2013 IEEE/RSJ International Conference on
Intelligent Robots and Systems (IROS)
November 3-7, 2013. Tokyo, Japan

Learning Sequential Tasks Interactively from Demonstrations and Own Experience

Kathrin Gräve and Sven Behnke

Abstract— **Deploying robots to our day-to-day life requires them to have the ability to learn from their environment in order to acquire new task knowledge and to flexibly adapt existing skills to various situations. For typical real-world tasks, it is not sufficient to endow robots with a set of primitive actions. Rather, they need to learn how to sequence these in order to achieve a desired effect on their environment. In this paper, we propose an intuitive learning method for a robot to acquire sequences of motions by combining learning from human demonstrations and reinforcement learning. In every situation, our approach treats both ways of learning as alternative control flows to optimally exploit their strengths without inheriting their shortcomings. Using a Gaussian Process approximation of the state-action sequence value function, our approach generalizes values observed from demonstrated and autonomously generated action sequences to unknown inputs. This approximation is based on a kernel we designed to account for different representations of tasks and action sequences as well as inputs of variable length. From the expected deviation of value estimates, we devise a greedy exploration policy following a Bayesian optimization criterion that quickly converges learning to promising action sequences while protecting the robot from sequences with unpredictable outcome. We demonstrate the ability of our approach to efficiently learn appropriate action sequences in various situations on a manipulation task involving stacked boxes.**

I. Introduction

Robots today are typically used for highly specialized but repetitive tasks and their software is tailored to an isolated and controlled environment. On the other hand, considerable progress is being made on the development of humanoid robots that eventually may become a part of our everyday life. Endowing robots with the skills to interact safely with humans and to solve meaningful tasks in a diverse and dynamic environment remains a challenging problem. In practice, foreseeing every situation a robot may encounter is clearly infeasible which is why learning algorithms are actively being developed as a means of efficient knowledge transfer and to allow a robot to adapt its skills under varying starting conditions. A common assumption is that complex actions can be factorized into a sequence of elementary and often goal-directed movements. Accordingly, a substantial body of work addresses the problem of representing such motion primitives and investigates how they can be transferred to a robot efficiently [1]. Being endowed with a set of motion primitives alone does not enable a robot to perform complex tasks. Rather, it requires the ability to sequence primitive skills to achieve effects on the environment through skillful manipulation of objects. Depending on the situation, a different set of primitives may need to be combined in order to achieve the same effect.

In this work, we propose a system that combines reinforcement learning and imitation learning to teach a robot how to sequence motion primitives in order to solve a complex task. We assume that the value of actions and situations varies smoothly in the chosen representation and can, hence, reasonably be approximated by a Gaussian Process. By applying Gaussian Process Regression [2], we obtain an estimate on the value of an action sequence in a particular situation, along with an associated uncertainty of the prediction. Generalizing reward experienced from learned sequences this way, our approach is able to derive proper action sequences for similar tasks under varying conditions. This formulation draws on our previous work on single motion primitive learning [3], [4] which we extend here to variable-length movement sequences and a discrete task representation. Applying Gaussian Process Regression to action sequences entails a set of unique challenges, among them the need to consolidate continuous state spaces with inherently categorical spaces of action sequences. In this work, we employ a composition of task-specific kernels to incorporate different representations of tasks and action sequences of different lengths into the Gaussian Process framework.

In our system, we integrate learning from demonstrations and autonomous improvement as two alternate control flows. In contrast to many existing approaches that merely use human demonstrations during a bootstrapping phase for reinforcement learning [1], we decide in every situation which method to apply based on the available knowledge, in order to take advantage of the complementary strengths of both methods. If there is only little information available for the situation at hand, the predictions made by the Gaussian Process have a large uncertainty and searching for a promising action sequence might be too risky. In this case, our system will ask for a human demonstration. On the other hand, placing this burden on the demonstrator is unnecessary if sufficient data is available. In this case, autonomous improvement of the policy will likely produce better results.

To quickly converge our policy towards optimal action sequences, we devise an exploration policy based on optimizing the *expected deviation*. By trading the *expected improvement* [5] which is well known in the Bayesian global optimization literature and its counterpart, the *expected*

The authors are with the Autonomous Intelligent Systems group, Department of Computer Science, University of Bonn, Germany. graeve@ais.uni-bonn.de This work was supported by the B-IT Research School.

degradation, this criterion leads to an optimization process that avoids unknown areas of the state space and potentially unsafe action sequences. Enabling a safe operation of robots is of particular importance for practical applications where robots solve tasks in collaboration with humans. More generally, we believe the application of learning principles well known from human behavioral sciences contributes to the accessibility of a technical learning system which in turn improves its acceptance among potential users.

The remainder of this paper is organized as follows: After a brief review of related work in Sec. II, we describe our combined approach in Sec. III and detail Gaussian Processes with the combination of different kernels. We evaluate our approach on a manipulation task involving stacked boxes as described in Sec. IV.

II. Related Work

Over the last decade, a lot of work has been dedicated to endowing robots with primitive skills in the form of motion primitives [6]. The results have motivated recent research on the open question: how can robots learn to sequence primitive actions in order to solve complex tasks.

One direction of research formulates the sequence learning problem as a planning task. Toussaint et al. [7] propose an approach based on a model of the environment and the effects of actions on it. They translate relational rules capturing the effects of actions into a Bayes network, allowing to infer predictions of future states and rewards. Based on this representation, they devise a probabilistic planning algorithm that incrementally selects actions based on predicted reward sequences. Once a task level action has been identified, motor commands to execute the corresponding low-level motion are generated using stochastic optimization. In the approach of Abdo et al. [8], preconditions and effects of actions are learned from human demonstrations of a movement. These serve as input to a general-purpose planning algorithm, allowing a robot to generate sequences of manipulation movements. Assuming a suitable segmentation of the demonstration is provided, the authors proceed to analyze the variance in state at both ends of demonstration segments to identify preconditions and effects of actions. Generalization is performed by analyzing multiple demonstrations of the same task, or by interactively asking the teacher to relax preconditions by hinting at irrelevant task features. One challenge inherent to planning-based approaches is to find a suitable symbolic representation of the environment.

Several approaches circumvent the choice of a symbolic representation by tackling the problem from a reinforcement learning perspective. Daniel et al. [9] developed a reinforcement learning approach called Hierarchical Relative Entropy Policy Search. In their approach, movement primitives are parametrized using a dynamical system model. For a given number of segments, the sequence learning problem is formulated as a constrained optimization problem whose solution yields the parameters of the dynamical system. They reported results where a robot arm achieved a reasonable performance within 300 episodes of a robot hockey game.

To find a good policy more quickly, Stulp et al. [10] use a demonstrated trajectory to initialize the optimization of shape parameters of primitive movements. In their approach, they extend the PI^2 algorithm [11] to optimize the shape and goal parameters of a sequence of movement primitives. Both approaches are limited to a predefined number of primitives, restricting the ability of the robot to optimize its policy.

To learn sequences of actions, graph-based representations on the task-level have been proposed in conjunction with reinforcement or imitation learning methods. Konidaris et al. [12] encode generalized action sequences in skill trees. Sequences may be trained either by demonstrating viable solutions or by explorative reinforcement learning. Multiple sequences achieving the same goal from different starting points are then merged to a tree by considering their statistical similarities in reverse order. The authors note in their experiments that the availability of demonstrations may greatly accelerate the learning process. Similar to our approach, skill trees encode a deterministic task policy that does not require planning at runtime. In contrast, we propose an interactive approach that integrates imitation and reinforcement learning at task level using a probabilistic decision. Kulić et al. [13] recently proposed a framework where different motion sequences are encoded as alternate paths in a motion primitive graph. Demonstrated trajectories are segmented and clustered hierarchically to motion primitives which are subsequently encoded as Hidden Markov Models for retrieval and motion reproduction. In the motion primitive graph, nodes correspond to actions and edges are labeled with transition probabilities learned from the observed data.

Our work presented here is in line with the model-free approaches outlined above in that we require neither extensive planning at runtime nor the availability of a dynamical model of the robot. Instead, we propose to learn a probabilistic approximation of the state-action sequence value function using a combination of imitation and reinforcement learning that puts the teacher in the loop where appropriate. One main focus of our work is the safety of human-robot interaction. To this end, we propose an efficient yet safe optimization strategy to guide reinforcement learning.

III. Proposed Method

In this paper, we are concerned with the task of combining reinforcement and imitation learning to teach a robot how to apply action sequences to solve a task. In our approach, we approximate the value function on the combined space of states and action sequences by a Gaussian Process (GP) to generalize observed values to similar situations and action sequences. Based on this information, we decide in every situation whether a valuable and safe action sequence can be generated or whether a human demonstration is necessary. During imitation learning, demonstrations are segmented and classified to obtain a symbolic representation of the action sequence, suitable for the kernel used by the Gaussian Process. Reinforcement learning in our approach is based on a Bayesian optimization criterion involving the expected

deviation to safely and efficiently find promising action sequences.

A. Probabilistic Model

Let $\mathcal{S}$ be the partially continuous state space of the environment and $\mathcal{A}$ denote the space of action sequences from a predefined set of motion primitives parametrized with reference points. To generalize the observed values and to establish a guide for autonomous improvement, we approximate a task-dependent scalar value function on the combined state-action sequence space with a Gaussian Process

$$Q : \mathcal{S} \times \mathcal{A} \to \mathbb{R} \sim \mathcal{GP}\big(m(\vec{x}), k(\vec{x}, \vec{x}')\big),$$

where m and k denote the mean and covariance functions of Q, respectively. This allows us to generalize the values of known situation-action sequence pairs and to obtain predictions for varied action sequences or unknown situations. For every prediction, we also obtain an associated uncertainty which reflects the dissimilarity of the query point from the training data. Assuming that action sequences which are dissimilar from those previously seen are potentially unsafe, we interpret the uncertainty together with the mean of the GP prediction as a measure of safeness of an action sequence and use it for making a decision for one of the two learning methods and to guide reinforcement learning.

While Gaussian Process Regression (GPR) is often used on Euclidean inputs, one of its main benefits—and more generally of kernel methods—is that it can be applied to a wide range of representations, provided a valid covariance function k can be defined. In particular, action sequences of variable length aren't reasonably represented by Euclidean vectors. Therefore, in our approach, we define the kernel by combining distance kernels on the application-dependent state space and the space of action sequences.

For example, in our experiments described in Sec. IV, states are defined on an Euclidean space whereas action sequences are described using sequences of motion primitive identifiers and corresponding reference points. To obtain a notion of similarity among these inputs, we combine a Gaussian kernel with the string subsequence kernel first proposed by Lodhi [14] for text classification. Since the space of individual actions forms a finite set, it is only endowed with a discrete metric, i.e. every action is dissimilar to all others. Considering all ordered subsequences of an action sequence leads to an elastic distance that accepts inputs of different length and allows us to generalize value across action sequences. As subsequences do not necessarily have to be contiguous, the kernel gracefully handles dissimilarities due to gaps in the inputs. The subsequence kernel over action sequences is defined as:

$$k_{\mathrm{str}}(x, x') = \sum_{u \in \mathcal{A}^*} \phi_u(x)\phi_u(x'), \tag{1}$$

$$\phi_u(x) = \sum_{\mathbf{i}: u = x[\mathbf{i}]} \lambda^{l(\mathbf{i})}, \tag{2}$$

where $\mathcal{A}^*$ denotes the set of all sequences with elements from $\mathcal{A}$, $\mathbf{i}$ is a sequence of length $|u|$ of monotonically increasing

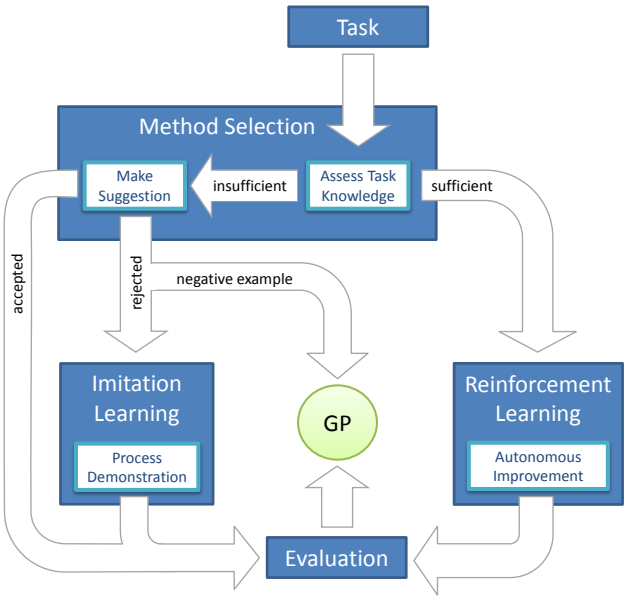

Fig. 1. Schematic overview of the proposed system

indices from $[1, \ldots, |x|]$, $x[\mathbf{i}]$ denotes the set of elements of x identified by $\mathbf{i}$, $l(i)$ is the length of the subsequence $x[\mathbf{i}]$ in x and $\lambda \in (0, 1]$ is a decay factor. In our experiments, we construct a kernel on the combined state-action sequence space by taking the product of subsequence and Gaussian kernels which, as shown in [2], is again a valid kernel for a Gaussian process:

$$\begin{aligned} k(x, x') &= k_{\mathrm{rbf}}(s, s') \cdot \hat{k}_{\mathrm{str}}(a, a'), \\ \hat{k}_{\mathrm{str}}(a, a') &= k_{\mathrm{str}}(a_{1:p-1}, a'_{1:q-1}) \cdot k_{\mathrm{str}}(a_{p:n}, a'_{q:m}). \end{aligned} \tag{3}$$

Here, $x = (s, a)$ denotes a state-action sequence pair, m and n denote the lengths of action sequences a and a' which are not necessarily the same and $p \in [1, n]$ and $q \in [1, m]$ are used to split the inputs a and a' into a sequence of motion primitive ids and a sequence of associated reference points.

B. Learning Method Selection

Since the scenarios where learning from demonstration and reinforcement learning can make use of their strengths best are quite complementary, we integrate them as two alternate control flows in our approach as depicted in Fig. 1. When asked to provide a sequence of actions that fulfills a task in a given situation, our system decides whether the previously collected task knowledge is sufficient to autonomously generate a reliable solution, or whether human assistance is needed. This is in contrast to related approaches that often restrict the human expert to a bootstrapping phase and exclusively rely on autonomous improvement thereafter. Our approach offers a trade-off that involves the human expert only where necessary, thus keeping the number of expensive demonstrations at a minimum. On the other hand, it allows the system to adapt to changes in the environment.

To make the decision for one of the learning methods, we consider the predicted value of previously experienced situation-action sequence pairs, combined with the similarity

to the current situation. To find a training example $\hat{x} = (\hat{s}, \hat{a})$ that has high value and is akin to the current situation s_{curr}, we search for a minimum of

$$\hat{x} = \underset{(s,a)\in\mathcal{X}}{\operatorname{argmin}} \, \alpha \left(Q_{max} - \mu(s,a)\right) + \left\| s_{\text{curr}} - s \right\|_2 , \qquad (4)$$

where $\mathcal{X} \subseteq \mathcal{S} \times \mathcal{A}$ is the set of positive training examples, Q_{max} is the optimal value, and α expresses the relative preferences for high value or similarity of the situation.

If a valuable action sequence in a similar situation is found, i.e. its score undercuts a predefined threshold θ, it is taken as a starting point for autonomous improvement as described in Sec. III-D. Otherwise, expert knowledge is requested. The threshold θ determines the carefulness of the system. With a large θ, the system will lean on expert knowledge less often. On the other hand, the outcome of action sequences becomes less predictable and the risk of failing the task increases. Decreasing θ reduces the risk at the cost of an increased human effort.

In order to reduce the cost of providing demonstrations to the system if no suitable action sequence was found, our system first attempts to suggest the best action sequence found before asking for a demonstration. The expert is then asked to either approve or reject this suggestion. In the latter case, a demonstration is requested and a new training sample is added to the Gaussian Process to remember the sequence as being unsuited for the task in the current situation. The demonstrated action sequence or an accepted suggestion is executed to obtain a reward and is added as new GP sample.

C. Imitation Learning

In our approach, a demonstration consists of a trajectory comprising a sequence of actions that together achieve a desired effect on the environment. To learn to reproduce the effect, we determine the ordering of individual actions by segmenting the trajectory and labeling its components according to a library of pre-trained motion primitives. At the same time, we determine the origin and the final points of the segments and assign each to a reference point from a predefined set by nearest neighbor classification. The reference points identify task-dependent locations in the environment rather than fixed coordinates. For example, they may correspond to the location of objects or landmarks in the world. To create a complete training sample for the Gaussian Process model, the sequence of motion primitives has to be executed with respect to the identified reference points and its reward needs to be computed.

In our implementation, we use an iterative segmentation and classification algorithm that relies on Hidden Markov Models (HMM) to encode spatio-temporal features characterizing actions [15]. By optimizing the likelihood of a segment ending at a point as well as the likelihood of the succeeding segment starting at that point, motion boundaries are delineated reliably. The HMM representation of motion primitives also allows to generate smooth trajectories for movement classes and to generalize them to arbitrary goals. In our experiments, we use trajectories generated this way to simulate the effect of action sequences on the environment.

D. Reinforcement Learning

While learning from demonstrations allows to transfer detailed task knowledge in close interaction with a robot, reinforcement learning allows to develop a policy autonomously in situations where human demonstrations are expendable, reducing human effort. In our approach, we approximate the reward the system receives for executing a chosen action sequence by a Gaussian Process. This allows us to obtain predicted values for candidate action sequences with an associated uncertainty. From these, we compute the *expected deviation* of an action sequence, which trades off the expected improvement versus its counterpart, the expected degradation. For a maximization problem, the expected improvement $\mathbf{E}D^{\oplus}$ at a point is the expected value of the predicted improvements over the maximum Q_{best} of all previous function evaluations. Similarly, the expected degradation $\mathbf{E}D^{\ominus}$ measures how much the function value is expected to deteriorate with respect to Q_{best}. In reinforcement learning, we select candidate sequences based on their expected deviation $\mathbf{E}D$ to rapidly converge our policy towards optimal action sequences:

$$\mathbf{E}D(x) := \mathbf{E}D^{\oplus}(x, Q_{\text{best}}) - f(\mathbf{E}D^{\ominus}, Q_{\text{best}}), \quad (5)$$

$$f(\mathbf{E}D^{\ominus}, Q_{\text{best}}) = \Big(Q_{max} - \big(Q_{\text{best}} - \mathbf{E}D^{\ominus}(x, Q_{\text{best}})\big)\Big)^2 .$$

At the same time, the expected deviation criterion ensures that possibly unsafe action sequences and sequences that are unlike any previously observed demonstration are avoided. We deem this an important feature of our approach for real-world scenarios because it increases the predictability of action sequences and mitigates the risk of damage from executing arbitrary sequences. The method used to optimize the expected deviation is not restricted by our approach and only depends on the action sequence representation. In our experiments, we use an action sequence representation that does not exhibit a meaningful notion of a gradient. Consequently, we employ a local neighborhood search that evaluates all sequences differing by at most one element from the known good sequence found in the decision step.

IV. Experiments

To validate our approach, we consider a tabletop manipulation setting where the task is to move a stack of boxes from one position to another safely. Instances of this simplified scenario are ubiquitous in our daily life, for example in setting a table or clearing up a desk. At the same time, the task is challenging because different manipulation strategies need to be applied depending on the situation at hand. For example, obstacles may prevent a direct movement towards the desired goal and instabilities of the object stack limit the number of stacked boxes that can be displaced at the same time without causing the stack to collapse. In this case, the system should learn to decompose the task and move only a limited number of boxes at a time. We report here results from experiments we conducted using the physics-based simulator Gazebo [16] to demonstrate the ability of our approach to safely learn a policy solving this task.

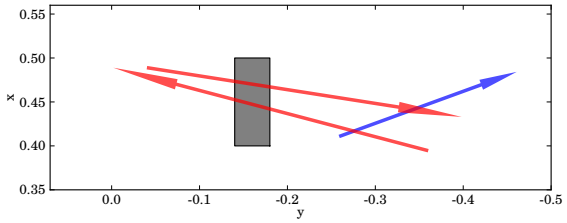

Fig. 2. Top-down view on the table with the obstacle (gray) and tasks where a human demonstration was requested depicted as arrows connecting the initial and target locations of the box. The color of the arrows indicates whether the box was lifted over the obstacle (red/light) or pushed to the target location (blue/dark).

A. Experimental Setup

In every episode of our experiments, a stack of boxes is placed in the workspace and one box is highlighted. The task is to move this box and all boxes on top of it to some target location in the workspace. All components of this setup, i.e. the initial position of the stack, its size, the highlighted box and the target location are chosen at random. In particular, the system may be tasked with moving an entire stack of boxes or only part of it, leaving a stub at the original location. The task is complicated by a static obstacle placed in the center of the workspace. Furthermore, our simulation models the physical interactions of the stacked rigid bodies. For our experiments, we set up friction coefficients such that the stack becomes instable and collapses if more than two boxes are relocated at the same time. To solve the task, the system is equipped with four primitive actions: grasping an object, displacing it, pushing it to a destination without lifting it, and retracting the hand. These primitive actions were trained using the approach described in our earlier work [15] and can be sequenced arbitrarily to create complex movements.

B. Task Definition

We measure the performance in relocating a stack of boxes as the work W spent throughout an episode of length T. It is defined as the change in kinetic energy E_k of the moved objects, which in turn is derived from the mass m of objects and their velocity v at time t:

$$W = \sum_{t=1}^{T} |E_k(t) - E_k(t-1)|\,; \quad E_k(t) = \frac{1}{2} m v^2(t). \quad (6)$$

At the end of an episode, the system is given a reward proportional to this value if it succeeds in relocating the stack. If its movements lead to a collision with the obstacle, a negative reward is assigned to discourage the action sequence that caused the collision. Action sequences that were rejected by the teacher are also given a negative reward according to the teacher's assessment. If relocating the boxes fails for any other reason, the system is given a reward of zero:

$$r(s,a) = \begin{cases} -1 & \text{collision or rejected suggestions,} \\ 0 & \text{failure,} \\ 1 - W & \text{success.} \end{cases} \quad (7)$$

In reinforcement learning, the Q value estimates the expected long-term reward for state-action sequence pairs. In our approach, an episode consists of a single task-level action, which abstracts an entire sequence of primitive actions, so Q is equal to r. Rather than approximating the reward function with a single fixed kernel width, we apply GPR for each component and combine the predictions in a Gaussian mixture model. This allows us to assign different weights to the components. Throughout our experiments, we set the kernel widths to 0.25 for the successful and 0.055 for the unsuccessful cases. By choosing narrower kernel widths for the latter, they are given a more local influence. The threshold θ used to decide for reinforcement or imitation learning was set to $\theta = 0.25$. This value provides a good trade-off between safety and the number of interactions with the human expert.

The state space $\mathcal{S}$ in our experiments consists of the task-space coordinates of the stack and the target position on the table, the number of objects in the stack and the number of the box on the stack that has to be moved to the target location. While states encode the locations of objects, task-level actions refer to objects in terms of their semantics in

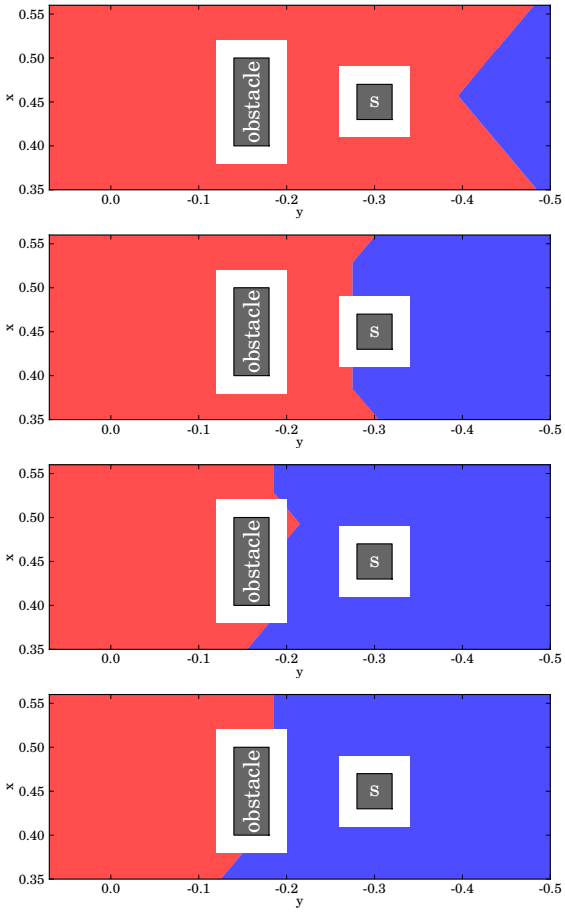

Fig. 3. Top-down view on the workspace depicting the evolution of the strategy of our reinforcement learning approach after 10, 20, 30 and 60 episodes. The gray box on the right side represents the start position s of the box which was fixed for this experiment. The movement sequence produced by our reinforcement learning module for arbitrary target locations is color coded. Target locations depicted in red (light) would lead to a sequence consisting of grasping, lifting and retracting whereas the box was pushed to locations depicted in blue (dark). Targets next to the initial position and the obstacle, depicted as white buffers, were not considered to avoid collisions.

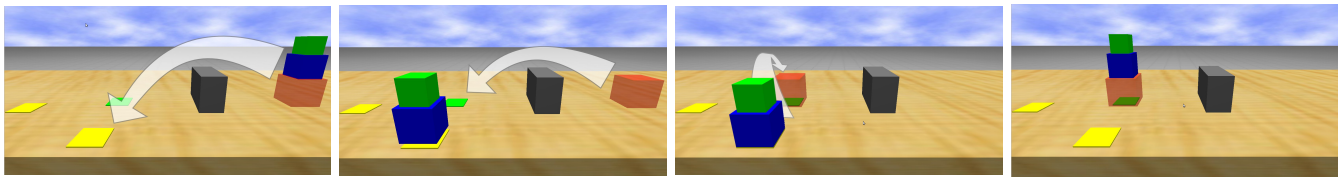

Fig. 4. Visual representation of the strategy required to relocate a stack of three boxes in our experiment. The target location is marked green, whereas yellow markers designate additional storage locations. The obstacle is represented by a gray box. Since three stacked boxes are unstable in our experiment, the stack has to be split in the first step by moving the top boxes to a temporary location. Then the stack's base may be moved to the target, where the stack is re-assembled in a third step.

the context of the task. We identify several locations that are relevant for the task with reference points, independent of their position. The set of reference points contains the initial and target locations of the stack, two locations where objects may be temporarily stored and a resting position the robot may assume between actions. All reference points have different levels within the stack. Since only the initial and target locations of the stack vary across episodes, their coordinates are the only ones that need to be encoded in the state vector. Task-level actions $a \in \mathcal{A}$ then consist of a variable-length sequence of motion primitive ids to be executed and a list of associated reference point ids.

For our experiments, human demonstrations are simulated using a statistical model trained on collected motion capture data [15]. Using the learned motion primitive models, action sequences generated by our system are converted to trajectories and executed by a controller that computes their effects on objects and updates a simulated environment accordingly.

C. Mutual Benefit of Imitation and Reinforcement Learning

To illustrate how our algorithm benefits from both imitation and reinforcement learning to quickly and safely find a policy, we consider in this experiment a simplified version of the task involving only a single box and random positions.

Fig. 2 depicts a top-down view of the table with the obstacle on it. Learning was started from scratch causing the system to ask for demonstrations during episodes one, three and seven. Prior to requesting a demonstration, the system recognized that it did not have enough information to generate an action sequence. Indeed, the best known action sequence suggested by the system would have caused a collision and was rejected by the trainer. Thereafter, the collected task knowledge was sufficient for the system to find solutions for further generated tasks autonomously.

In order to illustrate how the system incrementally gathers task knowledge across the state-action sequence space during the learning process, we generate a series of queries from a common starting point to target locations on a regular grid across the workspace. Fig. 3 depicts the distribution of action sequences proposed by the system at various points in time. Since our reward function penalizes the kinetic energy of the movement sequence, the system may achieve better results by pushing the object. On the other hand, pushing is only possible if no obstacle obstructs the direct path to the target location. The figure shows how the system suggests lifting for almost every target location after 10 episodes. At this time, all demonstrations depicted in Fig. 2 are already available. Lifting the box to locations in the right part of the workspace is sufficiently similar to previous examples involving lifting to be considered safe. On the other hand, there are not enough examples yet to reliably estimate the reward, leaving room for potential improvement. As more examples are gathered, the uncertainty of the reward estimates decreases and the system correctly selects the more energy efficient action where possible. Note that trading off expected improvement and degradation and involving the teacher, the system always produced action sequences suitable to solve the task and not a single collision with the obstacle was observed.

D. Variable-Length Policies

To demonstrate the ability of our system to efficiently learn a policy involving action sequences of variable length, we consider in this section the complete task with up to three stacked boxes. Fig. 4 depicts a strategy the system needs to learn in order to relocate a stack of three boxes. In our simulation, moving stacks with more than two boxes causes the stack to become unstable and collapse. As a consequence, relocating can only be achieved by first lifting a part of the stack, then moving the remaining stub to the target location and finally re-assembling the parts. Furthermore, since our reward function involves the kinetic energy of movements, better results can be achieved by pushing boxes to the target location. This is only possible if there is no obstacle in-between and the subject is located directly on the table. We report results attesting that our system captures all these task aspects and reliably generates safe action sequences without producing a single collision.

For this experiment, we ran 150 episodes with randomly generated tasks. Fig. 5 shows the sequence of decisions made by our system. Altogether, human assistance was

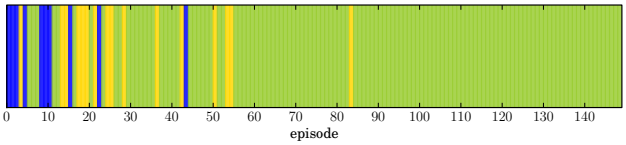

Fig. 5. Choices made by our system during an experiment with 150 episodes of the full task with up to three boxes. In episodes marked in blue (dark), our system asked for a demonstration after its initial suggestion was rejected by the teacher. Episodes where the suggestion was accepted are depicted in yellow (light). In the majority of episodes, highlighted in green (medium), our system selected an action sequence autonomously.

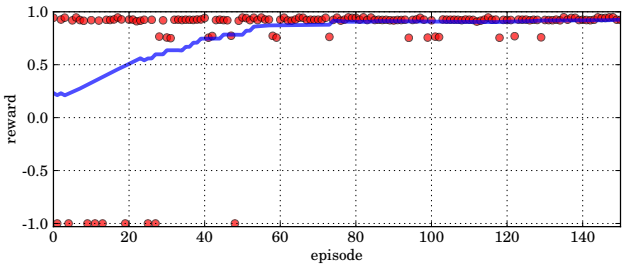

Fig. 6. Received rewards and their moving average over 50 episodes.

requested in 24 episodes. In the early episodes, little task knowledge has been gathered to autonomously generate an action sequence. Consequently, in 20 of the first 50 episodes, the teacher's assistance is requested and in 45 % of them the suggestion made by the system was rejected. This leads to nine requests for a demonstration of the task. Beyond this point, no further demonstrations were requested. As more experiences are gathered from demonstrations and successful autonomous attempts, the quality of suggestions increases and they are increasingly accepted by the teacher. At the same time, the number of instances where the teacher's approval of a suggestion is requested decreases. After 84 random episodes, an exhaustive policy was learned and the system no longer observed unknown situations requiring the teacher's consent. Since our exploration policy involves the uncertainty associated with predicted values, the system refrains from suggesting arbitrary action sequences and instead varies its actions within limits prescribed by previous experiences. Fig. 6 shows the rewards obtained throughout the learning process and their moving average over 50 episodes. While rejected suggestions led to several negative rewards during the first 50 episodes, only positive rewards were received thereafter. The outliers correspond to situations that required disassembling the stack. The additional actions require additional energy and lead to a reduced reward.

V. Conclusions

In this work, we presented a novel approach to learning action sequences of arbitrary length in order to solve complex tasks. Learning is achieved using a combination of reinforcement learning and imitation learning in a way that exploits their individual strengths and avoids their shortcomings. In every episode, the system makes an autonomous decision for either way of learning that is based on task knowledge acquired so far. We apply Gaussian Process Regression to generalize long-term rewards of action sequences across the combined state-action sequence space and make use of predictions and their uncertainty to assess the value of state-action sequence pairs. By combining continuous Gaussian kernels with string kernels, our approach gracefully handles states and action sequences of varying lengths comprising Euclidean coordinates and categorical values. We consider safety a major concern in applications involving close interaction of humans and robots and account for it by devising an exploration strategy during reinforcement learning based on expected deviation. This provides us with a trade-off among the expected improvement of an action sequence and the risk for the robot and its environment of executing the sequence in terms of similarity to previous experiences. We evaluated our approach on a typical manipulation task involving stacked boxes and reported results demonstrating the ability of our system to safely and efficiently learn the task from few human demonstrations. This was achieved by allowing the system to ask for demonstrations in situations where little task knowledge is available and executing the best guess would entail the risk of failure or even collision.

In future work, we would like to extend the formalism presented in this work to construct a coherent hierarchical system for robot learning, allowing to simultaneously learn complex tasks and low-level motion primitives.

References

[1] Brenna D. Argall, Sonia Chernova, Manuela Veloso, and Brett Browning. A survey of robot learning from demonstration. *Robotics and Autonomous Systems*, 57(5):469–483, May 2009.

[2] Carl E. Rasmussen and Christopher K. I. Williams. *Gaussian Processes for Machine Learning*. Adaptive Computation And Machine Learning. MIT Press, 2006.

[3] Kathrin Gräve, Jörg Stückler, and Sven Behnke. Learning Motion Skills from Expert Demonstrations and Own Experience using Gaussian Process Regression. In *International Symposium on Robotics (ISR) and German Conference on Robotics (ROBOTIK)*, 2010.

[4] Kathrin Gräve, Jörg Stückler, and Sven Behnke. Improving Imitated Grasping Motions through Interactive Expected Deviation Learning. In *Humanoid Robots (Humanoids)*, pages 397–404, 2010.

[5] Donald R. Jones. A Taxonomy of Global Optimization Methods Based on Response Surfaces. *Journal of Global Optimization*, 21:345–383, 2001.

[6] Dana Kulic, Danica Kragic, and Volker Krüger. Learning action primitives. In *Visual Analysis of Humans*, pages 333–353. 2011.

[7] Marc Toussaint, Nils Plath, Tobias Lang, and Nikolay Jetchev. Integrated motor control, planning, grasping and high-level reasoning in a blocks world using probabilistic inference. In *IEEE International Conference on Robotics and Automation (ICRA)*, 2010.

[8] Nichola Abdo, Henrik Kretzschmar, Luciano Spinello, and Cyrill Stachniss. Learning manipulation actions from a few demonstrations. In *IEEE International Conference on Robotics and Automation (ICRA)*, Karlsruhe, Germany, 2013.

[9] Christian Daniel, Gerhard Neumann, Oliver Kroemer, and Jan Peters. Learning sequential motor tasks. In *IEEE International Conference on Robotics and Automation, (ICRA), Karlsruhe, Germany*, 2013.

[10] Freek Stulp, Evangelos Theodorou, and Stefan Schaal. Reinforcement learning with sequences of motion primitives for robust manipulation. *IEEE Transactions on Robotics*, 28(6):1360–1370, 2012.

[11] Evangelos Theodorou, Jonas Buchli, and Stefan Schaal. A generalized path integral control approach to reinforcement learning. *Journal of Machine Learning Research*, 11:3137–3181, December 2010.

[12] George D. Konidaris, Scott R. Kuindersma, Roderic A. Grupen, and Andrew G. Barto. Robot learning from demonstration by constructing skill trees. *International Journal of Robotics Research*, 31(3):360–375, 2012.

[13] Dana Kulić, Christian Ott, Dongheui Lee, Junichi Ishikawa, and Yoshihiko Nakamura. Incremental learning of full body motion primitives and their sequencing through human motion observation. *The International Journal of Robotics Research*, 31(3):330–345, 2012.

[14] Huma Lodhi, Craig Saunders, John Shawe-Taylor, Nello Cristianini, and Chris Watkins. Text Classification using String Kernels. *The Journal of Machine Learning Research*, 2:419–444, 2002.

[15] Kathrin Gräve and Sven Behnke. Incremental Action Recognition and Generalizing Motion Generation based on Goal-Directed Features. In *IEEE/RSJ International Conference on Intelligent Robots and Systems, (IROS)*, pages 751–757, 2012.

[16] Nathan Koenig and Andrew Howard. Design and use paradigms for Gazebo, an open-source multi-robot simulator. In *IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, volume 3, pages 2149–2154, 2004.