Teaching by Demonstration of Robotic Manipulators in Non-Stationary Environments

Alexander Skoglund, Tom Duckett, Boyko Iliev, Achim Lilienthal and Rainer Palm
Center for Applied Autonomous Sensor Systems
Department of Technology, Örebro University, Sweden
Email: {alexander.skoglund, tom.duckett, boyko.iliev, achim.lilienthal, rainer.palm}@tech.oru.se

***Abstract*— In this paper we propose a system consisting of a manipulator equipped with range sensors, that is instructed to follow a trajectory *demonstrated* by a human teacher wearing a motion capturing device. During the demonstration a three dimensional occupancy grid of the environment is built using the range sensor information and the trajectory. The demonstration is followed by an exploration phase, where the robot undergoes self-improvement of the task, during which the occupancy grid is used to avoid collisions. In parallel a reinforcement learning (RL) agent, biased by the demonstration, learns a point-to-point task policy. When changes occur in the workspace, both the occupancy grid and the learned policy will be updated online by the system.**

I. Introduction and Motivation

In industry, manipulators are mainly used by companies that make high volume products or products that require high repeatability in the assembling task. Small and medium sized companies are unlikely to invest in an expensive robot and reprogram it for different products when they make changes, unless the transition from assembling or handling one product to the other requires much less effort than to perform the work manually. A promising approach to minimize the effort when programming manipulators is Teaching by Demonstration (TbD). Full-blown Teaching by Demonstration platforms exist in today's laboratories, see [1], [2]. Some use a symbolic approach of high level task understanding, to more connectionistic approaches for imitation; for an overview see [3].

There are several important issues for making these industrial manipulators "smarter"; one is motion planning. Motion planning and control for manipulators using methods from reinforcement learning has received attention in some earlier work, see [4], [5], [6], [7]. In previous work Martín and Millán [5] used a manipulator with sonars for a reaching task using actor-critic learning. Similarly, Santos [8] used a modified Q-learning algorithm for a reaching task. In distinction to their approaches we will use the TbD paradigm and take non-stationary environments into account.

II. Teaching of a reaching task

The goals with this project are to achieve a robot system with a manipulator that is easy to teach and possesses learning and perceiving capabilities. This means a system that can be taught a task by demonstration, and during this, record state information of free spaces shown by the teacher. By using feedback from range sensors the system is able to learn an internal model of the environment, biased by the demonstration. It should also determine for itself where it can go safely, which means obstacle avoidance is included. The task's trajectory is optimized by using self-improvement by off-policy reinforcement learning, but if the uncertainty is too high for making exploration, i.e., self-improvement, it must have further guidance from the teacher. This will altogether become a platform that is flexible and self-improving for simple handling tasks. Thus, a manipulation scenario becomes possible, where new items can be introduced and removed from the workspace, i.e., making the robot work in a non-stationary environment. By using a motion capture device instead of a teach pendant (the manipulator's control panel) the teaching is intuitive, fast and natural, however, not very accurate. The teacher can very easily show the manipulator a task's trajectory and which parts of the workspace are free.

A. Preliminary work

In our preliminary work, presented in [9], we have implemented a system that can follow a demonstrated trajectory, and learn a simple reaching task. The teacher's hand movement is recorded by a motion capture device and mapped onto a 4 link analytical model of the human arm with 3 degrees-of-freedom (DOF) in the shoulder and the wrist respectively and 1 DOF in the elbow. The redundancy of the human arm model, together with the different numbers of DOF of the human arm model and the robot (the robot has 6 DOF) requires the calculation of their direct and inverse differential kinematics, taking further into account the position and the orientation both of the human hand and the robot's end-effector. This is done by using the pseudo inverted Jacobian matrix, which gives us the joint velocities from the end-effector's velocity. The model's trajectory is used by the control system to move the end-effector from point A to point B using the recorded trajectory AB, see figure 1. For ergonomic reasons the movement is mirrored.

The second part of our system is a reinforcement learning (RL) agent that learns to move from any start configuration to a predefined goal configuration. By using the demonstrated trajectory learning is sped up greatly by using Q-learning with a discretized tabular Q-table. However, computing the full Q-table is not feasible, so in our current implementation we are using a locally weighted regression scheme to approximate the Q-function.

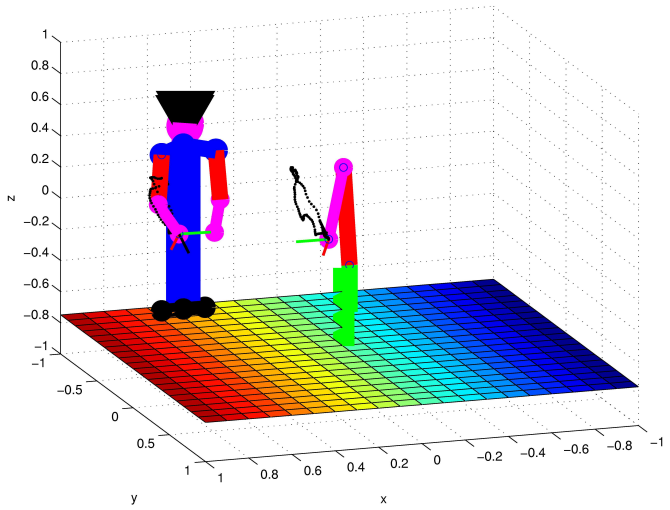

Fig. 1. The captured demonstration is used to drive the human arm model that in turn is mapped to the simulated robot.

III. Proposed Method

From a RL-agent point of view the purpose of the initial demonstration is two-fold; first we obtain the start and goal position of the task, secondly the supervised solutions will guide the agent and avoid the long trial-and-error search that the RL-agent must otherwise face. This way the agent will not perform any trial-and-error search until it has gained some confidence about the current state of the environment. If the self-improvement property is unwanted then there is no need for exploration until the sensors indicate a change in the environment, e.g., an obstacle has been detected.

During the demonstration we will record trajectories and store them in a three dimensional occupancy grid, where the traversed grid cells are marked as free, i.e., no obstacles. When the robot approaches a state not visited before, or the environment has changed then the agent needs to explore. In this case, it must be very careful (i.e., run at slow speed) and will rely on the occupancy grid to avoid collisions by using *virtual collisions*, i.e., when the range detector indicates an object but before a real collision occurs, see [5].

A. The three dimensional occupancy grid

The occupancy grid is a 3D representation of the workspace where information of the perceived grid cell is stored. This grid is used by the policy learning to limit random exploration used by the RL-agent in dangerous cells (i.e, not visited before) to avoid collision. The occupancy grid will be continuously updated using the sonar information. This will become very important when we investigate non-stationary workspaces and the teacher is no longer available. The occupancy grid and the policy learning are updated in parallel. An open issue here is how to use the 3D model for indirect RL.

The occupancy grid can be represented by a number of cells, where each cell is associated with a probability of being occupied by either an object that should be manipulated or an obstacle. Cells visited many times in the demonstration are free with a high probability. The occupancy grid will be made quite approximate for two reasons; the measurement error in the sonar sensors is relatively high and too high resolution will make it unattractive from a computational view point. The cells in the occupancy grid may also have irregular shapes with less accuracy in the space far from the robot and with higher accuracy closer to the center of the workspace.

B. Policy learning and planning

The RL-agent's objective is to learn a policy, a mapping from given states to actions, that leads to the highest accumulated reward. The reward is given at the end of the task when the agent has reached the goal or, if the agent fails, a punishment (negative reward) is received. By giving a small negative reward every time step the agent will also try to minimize the time spent in searching for the goal state. To avoid tedium when waiting for the agent to learn an acceptable policy, which can take a long time, the initial policy is obtained from the demonstration.

To shorten the planning sequence we will use the Dyna-Q algorithm (introduced by Sutton [10]), which uses an internal model of the world by storing the next state and the expected reward for each experienced state-action pair. Dyna-Q provides an architecture for making *mental rehearsals*, i.e., planning or actions that take place only in simulation. This accelerates the learning and improves the performance without performing actual actions (i.e., indirect reinforcement learning). This is a very important property since it can prevent the manipulator from making real mistakes, which is not tolerable.

Putting it all together we must formalize the problem in the form of states, actions and rewards. This we can define as follows:

States	s	the joint space where the manipulator works, θ
Actions	a	joint velocities, $\dot{\theta}$
Reward	r	a scalar reward for reaching the goal position, given by the robot's internal sensors,
Punishment	$-r$	a negative reward given by the range sensors and for each time step.

The state space is continuous covering the manipulator's work space. The actions are the joint velocities, with appropriate limits for each joint.

When the agent explores it also indirectly improves the occupancy grid making it more reliable. Exploration should be possible even if the uncertainty is high in some regions, i.e., joint velocities can therefore be limited by an *internal supervisor*. The internal supervisor interprets the occupancy grid and decides what actions to forbid. It can also provide the trade-off between exploration and exploitation based on the distance to the goal and the time spent in search for the goal. Because of the collision risk the actions should further be limited by the range detectors, preventing the manipulator from moving into occupied states, by making some actions forbidden. The range sensors are also used for giving a negative reward signal.

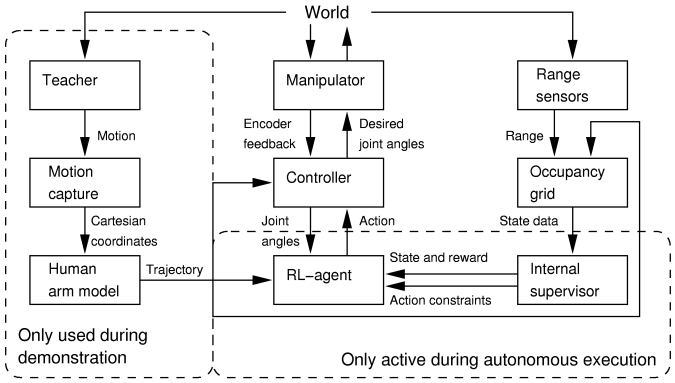

Fig. 2. A schematic overview of the system.

The RL-agent receives a reward signal when taking the end-effector to the goal position, this signal is generated by using the manipulator's internal sensors i.e., the encoders.

For continuous state and action spaces, Locally Weighted Learning (LWL), presented by Atkeson et. al. [11], will be used as a function approximator for the state-action values, i.e., the Q-values. LWL is an instance-based learning technique, hence no explicit training phase is needed, the training examples are simply stored in the memory and a generalization is made upon a query, which can make the query time a bit slow when storing large amounts of data.

C. From static to non-stationary environments

The demonstration is conducted in a static environment, where the RL-agent builds its internal model of the world. In parallel the 3D occupancy grid is filled with the perceived information. When the manipulator has performed the task, supervised by the teacher and trained by the mental rehearsals it can gradually become more and more autonomous, i.e., it takes actions according to the learned policy instead of the recorded supervised actions. The robot must also be able to cope with the situation that the environment changes after the initial learning phase is concluded.

In Figure 2 an overview is shown of how the different parts discussed in the above sections interact.

D. The platform

Different small sized manipulators will be used for testing the simulated results in a real environment. The range sensors can be mounted in several different ways, aligned with the links, perpendicular to the links, something in between or a combination, see Figure 3 for examples.

Another issue to investigate is whether the sensors will fail to detect an object, due to the effects of reflection.

Since we have chosen cheap sensors they might not be very reliable, so they should not be considered as a replacement of safety system to protect humans, but as a safety system for the manipulator. A natural extension is to also mount a range sensor on the gripper, making the approaching of objects safer.

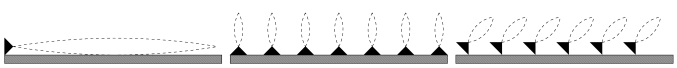

Fig. 3. Three different ways of mounting the range detectors onto the links. The filled triangles mark the sensor locations and the dashed lines the beam patterns.

IV. Action plan

This project will serve as a test bed for developing teaching manipulation tasks and to test techniques developed for mobile robots on a manipulator. The project's main steps will include the following:

- Develop an algorithm that can build a 3D occupancy grid from the teacher's motions,
- include the range sensors as a kind of "skin" for the system,
- investigate how the 3D model can be used as a dynamic map of a non-stationary environment,
- investigate how the Dyna-Q learning algorithm must be modified and extended to suit our needs in a non-stationary environment.

V. Discussion

Our proposed platform is capable of recording human actions and executing them on a robot manipulator, and with the integration of learning capabilities we are working towards a self-improving system with planning abilities and a basic obstacle avoidance strategy. In future research other behaviors will be investigated. It is our belief that manipulators of the future must be simple to program (teach) and need to possess basic behaviors such as avoiding collision, pick and place, trajectory following, etc.

References

[1] Y. Jiar, M. Wheeler, and K. Ikeuchi, "Hand Action Perception and Robot Instruction," Carnegie Mellon University, Tech. Rep., 1996.

[2] R. Dillmann, "Teaching and learning of robot tasks via observation of human performance," *Robotics and Autonomous Systems*, vol. 47, pp. 109–116, 2004.

[3] S. Schaal, A. Ijspeert, and A. Billard, "Computational approaches to motor learning by imitation," *Philosophical Transaction of the Royal Society of London: Series B, Biological Sciences*, vol. 358, no. 1431, pp. 537–547, 2003.

[4] C. K. Tham and R. W. Prager, "A modular Q-learning architecture for mobile manipulator task decomposition," in *Proceedings of the 11th International Conference on Machine Learning*. Morgan Kaufman, 1994, pp. 309–317.

[5] P. Martín and J. del R. Millán, "Learning reaching strategies trough reinforcement for a sensor-based manipulator," *Neural Networks*, vol. 11, pp. 359–376, 1998.

[6] J. Izawa, T. Kondo, and K. Ito, "Motor learning model using reinforcement learning with neural internal model," in *IEEE International Conference on Robotics and Automation*, 2003, pp. 3146–3151.

[7] M. T. Rosenstein, "Learning to Exploit Dynamics for Robot Motor Coordination," Ph.D. dissertation, University of Massachusetts Amherst, May 2003.

[8] J. M. Santos, "Contribution to the study and the design of reinforcement functions," Ph.D. dissertation, Universidad de Buenos Aires, 1999.

[9] A. Skoglund, R. Palm, and T. Duckett, "Towards a Supervised Dyna-Q Application on a Robotic Manipulator," in *Advances in Artificial Intelligence in Sweden*, P. Funk, T. Rognvaldsson, and N. Xiong, Eds. SAIS-SSLS, April 2005, pp. 148–153.

[10] R. S. Sutton, "Dyna, an Integrated Architecture for Learning, Planning, and Reacting," in *SIGART Bulletin*, vol. 2, 1991, pp. 160–163.

[11] C. G. Atkeson, A. W. Moore, and S. Schaal, "Locally weighted learning," *Artificial Intelligence Review*, vol. 11, no. 1-5, pp. 11–73, 1997.