Richard Cubek

Master Thesis Proposal

Robot Programming by Demonstration on a Higher Abstraction Level

Contents

1 Motivation

It is needless to say, that autonomous, intelligent robots will be part of our everydays life in the near future. In contrast to preprogrammeed industrial robots, a mobile service robot has to face the challenge of a changing, complex world. Even for a simple household environment with only a few objects and a few pieces of furniture, it is impossible to predict all possible situations, the robot will find itself in. Hence, it is also impossible to program a strategy for each situation beforehand, rather, the robot must be able to act useful in unknown situations by learning. There is a host of approaches, how to design such learning capabilities for robots. In Reinforcement Learning, the robot faces a known environment, but with unknwon state transition probabilities (system dynamics). Based on a received reward when acting, the robot (or a learning agent in general) can optimize its policy, which is nothing else than a function mapping states to actions. Applying approximation techniques, the robot can learn strategies for states, which have not been faced when interacting with the world. Another approach is to learn and update the world model stepwise by learning the transition probabilities when acting in the world with the intention, to apply dynamic programming algorithms on the model (e.g. Value Iteration).

Thinking of service robots, the robot should be able to solve tasks which are typical for household environments, for example so called pick-and-place tasks. This requires skills, which the robot has to aquire beeing teached and instructed by humans without any technical background. Such an approach is known as Programming by Demonstration, which is described below. This way of learning is the main topic of this thesis.

2 Programming by Demonstration

In Programming by Demonstration (PbD), also referred to as Learning from Demonstration, the human teaches the robot to solve a task. This teaching can be achieved in different ways. Doing kinesthetic teaching, the teacher directly moves the robots actuators while demonstrating. In other methods, the robot observes the human when doing a task, which by some authors is also denoted as Imitation Learning. Most often, more than one task demonstration is necessary. The main difficulty is to recognize the task constraints during the demonstrations. The key property of an agent regarding a demonstration learned skill is its ability of generalization, that is, applying a learned skill to situations that differ from those during demonstration. In general, the representation of a skill can take place on two abstraction layers: a low level representation (trajectory encoding) for generic motions and a high level representation (symbolic encoding) for sequences of predefined actions. In a corresponding way, one can divide PbD approaches in general [1].

3 Three-Layer Robot Architectures

Since the terms of symbolic and trajectory encoding are closely related to layered robot architectures, a short overview is given below.

A classical three-layer robot architecture consists, as the name implies, of three layers [5]. The lower layer is the so called reactive layer and is responsible for direct actuator motions. Coupling sensory data directly to motor control, this layer can react to sudden state changes in the environment. Typically, so called low level actions like arm movements, grasping or obstacle avoidance are implemented at this layer. The trajectory encoding of a demonstration learned skill can be situated at the reactive layer.

The upper layer is the so called deliberative layer, which has a world model on a higher abstraction level. Here, states are described in the form of objects, their properties and relations between objects, as for example it is done in first order logic. The available low-level actions of the robot, their preconditions and their effects are also known. Having the knowledge of the world and our possible actions, this layer generates sequences of actions to achieve specific goals. Since sequences of actions are nothing else then plans, the deliberative layer is responsible for planning.

Summarizing, one can speak of high-level (symbolic) planning at the deliberative layer, and of low-level (numerical) control at the reactive layer. The integration between the deliberative and the reactive layer is realized by the sequencing layer in the middle. This layer is also referred to as reactive plan execution mechanism, since it is responsible to call actions at the reactive layer or to trigger the deliberative one to plan or replan (e.g. if the plan could not be realized or the world state changed in a way that the current plan is not applicable anymore).

Properties of the reactive layer:

- low abstraction
- representation of low-level skills
- direct coupling between sensory data and motor control
- high frequency of world state updates
- high reactivity
- realizing short range goals

Properties of the deliberative layer:

- high abstraction
- representation of high-level skills
- low frequency of world state updates
- planning for long range goals

Summarizing the properties of the layers, we can derive the advantages (or drawbacks) of skill encoding at the symbolic or trajectory level, respectively.

Trajectory encoding:

- Generic representations of motions (pick-and-place, grasp, etc.).
- Generalization over different inital states, but only in similar situations.
- No representation of complex skills consisting of several actions.
- Portability to other robots very difficult.

Symbolic encoding:

- Generic representations of action sequences or task goals.
- Possible reproduction even in different situations.

- Portability to other robots.
- Realizing long range goals.
- Requires already implemented low-level skills for each robot.

4 Thesis

In this thesis, a concept of a high-level PbD approach will be developed and presented in a virtual environment. The concept is based on an already published idea of recognizing task goals with the intention to use a planner to achieve these goals [3, 4]. In the corresponding papers, the authors emphasize, that it is not modelled how to reach a goal storing trajectories, but rather what has been done during demonstration. This is achieved by segmentating the task into sequences of operations or abstract states (both terms are used). Hereby, a technique to detect and model spatio-temporal constraints within this sequences (and thus the task demonstration) is presented.

4.1 Contribution of the Thesis

1) Implementation of the published idea with modifications and additions (listed below). It is not part of the thesis, to implement a whole three-layer architecture, since the focus of the thesis lies at the deliberative layer and high-level PbD.
2) In [4] it is described literally, that not the hand trajectories are stored, but instead what has been done. While beeing more precisely in the thesis, the author will state the role of the presented method in the context of layered architectures.
3) A new approach, how abstract, static states could be extracted from the numerical, trajectory-based observation-data of motions will be suggested. The idea is based on the clustering of object positions over time. Optionally, the method will be implemented and tested.
4) In [4], a STRIPS based planner using breadth-first search is used for planning action sequences. In this work, the author will test another planner (see 4.5.2 for details).
5) The author will use a simpler and in his opinion more natural method to detect spatial constraints between objects based on mean deviation of relative or absolute positions.
6) Test of the developed method, solving tasks in a so called blocks-world within a virtual environment.

4.2 Virtual Environment

The virtual environment Openrave [2] will serve as the platform for the environment-modelling and the implementation of low-level skills (which are required for high-level PbD). The simulated robot will be an UNIMATE Puma 500. The advantage in comparison to the Katana is an additional degree of freedom, what makes the implementation of low-level skills much easier (and they are not the focus of the thesis).

The author just discovered and reported a serious bug in the graspplanning module of Openrave, which was patched. It can't be guaranteed, that other bugs sometimes won't lead to unexpected outcomes.

4.3 Workflows: Task Demonstration and Reproduction

The realization of the concept can be divided in two phases: Learning while observing a task demonstration and reproduction of the task. The respective workflows are outlined in Figure 1.

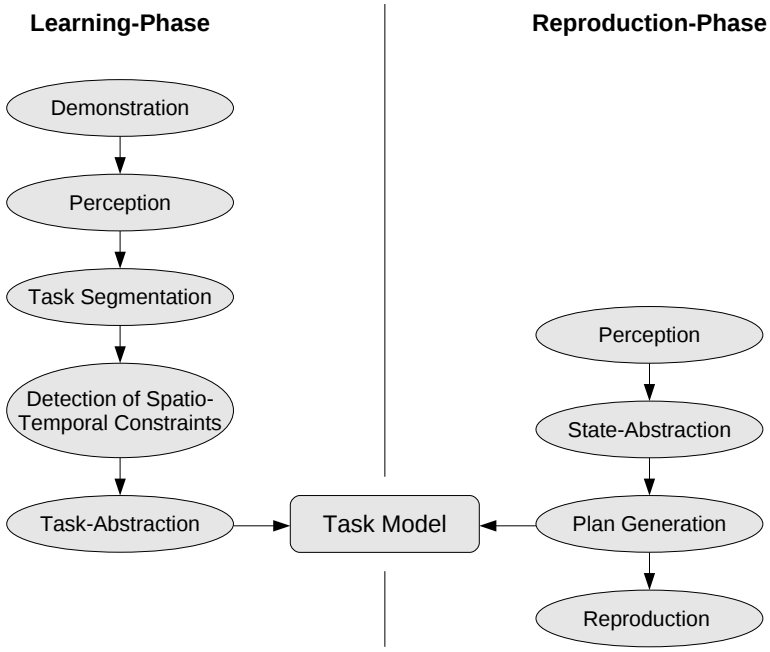

Figure 1: The workflows in the PbD concept.

4.4 Learning-Phase

4.4.1 Demonstration, Perception and Task Segmentation

The demonstration will be achieved simply by adding the objects to the desired positions (demonstrating the task) step by step, whereby each step represents a single action of object movement. The robot will take a scene snapshot at each of these steps (after completion of each action). This is similar to a real demonstration, where the human would signal the robot to take a snapshot (for example with an "ok") of the scene after a demonstration step. This automatically gives us the task segmentation. Optionally, tasks will be demonstrated using artificially generated trajectories or using a second robot to demonstrate the task. In the latter cases, a new task segmentation approach will be tested using clustering of object positions over time.

The visual perception is not part of the work, it will be simulated by directly reading out an object position and the addition of an gauss-distributed error, as it is realistic for most visual object-recognition systems.

4.4.2 Detection of Spatio-Temporal Constraints

Depending on the object positions during the demonstration, it will be determined, whether they have been placed in relation to another object or absolutely in order to detect the spatial constraints. The detection of temporal constraints is realized by the examination, where to keep a special order of object manipulation and where not.

4.4.3 Task-Abstraction

Depending on the spatio-temporal constraints (see 4.4.2 above), task constraints will be modelled using a relational (logical) language. The choosen language will depend on the used planner (because the planner has to use the abstracted task). If for example at the start, Object A is positioned at Location X in absolute coordinates, a possible statement could be *atAbs(A, X)*. The method of modelling the temporal constraints is still to be determined. A possibility is the use of a relational (logical) statement, e.g. *mustOccureBefore(A, X, B, Y)*. An alternative is the definition of intermediate goals (ordered subgoals) and a single plan generation for each of the subgoals. The task abstraction delivers the model of the task, which structure depends on the way to model temporal constraints. With subgoals, the model will contain the abstract, relational description of each of the subgoals in the desired order. If using relational statements for temporal constraints, the task model will containg relational statements for the end-goal only.

4.5 Reproduction-Phase

4.5.1 Perception and State-Abstraction

Perception is simulated as already described (4.4.1). State-Abstraction is very similar to the Task-Abstraction in the learning phase (4.4.3). The difference in the reproduction phase is, that we only have to model the initial state.

4.5.2 Planning

The planner takes the initial state and the task model (from 4.4.3) as input. Knowing the preconditions and effects of actions, it generates a plan to reach the desired task goal under the consideration of the constraints. The planner to use is still to be determined. There are several well known participants of the International Conference on Automated Planning and Scheduling (ICAPS), which could be tried. Another possibility is the use of an own planner, implemented by Value Iteration and a propositionalization with symbolic state sets.

4.5.3 Reproduction

At reproduction, the robot has to solve the task given initial states, that differ from those during demonstration. This initial state distinction can be characterized in a way, that trajectory-based skills could not be able to solve it.

References

[1] A. Billard, S. Calinon, R. Dillmann, and S. Schaal. Robot programming by demonstration. In B. Siciliano and O. Khatib, editors, *Handbook of Robotics*. Springer, 2008. In press.

[2] Rosen Diankov and James Kuffner. Openrave: A planning architecture for autonomous robotics. Technical Report CMU-RI-TR-08-34, Robotics Institute, July 2008.

[3] Staffan Ekvall and Danica Kragic. Learning task models from multiple human demonstrations. In *ROMAN 2006 - The 15th IEEE International Symposium on Robot and Human Interactive Communication*, pages 358–363, Univ. of Hertfordshire, Hatfield, UK, 2006.

[4] Staffan Ekvall and Danica Kragic. Robot learning from demonstration: A task-level planning approach. *International Journal on Advanced Robotics Systems*, 5(3):223–234, 2008.

[5] D. Kortenkamp and R. Simmons. Robotic systems architectures and programming. In B. Siciliano and O. Khatib, editors, *Handbook of Robotics*. Springer, 2008. In press.

List of Figures

List of Tables