller approximates the time-optimal behaviour, improving the performance of the linear controller in time without overshoot in the trajectory.

We also conducted some experiments to test the robustness of the whole system with respect to changes in the environment. If the object is moved during docking, then the robot follows the object (provided that the object is not moved too quickly and stays within the robot's field of view). If the table is turned while the robot is close to the object, then the robot turns to avoid hitting the table.

VII. Conclusion

We presented a solution for mobile robot docking using reinforcement learning in a visual servoing framework. A new RL algorithm was presented that is better suited to real robots that operate in continuous state spaces (by exploiting the adjoining property and model-based sweeping). We also showed that an easy-to-implement linear controller could be used to accelerate the learning even further.

The approach requires no calibration or geometric models, and the reactive behaviour is robust to perturbations and noise. The closed loop solution is based on a relative coordinate system: no global reference frame is required, so the system is robust to positioning errors, e.g., due to odometry drift.

In future research, we intend to extend the approach to more complex tasks and robots with higher dimensional state and action spaces.

Acknowledgment

The authors would like to thank Per Larsson, Grzegorz Cielniak and Henrik Andreasson for much useful help with the robotic implementation on the PeopleBot.

References

[1] R. Sutton and A. Barto, *Reinforcement Learning: An Introduction.* MIT Press, 1998.

[2] P. Zufiria and T. Martínez-Marín, "Improved optimal control methods based upon the adjoining cell mapping technique," *Journal of Optimization Theory and Applications*, vol. 118, no. 3, pp. 657–680, 2003.

[3] T. Martínez-Marín, "Optimal path planning for car-like vehicles in the presence of obstacles," in *Proc. IEEE Int. Conf. on Intelligent Transportation Systems*, Shanghai, 2003, pp. 1161–1164.

[4] S. Hutchinson, G. Hager, and P. Corke, "A tutorial on visual servo control," *IEEE Trans. Robotics and Automation*, vol. 12, no. 5, pp. 651–670, 1996.

[5] R. Sutton, "Reinforcement learning architectures for animats," in *From Animals to Animats, Proc. First Int. Conf. on Simulation of Adaptive Behaviour*, J. Meyer and S. Wilson, Eds. MIT Press, Cambridge MA, 1991.

[6] W. D. Smart and L. P. Kaelbling, "Effective reinforcement learning for mobile robots," in *Proc. IEEE Int. Conf. on Robotics and Automation (ICRA)*, vol. 4, 2002, pp. 3404–3410.

[7] C. Gaskett, L. Fletcher, and A. Zelinsky, "Reinforcement learning for visual servoing of a mobile robot," in *Proc. Australian Conf. on Robotics and Automation (ACRA2000)*, 2000.

[8] C. Weber, S. Wermter, and A. Zochios, "Robot docking with neural vision and reinforcement," in *Proc. IROS-2003 Workshop on Robot Programming by Demonstration*, Las Vegas, 2003.

[9] M. Spong and M. Vidyasagar, *Robot dynamics and control.* John Wiley and Sons, New York, 1991.

[10] P. Zufiria and R. Guttalu, "The adjoining cell mapping and its recursive unraveling, part i: Description of adaptive and recursive algorithms," *Nonlinear Dynamics*, vol. 4, pp. 204–226, 1993.

[11] A. Moore and C. Atkeson, "Priortized sweeping: Reinforcement learning with less data and less time," *Machine Learning*, vol. 13, pp. 103–130, 1993.

[12] S. Suzuki, Y. Takahashi, E. Uchibe, M. Nakamura, C. Mishima, H. Ishizuka, T. Kato, and M. Asada, "Vision-based robot learning towards RoboCup: Osaka University "Trackies"," in *RoboCup-97: Robot Soccer World Cup I (Lecture Notes in Artificial Intelligence)*, H. Kitano, Ed. Springer, 1998.