ESANN 2013 proceedings, European Symposium on Artificial Neural Networks, Computational Intelligence and Machine Learning. Bruges (Belgium), 24-26 April 2013, i6doc.com publ., ISBN 978-2-87419-081-0.
Available from http://www.i6doc.com/en/livre/?GCOI=28001100131010.

Hierarchical Reinforcement Learning for Robot Navigation

B. Bischoff[1], D. Nguyen-Tuong[1], I-H. Lee[1], F. Streichert[1] and A. Knoll[2]

1- Robert Bosch GmbH - Corporate Research
Robert-Bosch-Str. 2, 71701 Schwieberdingen - Germany

2- TU Munich - Robotics and Embedded Systems
Boltzmannstr. 3, 85748 Garching at Munich - Germany

Abstract. For complex tasks, such as manipulation and robot navigation, reinforcement learning (RL) is well-known to be difficult due to the curse of dimensionality. To overcome this complexity and making RL feasible, hierarchical RL (HRL) has been suggested. The basic idea of HRL is to divide the original task into elementary subtasks, which can be learned using RL. In this paper, we propose a HRL architecture for learning robot's movements, e.g. robot navigation. The proposed HRL consists of two layers: (i) movement planning and (ii) movement execution. In the planning layer, e.g. generating navigation trajectories, discrete RL is employed while using movement primitives. Given the movement planning and corresponding primitives, the policy for the movement execution can be learned in the second layer using continuous RL. The proposed approach is implemented and evaluated on a mobile robot platform for a navigation task.

1 Introduction

Incorporating autonomous robots into daily life has been a long-standing goal in the robotics community. While modern service robots can fulfil simple tasks, e.g. vacuum cleaning, more complex tasks, such as manipulation and navigation, pose a remarkable challenge [1, 2, 3]. In such cases, it can be difficult to plan and execute the tasks due to uncertainties in the physical system, as well as in the environment. For example, for complex robot systems, controllers based on analytical models often fail to provide satisfying task performance [4]. Here, learning desired tasks from experience using machine learning techniques, e.g. reinforcement learning (RL) [5, 6], can offer an appealing alternative.

When learning tasks from data using RL, uncertainties in the environment can also be taken into account, as they are encoded in the sampled data [1]. Although RL techniques are well developed during the last decades, their major drawback is the inefficiency and lack of robustness for complex learning problems. For navigation tasks, for example, if the desired trajectory is sufficiently complex, learning a corresponding controller using RL can be difficult. To overcome this limitation and making RL feasible, hierarchical RL (HRL) has been suggested [7, 8]. Inspired by the idea of HRL, we propose a RL architecture appropriate for learning complex robot movements. The proposed approach includes two hierarchical layers: (i) motion planning and (ii) movement execution.

ESANN 2013 proceedings, European Symposium on Artificial Neural Networks, Computational Intelligence and Machine Learning. Bruges (Belgium), 24-26 April 2013, i6doc.com publ., ISBN 978-2-87419-081-0.
Available from http://www.i6doc.com/en/livre/?GCOI=28001100131010.

In the planning layer, the original complex motion of the robot is decomposed into movement primitives. By using the primitives, trajectory planning can be performed for the desired task. In this paper, we employ discrete RL [5] for the task planning based on decomposed primitives. Subsequently, the execution of primitives can be learned using continuous RL [5, 6] in the second layer. Given the trajectory and learned primitives, robot movements can be performed fulfilling desired tasks. Due to the straightforwardness of primitive movements, RL becomes more robust and efficient.

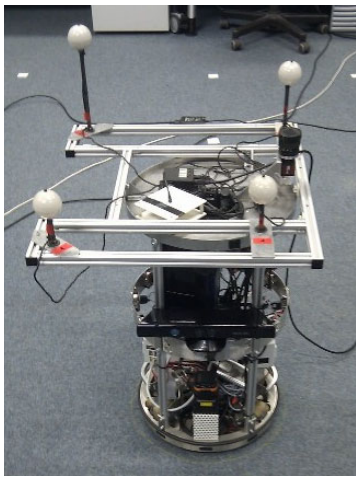

Figure 1: Service robot platform Festo Robotino used for evaluation.

In this paper, we evaluate the proposed approach for learning navigation tasks. Here, the robot needs to learn how to move from a starting point to a goal position. We first identify the primitives required for navigation. Using the primitives, discrete RL is employed to learn a feasible path from starting point to goal position while taking in account possible obstacles. The learned policy, i.e. the navigation path, can be updated online when new obstacles are detected. On the second level, we learn the execution of the primitives for the navigation path. In this work, we employ Value-Iteration [5] for discrete RL and Pilco [6] as continuous RL. For evaluation, we implement the proposed hierarchical approach on a mobile robot system, e.g. the Festo Robotino shown in Figure 1.

The remainder of the paper will be organized as follows: in Section 2, we briefly review the basic concepts of RL. In Section 3, we explain our navigation setting and show how the proposed approach can be realized in this framework. In Section 4, the approach will be evaluated on a real robot platform for an indoor navigation task. A conclusion and outlook will be given in Section 5.

2 Reinforcement Learning: A Review

In this section, we provide a short review on the basic concepts of RL [5]. We proceed to discuss the relevant RL methods, i.e. Value-Iteration [5] for discrete RL, and probabilistic inference for learning control (Pilco) [6] for continuous RL.

In RL, we consider an agent and its interactions with the environment. The state of the learning agent is defined by $s \in S$. The agent can apply actions $a \in A$ and, subsequently, moves to a new state s' with probability $p(s, a, s')$. A reward function $R : S, A, S \to \mathbb{R}$ determines the reward for given state transitions, a policy $\pi : S \to A$ determines in every state the action to be used. The goal of RL is to learn a policy π^* that maximizes the expected long-term reward, $\pi^* = \arg\max_\pi \sum_{t=0}^{\infty} \gamma^t E(R(s_t, a_t, s_{t+1}))$ for all start states $s_0 \in S_0 \subseteq S$. Here, γ is a discount factor with $0 < \gamma \leq 1$.

ESANN 2013 proceedings, European Symposium on Artificial Neural Networks, Computational Intelligence and Machine Learning. Bruges (Belgium), 24-26 April 2013, i6doc.com publ., ISBN 978-2-87419-081-0.
Available from http://www.i6doc.com/en/livre/?GCOI=28001100131010.

2.1 Discrete RL

Discrete RL considers RL problems with finite state space S. Value-Iteration [5] is an established algorithm of this well-studied field. The main idea is to estimate the value function $V_{\pi^*} : S \to \mathbb{R}$, which returns the expected long term reward for any state $s \in S$ under an optimal policy π^*. The value function V_{π^*} can be written as given in (1) according to the Bellman optimality equation. Value-Iteration determines a sequence of value functions V^k, see (2), which is guaranteed to converge to V_{π^*} under some mild assumptions [9].

$$V_{\pi^*}(s) = \sum_{s' \in S} p(s, \pi^*(s), s') \left[R(s, \pi^*(s), s') + \gamma V_{\pi^*}(s') \right] \tag{1}$$

$$\forall s \in S : \; V^k(s) = \max_{a \in A} \sum_{s' \in S} p(s, a, s') \left[R(s, a, s') + \gamma V^{k-1}(s') \right] \tag{2}$$

$$\pi^*(s) = \arg\max_{a \in A} \sum_{s' \in S} p(s, a, s') \left[R(s, a, s') + \gamma V_{\pi^*}(s') \right] \tag{3}$$

Equation (3) describes how the policy π^* can be derived given V_{π^*}. Before using the Value-Iteration approach, the transition probabilities $p(s, a, s')$ must be determined, either analytically by prior knowledge or learned from experience.

2.2 Continuous RL

For continuous RL tasks, i.e. problems with continuous states $s \in S$, Value-Iteration as described in the previous section cannot be employed. Many approaches are suggested for continuous RL [5]. Here, we consider Pilco [6], which belongs to the RL class of Policy-Search methods. Algorithm 1 provides the basic concept of episodic Policy-Search RL.

Algorithm 1
Episodic Policy-Search framework

1: **for** $e = 1$ **to** *Episodes* **do**
2: Rollout: apply policy on system, collect experience $\{s_i, a_i, s'_i\}$
3: optional: learn dynamics model $p(s, a, s')$
4: Improve policy by adapting policy parameters
5: **end for**

The Policy-Search algorithm Pilco [6] is a model-based RL algorithm, where Gaussian processes (GP) are used to model the dynamics $p(s, a, s')$. Given a Gaussian state distribution $p(s_t)$, Pilco analytically approximates the action distribution $p(a_t) = \pi(p(s_t))$ according to the policy π as Gaussian distribution. Accordingly, a Gaussian approximation of the distribution $p(s_{t+1})$ for the next state is derived based on the GP dynamics model. With this technique, a rollout $[p(s_0), p(s_1), \ldots]$ of a policy π can be simulated. The policy parameters can then be adapted to maximize the sum of expected reward

$$J(\pi) = \gamma^0 E\left[R(s_0, a_0, s_1)\right] + \ldots + \gamma^T E\left[R(s_{T-1}, a_{T-1}, s_T)\right] .$$

The policy is parametrized as radial basis function network (RBF) with squared-exponential kernels as basis functions.

ESANN 2013 proceedings, European Symposium on Artificial Neural Networks, Computational Intelligence and Machine Learning. Bruges (Belgium), 24-26 April 2013, i6doc.com publ., ISBN 978-2-87419-081-0.
Available from http://www.i6doc.com/en/livre/?GCOI=28001100131010.

3 HRL for Navigation

In this section, we describe in details how HRL can be employed for navigation tasks. First, a set of movement primitives is defined. Here, we choose to decompose the navigation trajectory in 8 movement primitives, e.g. forward, backward, left and right translation with 0.25m and 1m displacements. The navigation task can now be employed in two layers. On the first level, the state $s \in \mathbb{N}^2$ represents the robot's grid-cell position in a map. Possible actions are the movement primitives, $a \in \{1\text{m forward}, 1\text{m backward}, \ldots\}$. Given the grid map and a desired goal, a navigation policy can be obtained for this grid map using discrete RL, as described in Section 2.1. The navigation policy thus determines how the robot should move in each grid-cell in order to reach the desired goal, as illustrated in Figure 2. Technically, the RL reward function is defined as 1 at the goal position and 0 else. We further include a penalty for moving directly along walls to avoid the risk of collisions. Additionally, we set higher rewards on primitives with 1m displacements (compared to 0.25m displacements) leading to trajectory decompositions with less primitives. Furthermore, policy updates can be performed online in realtime to address dynamic obstacles.

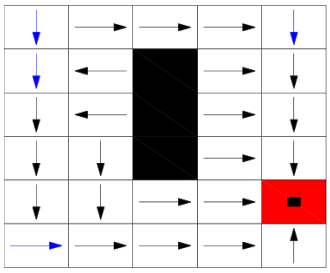

Figure 2: The figure shows an exemplary path policy. Black cells represent obstacles, the red cell is the goal position. Black arrows correspond to primitives with 0.25m displacements (blue arrows for 1m displacements).

The second hierarchical level corresponds to movement control. For each of the 8 movement primitives, a control policy is learned with continuous RL in the episodic setting, as described in Section 2.2. The state $s \in \mathbb{R}^3$ corresponds to x, y-position and yaw θ of the mobile robot, $(1, 0, 0)$ is e.g. the goal state for 1m forward translation. For an omni-directional robot drive, the control task is to adjust the rotational speed (and, thus, the action $a \in A$) of each wheel to follow the desired movement primitive. For the reward function of continuous RL, a saturated immediate reward is employed, $R(s, a, s') = R(s) = \exp\left(-\frac{|s-s_{\text{goal}}|}{2c^2}\right)$. Here, the hyper-parameter c describes the width of the reward.

4 Robot Evaluation

In this section, we evaluate the presented HRL approach on the mobile service robot platform Festo Robotino (see Figure 1). The omni-directional drive of the Robotino consists of three mecanum wheels. Additional devices are attached on the robot, for example, extra sensors, energy supply and localization devices. These additional build-ups significantly change the robots dynamics making the analytical control more difficult and, thus, motivates the use of machine learning for control. For robot localization, we use the A.R.T DTrack system which employs infrared markers on the robot detected by three external cameras.

ESANN 2013 proceedings, European Symposium on Artificial Neural Networks, Computational Intelligence and Machine Learning. Bruges (Belgium), 24-26 April 2013, i6doc.com publ., ISBN 978-2-87419-081-0.
Available from http://www.i6doc.com/en/livre/?GCOI=28001100131010.

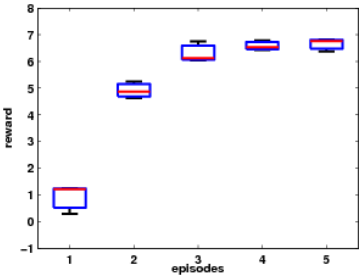

Figure 3: Learning results for 1m forward translation. The boxes show the reward for each episode averaged over 3 learning attempts.

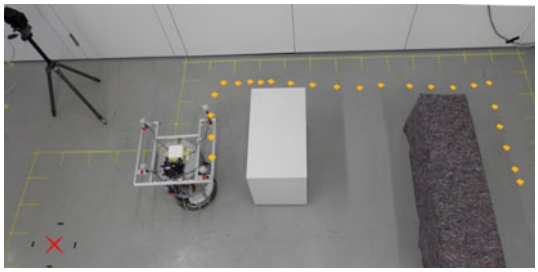

Figure 4: The figure shows a robot navigation task. Starting position is on the right, the goal is on the bottom left corner. HRL is employed for movement planning and execution. The yellow crosses mark the path already taken.

Task	Mean dist. error in cm	Mean angle error in deg.	# success. runs / total
1m forward	1.7	0.42	3/3
1m backward	1.48	0.57	1/1
1m right	2.97	0	1/1
1m left	2.28	1.2	1/1
0.25m forw.	2.49	0.90	3/3
0.25m back.	0.98	0.23	1/1
0.25m right	3.13	0.8	1/1
0.25m left	1.0	0.8	1/2

Table 1: Learning movement primitives using Pilco on the Robotino

As first step, we perform Value-Iteration on the upper hierarchical layer to learn a navigation policy given a map of our lab. The map is divided in a 18×9 grid of squared cells with 0.25m edge length. Even for larger maps, the navigation policy can be learned within a few seconds (e.g. 2.5 seconds for a map with 1000 cells). To address dynamic obstacles, policy updates can be performed in real-time. For example, the computation time for the policy update is less than 0.15s in our experiment. For larger maps, updates can also be performed locally around the obstacle to decrease computation time.

In the next experiment, the 8 movement primitives, forward, backward, left and right translation with 0.25m and 1m displacements, are learned as described in Section 2.2 and 3. The policy is implemented as RBF with 50 support points, in every episode a rollout consisting of 10 samples $\{s, a, s'\}$ is performed. Actions $a \in A = [-827, 827]^3$, which correspond to rotational speeds for the three-wheels, are applied constantly for 1 second. Figure 3 exemplary shows the learning result for 1m forward translation. The x-axis of the figure shows the learning episodes, the accumulated saturated immediate reward with $c = 0.25$ of three independent learning attempts is plotted on the y-axis as boxplot. The figure shows that the primitive can be learned in only 4 episodes in a robust manner. Table 4 gives the learning results for all 8 primitives. As evaluation, we consider the Euclidean distance from the last position to the goal state as distance error (in cm). The angle error is defined analogously (in $degree$). The results show that all directions seem to be almost equally difficult and sufficiently accurate policies can be learned for all primitives.

ESANN 2013 proceedings, European Symposium on Artificial Neural Networks, Computational Intelligence and Machine Learning. Bruges (Belgium), 24-26 April 2013, i6doc.com publ., ISBN 978-2-87419-081-0.
Available from http://www.i6doc.com/en/livre/?GCOI=28001100131010.

Finally, a robot navigation task with static and dynamic obstacles is performed, while combining the two hierarchical layers. The setting is shown in Figure 4. A video showing the policy learning on both layers as well as the navigation task with obstacles is available as supplementary material:
`http://www.youtu.be/6CWmoTHa968`.

5 Conclusion and Outlook

Inspired by the idea of HRL, we proposed a RL architecture appropriate for learning complex robot movements. The proposed approach includes two hierarchical layers for movement planning and execution, using decomposed movement primitives. Due to the simplicity of primitive movements, planning and execution using RL becomes more robust and efficient. In this paper, we employed the proposed approach for robot navigation while showing how the navigation task can be realized in this framework. We evaluated the approach on a mobile robot platform. The results show that the hierarchical architecture is suitable for learning robot navigation. Especially, the learned navigation policy, i.e. the movement planning, can be updated online and in realtime taking in account dynamic obstacles. In the next steps, we investigate possibilities for the automatic decomposition of complex trajectories into movement primitives. This can be done using machine learning methods, for example [10]. Further applications of this framework, e.g. robot manipulation, will also be considered in future work.

References

[1] A. Cassandra, L. Kaelbling, and J. Kurien. Acting under uncertainty: Discrete bayesian models for mobile-robot navigation. In *IROS*, pages 963–972, 1996.

[2] Takanori Fukao, Hisashi Nakagawa, and Norihiko Adachi. Adaptive tracking control of a nonholonomic mobile robot. *IEEE Transactions on Robotics*, 16(5):609–615, 2000.

[3] William D. Smart and Leslie Pack Kaelbling. Effective reinforcement learning for mobile robots. In *ICRA*, pages 3404–3410, 2002.

[4] Duy Nguyen-Tuong, Matthias W. Seeger, and Jan Peters. Model learning with local gaussian process regression. *Advanced Robotics*, 23(15):2015–2034, 2009.

[5] Richard S. Sutton and Andrew G. Barto. Reinforcement learning: An introduction. *IEEE Transactions on Neural Networks*, 9(5):1054–1054, 1998.

[6] Marc Peter Deisenroth and Carl Edward Rasmussen. Pilco: A model-based and data-efficient approach to policy search. In *ICML*, pages 465–472, 2011.

[7] Andrew G. Barto and Sridhar Mahadevan. Recent advances in hierarchical reinforcement learning. *Discrete Event Dynamic Systems*, 13, 2003.

[8] Bernhard Hengst. Hierarchical reinforcement learning. In *Encyclopedia of Machine Learning*, pages 495–502. 2010.

[9] Eugenio Della Vecchia, Silvia Di Marco, and Alain Jean-Marie. Illustrated review of convergence conditions of the value iteration algorithm and the rolling horizon procedure for average-cost mdps. *Annals OR*, 199:193–214, 2012.

[10] Dana Kulic, Wataru Takano, and Yoshihiko Nakamura. Online segmentation and clustering from continuous observation of whole body motions. *Transactions on Robotics*, 25(5):1158–1166, 2009.