3rd HAND
FP7-ICT-2013-10-610878
1 October 2013 (48months)

D3.1:
Scientific report on Protocols of Natural Instruction

Inria

⟨manuel.lopes@inria.fr⟩

Due date of deliverable: M12
Actual submission date: M12
Lead Partner: Inria
Partners: Inria, TUDa, USTT, UIBK
Revision: draft
Dissemination level: PU

This deliverable presents the results and plans on the protocols of natural instruction. We describe how a user can interact with a robot via demonstrations, a GUI, and gestures. Several capture systems are used to acquire information from those various modalities creating difficult challenges to merge and extract relevant information. We present an overall architecture for extracting hierarchical plans of assembly tasks from observing demonstrations and by interaction between the robot and the user. We describe several new algorithms able to: i) decompose a demonstration in simpler components; ii) learn high-level plans using inverse reinforcement learning; and iii) acquiring a symbolic representation of the task.

Executive Summary

This report presents the results of the first year on methods for Learning from Instructions (WP3). In this first period we developed new methods for interpreting demonstrations acquired using optical and magnetic tracking systems (WP1) in an hierarchical way. We created several segmentation methods to decompose a demonstration in simpler components: i) new method based on conditional random fields (CRFs); a segmentation methods based on clustering on DMPs; and iii) a graphical model to detect constraints.

To evaluate the results we considered metrics such as: precision of the detection of constraints; amount of data required to learn a task; robustness of the learned plans to changes and limited information; and number of corrections necessary during the interactive phase.

This report uses data from D1.1, it is complement with D4.1 for learning interactive motor primitives and the execution on the robot (simulation and real) is presented in D5.1.

Role of Protocols of Natural Instruction in 3rdHand

The work presented here gives the robot the fundamental skills to learn a task from a demonstration. This workpackage deals with all the interaction with the user both in an initial learning by interaction phase and later by allowing the user and the robot to work collaboratively. This first phase of work considered how to learn an assembly task from demonstration and how to interact with the user to present what was learned and allowing the user to provide further corrections.

In the future we will consider active requests for information, transfer of information between tasks and more in-depth collaborative tasks.

Contribution to the 3rdHand scenario

The results presented here contribute to the creation of an interactive robot by allowing it to understand how to execute a task by observing a person executing it. This process occurs not just in a passive way but also in an interactive exchange of information at several hierarchical levels. In more concrete terms we show how a complex tasks can be segmented in simpler elements that explain the creation of a constraint between objects. The same segments are also elementary unities that can be executed as a single motor primitives. We also show how to go from low-level information to more abstract plan level representations. These representation will allow the robot to better generalize between tasks and contexts. Also, to improve collaboration we need to create a shared understanding between the robot and the

user, and these representations are easier to visualize and to exchange with a person.

1 Tasks, objectives, results

1.1 Planned work

The goal of this task was: To develop a framework for robot instruction including the different types of instruction the user can provide and how they are used for learning. A first step will be to describe the different instructions types commonly used in the different algorithms and in natural human-human interactions. A second step will be to understand which signals can be better extracted in the particular scenarios we envisage. Resulting in protocols and algorithms to include multiple instructions to learn reward function to learn motor primitives (WP4) and manipulation plans (WP1).

It was only planned to integrate these capabilities in Milestone 2.

1.2 Actual work performed

The work plan was changed mostly to consider earlier integration of the skills in some of the robotic platforms already in Milestone 1. Parts of the study on human-human integration were delayed.

The work was divided in several axis presented next.

1.2.1 Interactive Learning Framework

This work considers all the factors that need to be taken into account to learn a complete description of a task in an intuitive way and efficient enough so that the effort of programming is reduced, and that the skills are reused. This framework will be updated with the results of the project.

The global workflow is presented on Figure 1 and described in [MMRL15]. The work on segmentation that allows to decompose a whole demonstration in shorter motions, is presented next. Each component represents a constraint and a motor primitive. The overall sequence of components is described as a relational plan.

1.2.2 Segmentation of Trajectories of Multiple Objects During a Manipulation Task

This work considered how an assembly task can be interpreted as a sequence of manipulations acts that change the constraints between pairs of objects. For this we created a generic CRF model that models a conditional probability distribution between the motions, and postures, of different objects. By representing the constraint between pairs of objects as a latent variable we can use several inference methods to estimate, for each time instant, if

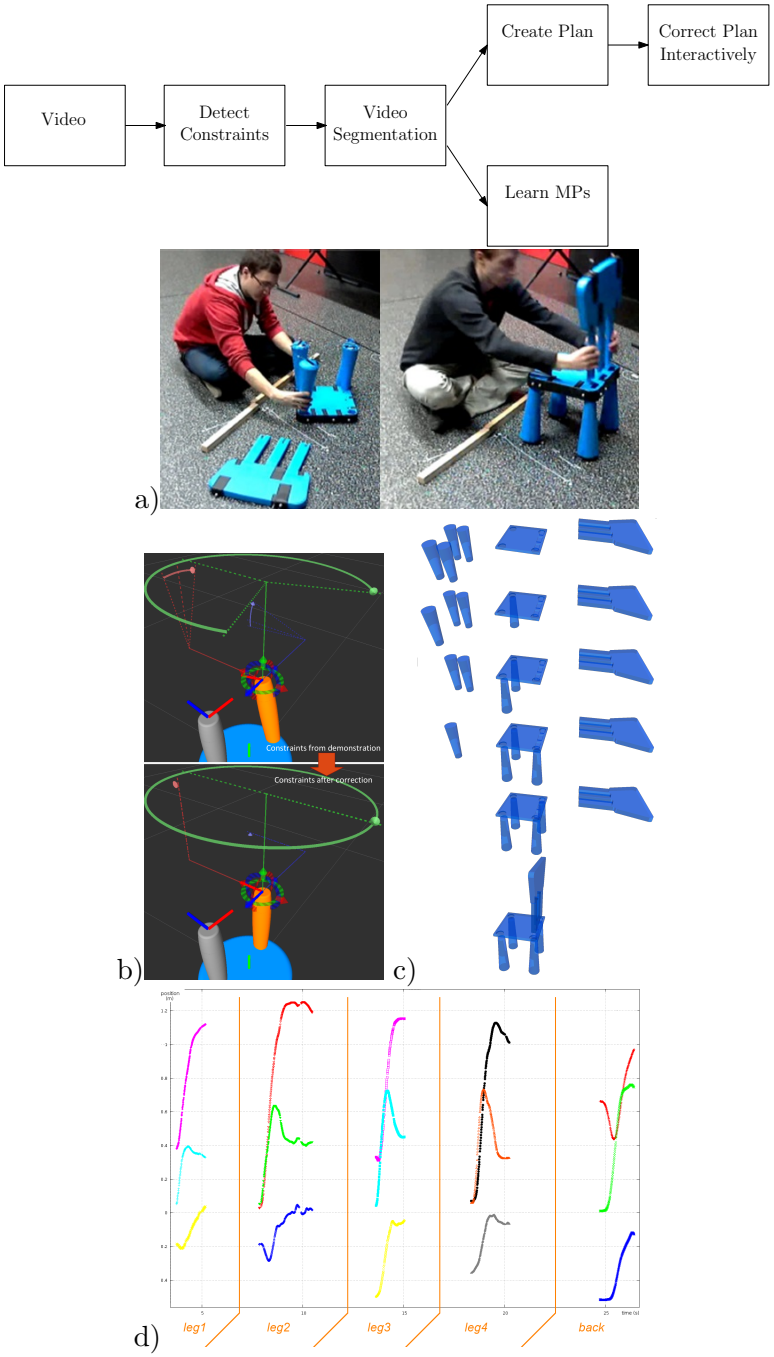

Figure 1: Interactive Learning Framework. Several processes need to be applied to a demonstration to recover a high-level task representation and motor primitives that can be reused in other situations. a) Task Demonstration; b) Interactive correction of the acquired knowledge using a GUI; c) Automatically detected key-frames of the demonstration; d) Motor primitives extracted from the demonstration.

there is a constraint between a pair of objects. The resulting segmentation provides information similar to the one presented in Figure 1. Full details of the algorithm are presented in [BML+15].

A complementary method is presented by [LMPN15]. Here the segments are the results of a clustering of motor primitives and not on the relations between objects. The rationale is that to learn true reusable motor primitives we have to rely on the particular kinematics of the objects and of the robot. A segment that represents the creation of a single constraint between two objects might require more than one

1.2.3 Learning of Relational Plans

An assembly task is a complex skill that includes both high-level planning and low-level action execution. The high-level plans have the capability to generalize better among different objects and situation and are easier to interpret by humans. To learn such plans we introduced a new inverse reinforcement learning method that works in the relational domain [MPG+15]. This method combines the advantages of relational learning with the ones of inverse reinforcement learning. Relational learning allows representing objects and work with a potential infinite number of objects, with the advantages of inverse reinforcement learning that allows to represent the cost function that explains the behavior of the user.

1.2.4 Head Tracking

We implemented a head tracking and head pose estimation system using Regularized Landmark Mean-Shift (RLMS) in combination with POSIT. For face tracking, an implementation of RLMS by [SLC11] was used. This method gives us 3D information for the feature points of the trained face-model. Using these features, tracked in 2D, together with the 3D information obtained during training, we can calculate an appropriate transformation of the 3D points so they coincide (after projection) with the 2D features. This gives us an estimation of the head pose relative to the pose determined during training. For obtaining this transformation, we used the OpenCV implementation of "Pose from Orthography and Scaling with Iterations" (POSIT) by [DD95]. Results of the head tracking and pose estimation can be seen in 2.

Additionally to the head pose estimation, the tracked points around the eyes are used to gather images for gaze estimation. The head pose– and gaze estimation will be part of a system to detect objects of interest. Together with hand pose estimation and gesture recognition they will aid in human robot interaction.

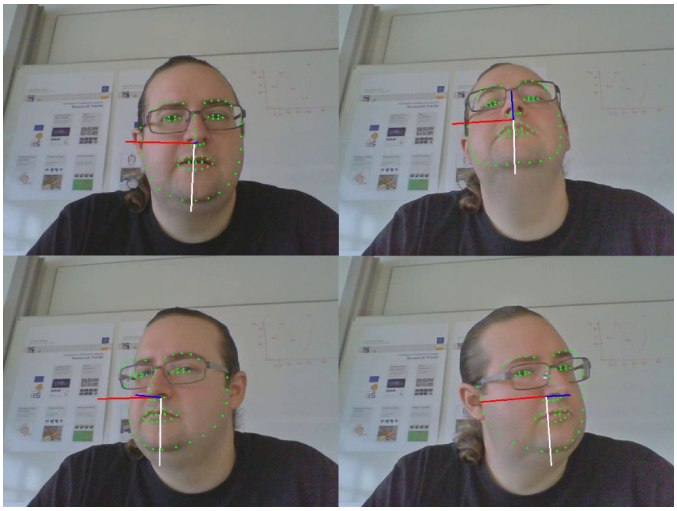

Figure 2: Head tracking and estimated pose for different orientations of the users head.

1.3 Relation to the state-of-the-art

1.3.1 Interactive Learning Framework

Learning from demonstration algorithms [Sch99, ACV09, Kha99] have focused on limited forms of demonstration, in most cases just working with a specific type of demonstration. Recently, many approaches started to consider a larger variety of demonstrations that go beyond trajectories acquired by means of optical tracking or kinesthetic teaching. This forms include Keypoints [ACJT12, AJCT11], learning from failures [GB11], active approaches [LMM09, CV09], and even the use of loosely specified protocols [GLO13]. In many cases we see a trend of unifying the training and execution part where the instructor is free to interrupt the system when a correction is needed [ML11], and also allows the robotic system to request further instruction when necessary [CV09].

Learning hierarchical representations of tasks in the robotic domain would consider learning high-level task plans and also low-level motor controllers. Works have mostly consider such levels separately even if several approaches already consider the execution of such hierarchical systems. A notably exception is [NOKB12, NCB+13] that presented an integrated approach to segment manipulation demonstrations into actions, each represented by a Dynamic Motion Primitive (DMP). In our work we extend this approach by including information about object relations and by learning high-level plans.

1.3.2 Segmentation of Trajectories of Multiple Objects During a Manipulation Task

Niekum et al. [NOKB12, NCB+13] presented an integrated approach to segment manipulation demonstrations into actions, each represented by a Dynamic Motion Primitive (DMP). The segmentation of demonstrations is realized with a Beta Process autoregressive HMM (BP-AR-HMM) [WSJF09], which not only produces a segmentation but also a corresponding labeling, i.e. an association with latent variable values. This approach is promising to identify segments of robot motion and compile these into DMPs. However, our aim is to identify the crucial interaction phases between manipulators and objects that signify the initial and final moments of an object manipulation. Therefore, in contrast to the problem setting of Niekum et al., we aim at a binary labeling of pair-wise interactions.

A series of works [WTS08, CP10, MTSS11, AML13] formulate integrated probabilistic models of sequential or superimposed motion primitives which, when fitted to data, imply a segmentation of motions. Again, the goal of these approaches is to extract specific motion primitives from data rather than to identify interaction phases. Barbic et al. [BSP+04] provide a good discussion of traditional methods for segmenting motion capture data, including the detection of zero crossings of angular velocities [FMJ02], and their own PCA approach. However, in all these approaches the problem setting focuses on the extraction of elemental motion primitives rather than analyzing pair-wise interaction phases between manipulators, tools and objects.

1.3.3 Learning of Relational Plans

When learning from observing another agent, we can aim at learning directly the behavior or, instead, the criteria behind such behavior. The former approach is usually called *Imitation Learning* (IL) while the latter is called *Inverse Reinforcement Learning* (IRL). The choice of one of the two hypothesis might be motivated by the application, the search for an explicit explanation for the behavior, the representation compactnesses or the capability of generalization among different problems. These two mechanisms can be used to model and explain many different social learning behaviors in animals [LMKSV09].

If an explanation is found for the behavior, as a reward function for instance, it is expected that the agent can generate the correct behavior in different situations, as it can plan its actions in those new situations. As a drawback, such approaches require some knowledge on both the original and the new domain. Another difference between IL and IRL is the compactnesses of the representation. Even in the same domain, there are tasks that are more compactly described as a reward function while others can be

better described as a policy. For instance, in a typical blocks world domain, the task of making a tower with all objects requires a non-trivial policy but can be described with a simple reward, see [DDRD01].

Another approach for generalizing to different problems is to use representations that are more expressive. The use of relational learning (also known as logical learning) allows one to generalize between worlds with different numbers of objects and agents. Learning to act from demonstration in relational domains have been a research problem for a long time [SD85, SD87, Kha99, YFG02]. The use of relational representations is attracting again more attention due to new algorithmic developments, new problems that are inherently relational and the possibility of learning the representations from real-world data, including robotic domains [LT10, LTK12].

Even though no approach to relational IRL has been proposed, this work is close to TBRIL [NJT+11]. The authors propose the use of gradient-tree boosting [Fri01] to achieve imitation learning in relational domains.

References

[ACJT12] B. Akgun, M. Cakmak, K. Jiang, and A.L. Thomaz. Keyframe-based learning from demonstration. *Inter. Journal of Social Robotics*, pages 1–13, 2012.

[ACV09] Brenna Argall, Sonia Chernova, and Manuela Veloso. A survey of robot learning from demonstration. *Robotics and Autonomous Systems*, 57(5):469–483, 2009.

[AJCT11] B. Akgun, K. Jiang, M. Cakmak, and A. Thomaz. Learning tasks and skills together from a human teacher. *AAAI Conf. on Artificial Intelligence*, pages 1868–1869, 2011.

[AML13] Javier Almingol, Luis Montesano, and Manuel Lopes. Learning multiple behaviors from unlabeled demonstrations in a latent controller space. In *Inter. Conf. on Machine Learning (ICML'13)*, Atlanta, USA, 2013.

[BML+15] Andrea Baisero, Yoan Mollard, Manuel Lopes, Marc Toussaint, and Ingo Lutkebohle. Temporal segmentation of pair-wise interaction phases in sequential manipulation demonstrations. In *Submitted to: Proceedings of 2015 IEEE International Conference on Robotics and Automation (ICRA)*, 2015.

[BSP+04] Jernej Barbi, Alla Safonova, Jia-Yu Pan, Christos Faloutsos, Jessica K. Hodgins, and Nancy S. Pollard. Segmenting motion capture data into distinct behaviors. In *Proceedings of the*

2004 Graphics Interface Conference, pages 185–194. Canadian Human-Computer Communications Society, 2004.

[CP10] Silvia Chiappa and Jan R. Peters. Movement extraction by detecting dynamics switches and repetitions. In *Advances in neural information processing systems*, pages 388–396, 2010.

[CV09] S. Chernova and M. Veloso. Interactive policy learning through confidence-based autonomy. *J. Artificial Intelligence Research*, 34:1–25, 2009.

[DD95] Daniel F Dementhon and Larry S Davis. Model-based object pose in 25 lines of code. *International Journal of Computer Vision*, 15(1-2):123–141, 1995.

[DDRD01] S. Džeroski, L. De Raedt, and K. Driessens. Relational reinforcement learning. *Machine learning*, 43(1):7–52, 2001.

[FMJ02] Ajo Fod, Maja J. Matari, and Odest Chadwicke Jenkins. Automated derivation of primitives for movement classification. *Autonomous robots*, 12(1):39–54, 2002.

[Fri01] Jerome H Friedman. Greedy function approximation: a gradient boosting machine. *Annals of Statistics*, pages 1189–1232, 2001.

[GB11] Daniel H Grollman and Aude Billard. Donut as i do: Learning from failed demonstrations. In *Robotics and Automation (ICRA), 2011 IEEE International Conference on*, pages 3804–3809. IEEE, 2011.

[GLO13] Jonathan Grizou, Manuel Lopes, and Pierre-Yves Oudeyer. Robot Learning Simultaneously a Task and How to Interpret Human Instructions. In *IEEE Inter. Conf. on Development and Learning and on Epigenetic Robotics (ICDL-EpiRob)*, Osaka, Japan, 2013.

[Kha99] Roni Khardon. Learning action strategies for planning domains. *Artificial Intelligence*, 113(1):125–148, 1999.

[LMKSV09] Manuel Lopes, Francisco Melo, Ben Kenward, and José Santos-Victor. A computational model of social-learning mechanisms. *Adaptive Behavior*, 467(17), 2009.

[LMM09] Manuel Lopes, Francisco S. Melo, and Luis Montesano. Active learning for reward estimation in inverse reinforcement learning. In *Machine Learning and Knowledge Discovery in Databases (ECML/PKDD'09)*, 2009.

[LMPN15] R. Lioutikov, G. Maeda, J. Peters, and G. Neumann. A parser for constructing movement primitive libraries. In *In preparation*, 2015.

[LT10] T. Lang and M. Toussaint. Planning with noisy probabilistic relational rules. *Journal of Artificial Intelligence Research*, 39(1):1–49, 2010.

[LTK12] Tobias Lang, Marc Toussaint, and Kristian Kersting. Exploration in relational domains for model-based reinforcement learning. *Journal of Machine Learning Research*, 13:3691–3734, 2012.

[ML11] Martin Mason and Manuel Lopes. Robot self-initiative and personalization by learning through repeated interactions. In *6th ACM/IEEE Inter. Conf. on Human-Robot Interaction (HRI'11)*, 2011.

[MMRL15] Yoan Mollard, Thibaut Munzer, Pierre Rouanet, and Manuel Lopes. Learning and representing object assembly tasks. In *under preparation*, 2015.

[MPG+15] Thibaut Munzer, Bilal Piot, Mathieu Geist, Olivier Pietquin, and Manuel Lopes. Inverse reinforcement learning in relational domains. In *submitted to AAAI'15*, 2015.

[MTSS11] Franziska Meier, Evangelos Theodorou, Freek Stulp, and Stefan Schaal. Movement segmentation using a primitive library. In *Intelligent Robots and Systems (IROS), 2011 IEEE/RSJ International Conference on*, pages 3407–3412. IEEE, 2011.

[NCB+13] Scott Niekum, Sachin Chitta, Andrew G. Barto, Bhaskara Marthi, and Sarah Osentoski. Incremental semantically grounded learning from demonstration. In *Robotics: Science and Systems*, volume 9, 2013.

[NJT+11] Sriraam Natarajan, Saket Joshi, Prasad Tadepalli, Kristian Kersting, and Jude Shavlik. Imitation learning in relational domains: A functional-gradient boosting approach. In *Inter. Joint Conf. on Artificial Intelligence (IJCAI)*, pages 1414–1420, 2011.

[NOKB12] Scott Niekum, Sarah Osentoski, George Konidaris, and Andrew G. Barto. Learning and generalization of complex tasks from unstructured demonstrations. In *Intelligent Robots and Systems (IROS), 2012 IEEE/RSJ International Conference on*, pages 5239–5246. IEEE, 2012.

[Sch99] S. Schaal. Is imitation learning the route to humanoid robots. *Trends in Cognitive Sciences*, 3(6):233–242, 1999.

[SD85] Alberto Segre and Gerald DeJong. Explanation-based manipulator learning: Acquisition of planning ability through observation. In *Robotics and Automation. Proceedings. 1985 IEEE International Conference on*, volume 2, pages 555–560. IEEE, 1985.

[SD87] Jude W Shavlik and Gerald DeJong. Bagger: An ebl system that extends and generalizes explanations. In *AAAI*, pages 516–520, 1987.

[SLC11] Jason M Saragih, Simon Lucey, and Jeffrey F Cohn. Deformable model fitting by regularized landmark mean-shift. *International Journal of Computer Vision*, 91(2):200–215, 2011.

[WSJF09] Alan S. Willsky, Erik B. Sudderth, Michael I. Jordan, and Emily B. Fox. Sharing features among dynamical systems with beta processes. In *Advances in Neural Information Processing Systems*, pages 549–557, 2009.

[WTS08] Ben Williams, Marc Toussaint, and Amos J. Storkey. Modelling motion primitives and their timing in biologically executed movements. In *Advances in neural information processing systems*, pages 1609–1616, 2008.

[YFG02] SungWook Yoon, Alan Fern, and Robert Givan. Inductive policy selection for first-order mdps. In *Proceedings of the Eighteenth conference on Uncertainty in artificial intelligence*, pages 568–576. Morgan Kaufmann Publishers Inc., 2002.

2 Annexes

- Learning and Representing Object Assembly Tasks In this report we present a framework for learning assembly tasks from demonstration. We show how a complex assembly task can be automatically decomposed in components allowing to learn a set of constraints between different objects. This information allows to create a high-level representation to plan manipulation sequences. We introduce also a graphical user interface that allows the user to see the sequence of actions and the relations between objects that were learned by the robot. The user can thus program the robot by combining demonstrations with corrections in the GUI minimizing the overall programming phase.

- Inverse Reinforcement Learning in Relational Domains In this work, we introduce the first approach to the Inverse Reinforcement Learning (IRL) problem in relational domains. IRL has been used to recover a more compact representation of the expert policy leading to better generalize among different contexts. Relational learning allows one to represent problems with a varying number of objects (potentially infinite), thus providing more generalizable representations of problems and skills. We show how these different formalisms can be combined by modifying an IRL algorithm (Cascaded Supervised IRL) such that it handles relational domains. Our results indicate that we can recover rewards from expert data using only partial knowledge about the dynamics of the environment. We evaluate our algorithm in several tasks and study the impact of several experimental conditions such as: the number of demonstrations, knowledge about the dynamics, transfer among varying dimensions of a problem, and changing dynamics.

- Temporal Segmentation of Pair-Wise Interaction Phases in Sequential Manipulation Demonstrations We consider how to learn a representation from bimanual assembly tasks from demonstration. We propose to analyze the demonstration in terms of the (potentially concurrent) interaction phases between any pair of involved bodies (hands, tools, objects). These interaction phases are the key to extract a more abstract description of the demonstration. In particular one may assume that the goal of each interaction phase is to achieve a certain geometric constraint. This gen- eralized previous approaches on LfD to consider not just the motion of the end-effector but also the relational properties of the motion of the objects. We present an approach to train a Conditional Random Field to detect the pair-wise interaction phases and based on this labeling analyze the geometric constraints that are established. In this way we extract a higher level task oriented description of the demonstrated sequential manipulation. We test our system using data from a person assembling a toolbox of 5 parts using a screwdriver and two hands. We consider how to learn a representation from bimanual assembly tasks from demonstration.

- A Parser for Constructing Movement Primitive Libraries Movement primitives are a well established approach for encoding and executing movements. While the primitives themselves have been extensively researched, the concept of movement primitive libraries has not received as much attention. The goal of this work is to learn a primitive library from unsegmented demonstrations. The learned library is used to parse previously unseen demonstration into a sequence of recurring primitives. At the same time the library provides a set of primitives which can be sequenced in order to solve previously undemonstrated

tasks. Current approaches separate the segmentation of the demonstration from the learning of the primitives. This separation neglects the dependency between the found segments and the learned primitives. However this dependency is crucial for a good segmentation and the learning of the underlying primitives. Therefore we propose a novel method which, in contrast to previous work, takes advantage of the dependency. Based on probabilistic inference our approach is able to learn the segmentation and the primitives simultaneously. We compare our work to state-of-art methods and show the advantages of such a combined approach. Experiments on a complex bi-manual robot platform demonstrate the applicability of our method in real world robot tasks.

in preparation

Learning and Representing Object Assembly Tasks

Yoan Mollard, Thibaut Munzer, Pierre Rouanet and Manuel Lopes
Inria Bordeaux Sud-Ouest
manuel.lopes@inria.fr

October 21, 2014

Abstract

In this report we present a framework for learning assembly tasks from demonstration. We show how a complex assembly task can be automatically decomposed in components allowing to learn a set of constraints between different objects. This information is used to create a high-level plan of the manipulation sequence. We introduce also a graphical user interface that allows the user to see the sequence of actions and the relations between objects that were learned by the robot. The user can thus program the robot by combining demonstrations with corrections in the GUI minimizing the overall programming phase. We show preliminary results of interpreting assembly tasks of furniture.

1 Introduction

The applications for robots are changing. New societal and economical challenges demand new robotic systems that are more adaptable to different environments and tasks, and easier to use. Robots are starting to be used at home, and in flexible industrial cells, both applications that demand a constant change between tasks and an easy way to instruct and personalize the behavior of the robotic system. A major challenge is to provide intuitive and easy ways so that a larger part of the population can instruct and work with those robotic systems. We can thus create a list of desired properties of a system that can be instructed and executes joint-tasks in a friendly way:
easy to program where a simple demonstration of a task execution must allow, in most cases, to program it;
easy to instruct where corrections of what has been learned and, at execution time, deviations from the standard path must be easy to provide;
shared task awareness where a joint comprehension of the task being executed, the world state and the role of each partner, must be shared;
shared initiative where the robot does not require a step-by-step instruction that renders the interaction very tiring, the system must be able to start its own behavior and wait when it is uncertain about the task, allows considering that the user might want to correct or change the behavior at any point in time;
hierarchical learning where all parts of the task are easy to program. Even if assuming that the robotic system is already equipped with many skills, at all levels there will be limitations that must be recoverable at runtime.

We start by discussing what we can learn from a demonstration considering the problem of furniture assembly. An easy way to program a task is just to execute it. From this information the system extracts keyframes that represent the moments where new constraints are created between objects. This segmentation of the demonstration allows to separate the motor primitives into clusters, and also to create a semi-symbolic representation of the task. A plan can also be learned from the semi-symbolic representation using logic based approaches. Unfortunately, even if there are no mistakes in the demonstrations, the system inference of all those components will be imperfect. For this we create a graphical interface where the user is able to see all the information that the system learned: sequence of manipulation steps, constraints, motor primitives and others. The user is then able to correct some of this information and to visualize an execution of the task using the Baxter robot simulator.

In this report we present our ongoing work and general perspective on the interaction modalities of the project. It will be completed in the future with with methods from other workpackages, and with further developments.

2 Related Work

Learning from demonstration algorithms [21, 4] have focused on limited forms of demonstrations, in most cases just working with a specific type of demonstration. Recently, some approaches started to consider a larger variety of demonstrations that go beyond trajectories acquired by means of optical tracking or kinesthetic teaching. This forms include Keypoints [1, 2], learning from failures [11], active approaches [14, 7], and even the use of loosely speci-

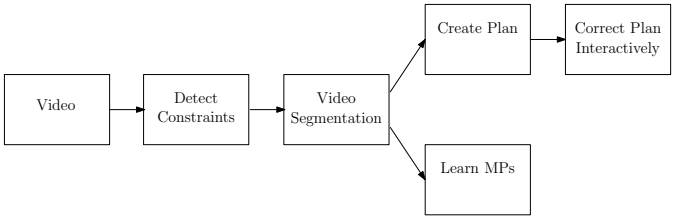

Figure 1: Workflow for learning a hierarchical task representation from demonstration.

fied protocols [10].

New approaches start to consider how to improve the knowledge acquired from a demonstration through different processes: training the user to provide better demonstrations [6], providing more demonstrations, allowing the system to request specific demonstrations or clarifications. The last option has several advantages: theoretically it allows faster learning [15], it provides information to the user about what the system understands and potentially increases the trust we can have in the system.

In many cases we see a trend of unifying the training and execution part where the instructor is free to interrupt the system when a correction is needed [16], and also allows the robotic system to request further instruction when necessary [7].

Learning hierarchical representations of tasks, in the robotic, domain would require learning high-level task plans and also low-level motor controllers. Works have mostly consider such levels separately even if several approaches already consider the execution of such hierarchical systems. A notable exception is [19, 18] that presented an integrated approach to segment manipulation demonstrations into actions, each represented by a Dynamic Motion Primitive (DMP). Low-level Motor primitives represented using DMPs can also be stored in a library and then called by a high-level planner[20].

3 Interpreting a Demonstration

Many tasks can be represented as a sequence of constraints that are established between different, or parts of-, objects. For instance the assembly of a chair can be seen as attaching four legs and a back to a seating. Arranging the dishes on a tray is another example where a set of constraints is established between the different objects.

The global workflow is presented on Figure 1: these constraints are computed from the videos that allow to segment a whole demonstration in shorter motions. Each of those motions creates a new constraint. From this, we can directly learn motor primitives or merge it with other videos to extract a general assembly plan. This plan can then be corrected in cooperation with the user.

3.1 Detecting Operations on Objects

According to our previous assumptions, a task is the result of a sequence of actions that establish constraints between objects. For a set of rigid objects, and commonly available parts, we can have a very precise definition of what a constraint is. It is a connexion that enforces a *kinematic constraint* between two parts. Standard constraints include: rigid, prismatic or rotational. Each one can be represented by a specif constraint between object parts:

Fixed constraint: $f(x_1, x_2; C_f) = \begin{cases} x_1 - x_2 = C \\ \dot{x_1} - \dot{x_2} = 0 \end{cases}$

Prismatic constraint: $f(x_1, x_2; C_p) = \begin{cases} x_1 - x_2 = Ct \\ \dot{x_1} - \dot{x_2} = C \end{cases}$

Unconstrained: $f(x_1, x_2; C_u) = \begin{cases} x_1 - x_2 = \epsilon \\ \dot{x_1} - \dot{x_2} = \epsilon \end{cases}$

Where x_1 and x_2 are the posture of each object, C is a constant, t is time and ϵ is a random variable of zero mean and high variance. To detect when a constraint has been established we will rely on the graphical model of Figure 2. The observation model is given by each of the previous constraint's models. In this model we rely on the motion of at least one of the objects to observe if the constraint is valid or not. A classic expectation-maximization algorithm can be used to learn the parameters of the model.

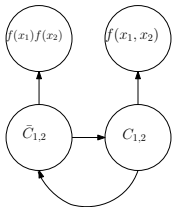

Figure 2: Constraint Detection

The formalism presented in [5] will be used in the future.

3.2 Extract Motor Primitives

The previous step allows to extract the frames in the assembly plan that result in the establishment of a constraint between parts. If a robot is highly capable it could be able to plan actions to establish such constraints, but in most cases new parts will require different motor primitives that can be learned from the information already contained in the demonstration. Taking into account the classes of objects, a single demonstration can already include more than one example of a single motor primitive, e.g. inserting a leg in the seating occurs 4 times in a single demonstration.

Other methods aim at segmenting and learning motor primitives just at the signal level, see for instance [8]. But, in many situations extra information already exists in the environment that can be exploited. Figure 3 shows the segmented trajectories on a single demonstration on a single dimension. Highlighted parts are the time between the last

detected constraint time and the time of the detected constraint.

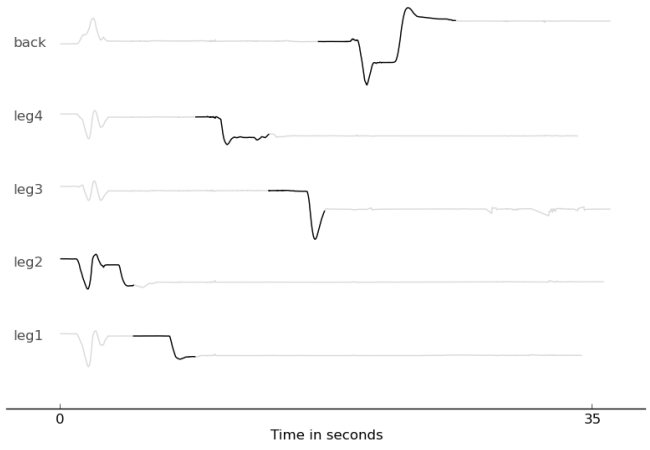

Figure 3: Segmented trajectories (relative height) with respect to the seating during a single demonstration

After having these segments of data, Dynamical Movement Primitives (DMPs) can be used to learn the segmented trajectories [12].

3.3 Learning the Motor Primitives

Our goal is to learn a generalized trajectory for each constraint given the segmented trajectory at each demonstration. The first step is to average the segmented trajectories to get a single trajectory on which we can learn the DMPs. As our trajectory is given in Cartesian space (x, y, z) for the position and in quaternion space (q_0, q_1, q_2, q_3) for the rotation, seven DMPs will be necessary to learn the complete trajectory. However, it is necessary to ensure that the unit length of the quaternion q is kept, thus a post-normalization step is required[20].

Following the segmented trajectories using DMPs gives us the possibility to change the position of the starting point and still bring the object toward the desired constraint position as seen in Figure 4. This add some flexibility in the system as an object can be picked at an arbitrary position.

4 Planning Actions

In order to learn and then execute the assembly plan, we propose the use of high-level reasoning relying on relational representations and leveraging new algorithmic developments.

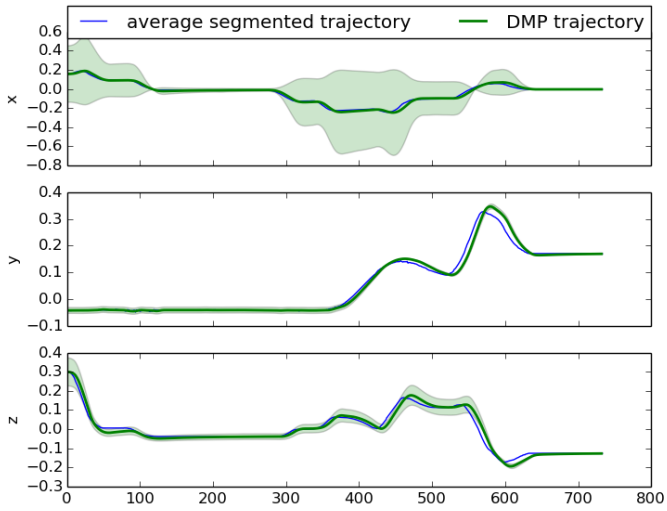

Figure 4: Example of a range of possible starting positions for the DMPs

4.1 Symbol Learning

Before being able to learn abstract plans of a task, it is necessary to find an abstract representation of the context where those actions are executed. Requiring symbols to be pre-defined would rather limit the usage to a particular system. During a demonstration, a robotic system should be able to acquire new symbols allowing it to represent new tasks. Those symbols can be the labeling of key feature of a state [13], or even the meaning of instruction signals [9].

For this case study, we propose to use only kinematic constraints. We can thus use a generic 3-ary relation "is_constraint(O1, O2, C)" where O1 and O2 are two objects and C is a constraint symbol. The constraint symbols are computed from extracted constraints that are clustered. In a similar fashion we propose to have 3-ary relational actions "action(O1, O2, C)". Applying such an action will produce the corresponding constraint.

4.2 Create an Abstract Representation of the Assembly Plan

Once symbols are known, one can deduce relational MDP trajectories from raw data. Decision points are placed each time a constraint C is detected. The starting state s_0 is the empty set and is modified in the following way: if the constraint "is_constraint(O1, O2, C)" is present from time t (counted as the number of past decision point) $s_{i+1} = s_t \cup \{is\_constraint(O1, O2, C)\}$ and $a_t = action(O1, O2, C)$.

A relational policy learning algorithm like BLMRIL [17] can then be used to learn the policy $\pi \in S \times A$.

4.3 Planning and Executing Hierarchical Plans

Two options are available to execute the previously learned actions: offline or online planning.

The offline approach computes a complete plan before executing it, it does not allow a shared task between a human and the robot. It is simpler in the sense that it does not require to detect actions that have already been undertaken by somebody else. By providing a model to the system we can determine the sequence of actions that have to be followed. In this case the model would be a single rule $action(O1, O2, C) : s_t \rightarrow s_t \cup \{is\_constraint(O1, O2, C)\}$.

On the contrary using online planning computes the next action according to the current scene state. It does require constraints detection in real time but endows the system with error recovery capabilities as it is able to re-plan. If one can detect constraints it is straightforward to follow the previously computed policy. In this setup a model is not compulsory.

We illustrated the offline approach with two datasets: an assembly of a chair and an assembly of a bench. Here is an example of the plan we learn from the chair:

- $action(seating, leg2, c2)$
- $action(seating, leg1, c3)$
- $action(seating, leg4, c1)$
- $action(seating, leg3, c5)$
- $action(seating, back, c4)$

Where c* symbols represents the clustered constraints.

5 Context Awareness

When several people work together sharing a task, communication between them is crucial. In human teams, people ensure this communication through team meetings, reports and more generally through their verbal and nonverbal communication. All of these must be part of an information system that make people aware about what the task and the activities of others. Moreover, in case things are not clear, people naturally ask questions to clarify the situation.

In the field of machine learning, clarifying questions have already been studied in active learning [14, 7, 15]. But, studying awareness in a context of learning from demonstration, has been little researched [3]. In this context a human and a robot working together on the same task should be aware of what the other coworker knows and is working on. Especially, after teaching a new task to a robot it is important that the humans know what the robot has understood about it and that they are able to correct this knowledge if necessary. This is a question of representing training data and correcting them either manually from coworkers or with active learning.

To achieve this goal we propose a Graphical User Interface (GUI) able to represent these constraints and to ask users for clarification and correction. This allows to correct a wrong interpretation of the task from the learning algorithms but also enables knowledge transfer. By assuming that some knowledge has been learned for the task "assembly of a chair", this can be reused to assemble a bench after small corrections provided by the user through the GUI since the scale and the parts – a bench has a seating and legs but no back – are slightly different.

5.1 Constraint representation and correction

In this GUI we first represent the fixed constraints, i.e. the learned coordinates of each object with respect to its frame of reference. To achieve this, a 3D view of the parts shows them with the pose defined by the fixed constraints. The user can browse into the different objects, see their parent and consult the values of the constraints. A specific color is attributed to the selected object (in orange, e.g. the back) and another one for its parent (blue, e.g. the seating). This color scheme is kept whatever the current selection is, and names of both objects are recalled with this color as well.

Additionally to the fixed constraints, we represent linear constraints using thick lines along the three axes of the constrained object. And we also represent angular constraints using arrows around fixed axes (roll, pitch and yaw). An example is presented on Figure 8 including zero angles, a full rotational joint and lower angles being noise in this case.

The GUI can ask the user for constraint clarification by animating sequentially the 6 possible degrees of freedom. This visual animation helps the user to focus on the wrong degrees of freedom needing corrections.

The user can provide corrections through the graphical widgets for rough adjustment or directly raw values he would have measured with a tape-measure. A right-click menu triggers a menu to correct parent (framing) or trigger animations. Once editing is finished the modified constraints are sent back to the system for execution.

5.2 Assembly plan representation

The assembly plan is represented thanks to keyframes. We call keyframe each time t a new constraint is created. A snapshot of all the objects – constrained or still unconstrained – represent a stage in the assembly plan. The user can view a list of ordered keyframes, which correspond to the assembly plan, example in Figure 5.

[1] translation is set to a default value of $\pm 1m$

[2] user is then asked along or around which axis x, y or z

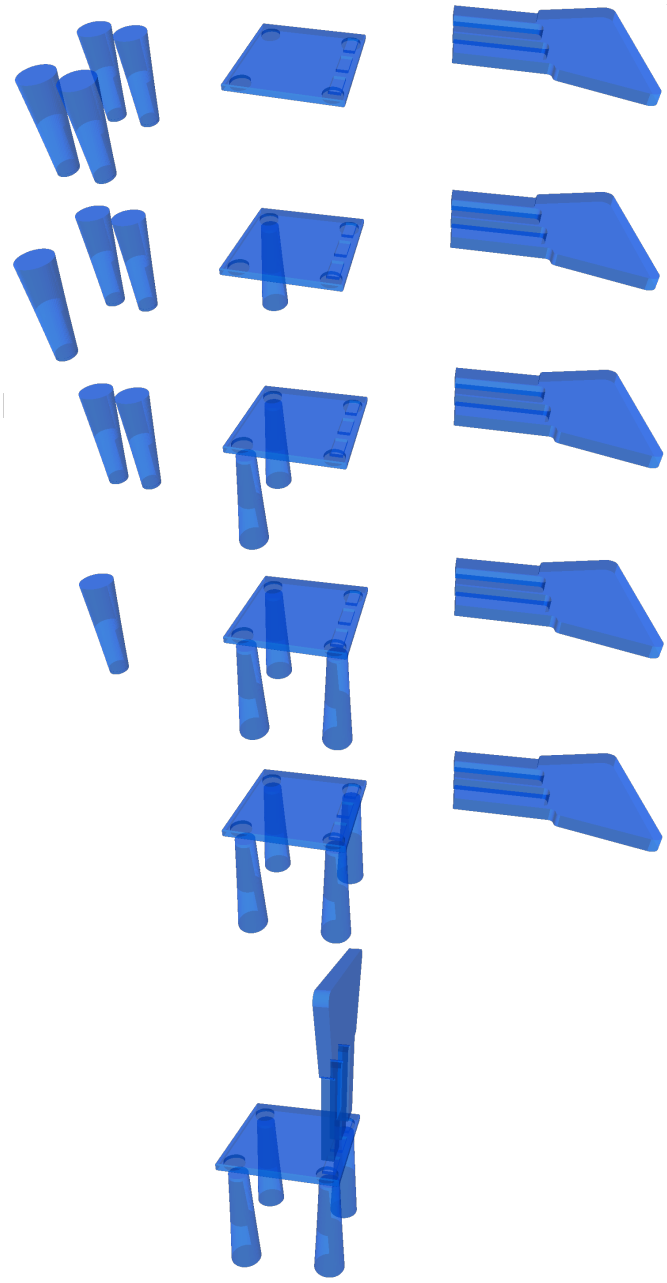

Figure 5: Sequence of keyframes shown to the user after learning the plan and the constraints

6 Experiments

6.1 Setup

Our experimental setup can be splitted in three phases: data recording, asking for clarification in the GUI, and execution. Similar to the data presented in Figure 6, we recorded 22 demonstrations of users taking different assembly and disassembly tasks using an Optitrack tracking system with four infrared cameras (Flex 13: 1280x1024 resolution at 120Hz) placed at the upper edges of the working space looking top-down. Reflective markers were placed so that we tracked each part of the whole object independently from each other. This system provided in output the absolute pose of all parts at 120Hz during the whole demonstration and also a video of the user and his working space recorded thanks to a webcam.

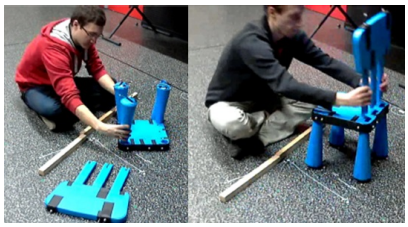

Figure 6: Recorded chair assembling

For the second phase, a 3D model and constraints learned from the previous demonstrations were loaded into the GUI. The 3D model could be either the same object assembled during demonstrations or a different one in case we wanted to perform task transfer. We used simplified 3D models (e.g. cylinders for legs while real legs have a more complex shape). A user was provided a screen, a keyboard, a space mouse to facilitate 3D navigation and a regular mouse to click on dialog boxes. He was then asked to assemble in simulation the object shown in the GUI by using available features to visualize, modify and animate the constraints. Figure 8 represents this step of correction of the wrong fixed constraints and degrees of freedom. The assembly plan is created and sent to the execution system in an offline mode.

Execution of the motions was ensured by Moveit software for motion planning and environment simulation, and by a 1-DoF electric gripper of a Baxter robot simulated in Gazebo for control. To simplify the process we ignored collisions between the fingers of the robot and the picked objects. We also did not simulate the grasp itself, so a grasp request (resp. a place request) did not actually close (resp. open) the gripper and systematically ended up to an object successfully attached to (resp. detached from) the gripper. The execution system was given an assembly plan as an input and was in charge of planning collision-free trajectories to pick and assemble objects in the order and with the constraints specified in the plan. In this context a constraint in an assembly task can be seen as the final pose of the gripper

in a pick-and-place task. In case the motion planner fails to assemble two objects, it automatically computed a random variation in the assembly pose according to what degrees of freedom are authorized in the constraints. If the system would not have been told that some degrees of freedom exist, that could cause the motion planning to fail definitely because the final pose of the gripper is unreachable or generates collisions.

6.2 Tasks

Our first task was the assembly of a chair for children composed of six parts: a seating, a back and four identical legs that can be inserted in any leg hole of the seating. A video showing the whole workflow can be found online [3].

Our second task was the assembly of a simpler bench for children composed of five parts: a seating and four identical legs. Assuming a reference coordinate system at the barycenter of the seating, both chair and bench have not the same constraints: the bench is smaller so the legs are closer to the center but they have the same relative orientations. Figure 7 represents the bench after assembly. We can observe that the final poses of the legs are not aligned, the robot having considered the possible degree of freedom of the legs around themselves in order to create feasible trajectories.

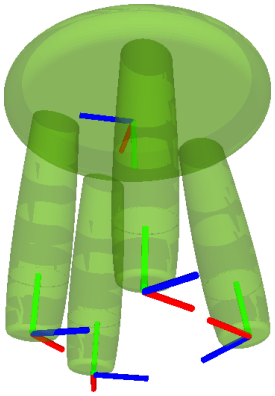

Figure 7: Bench and its degrees of freedom after assembly by the robot in simulation

Our last task illustrated task transfer: we learned constraints and plan from the demonstrations of the blue chair, asked user to adapt them in order to assemble the bench, and executed the assembly in simulation. A video can be found online [4].

[3] http://vimeo.com/109165299
[4] http://vimeo.com/109165300

6.3 Results

We measure the efficiency of the whole workflow in several ways: by counting the number of parameters (i.e. parent + relative pose) that the user needs to modify before getting a satisfying result with the displayed constraints; the precision of the detected constraints; and the quality of the plan that was found.

For the case of the chair, we can recover the correct constraints with and error of $.8cm \pm .1cm^2$ and an error in orientation of $4.7deg \pm .6deg^2$. For the bench the results are $2cm \pm .6cm^2$ of error in position and $14deg \pm 13deg^2$ of error in orientation.

In table 1 we show for each task the average number of parameters to be corrected manually compared to the total we should set to define the constraints from scratch with no demonstrations. By using demonstrations we reduce of 50.85%the number of parameters that the user needs to provide compared to a manual programming from zero.

Table 1: Number of parameters to be corrected from demonstrations compared to no demonstration

task	error pos	error orient	corrections
chair	$.8cm \pm .1cm^2$	$4.7deg \pm .6deg^2$	15.25/30
bench	$2cm \pm .6cm^2$	$14deg \pm 13deg^2$	14.25/28

Despite a fair numerical precision for a single object, a significant visual error is noticeable in the GUI, that is most explained by our use of meshes simpler than the real objects and our difficulty to align the frames of the tracked objects (whose real shape in quite complex) with their meshes using the tracking system.

To evaluate the quality of the assembly plan, we compute how many times a correct plan is generated. A plan is considered correct if all parts have been attached to the *seating* in different slots. The evaluation have been conducted for a number of demonstrations varying from 1 to 4. Because the algorithm is stochastic, each experiment is run 96 times and the results are averaged (when possible, different demonstrations are used). The results are summed up in Table 2. We observe that increasing the number of demonstrations allows to learn the plan more robustly.

Table 2: Plan quality in terms of success.

number of demonstrations	1	2	3	4
success rate	0.52	0.75	0.75	0.88

7 Conclusions and Ongoing Work

We proposed a way to program a robot by demonstration that considers automatic decomposition of subparts of the

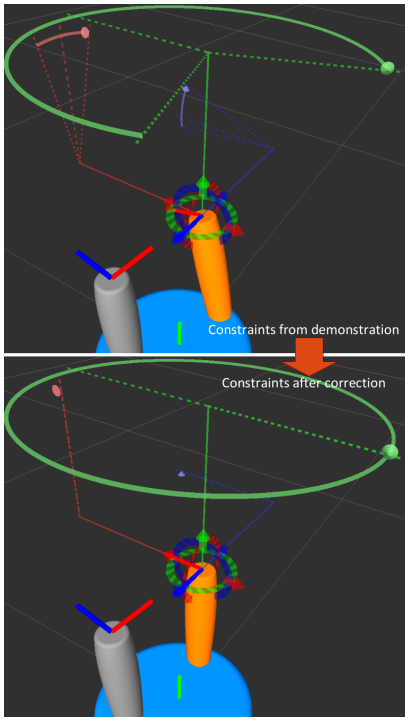

Figure 8: Correction of wrong constraints in the interface: The leg is tilted around z; as for degrees of freedom shown by arrows, noise around x and z are cleaned, and rotation around y is increased to 2π

demonstration, learning of hierarchical plans and an interface to improve mutual awareness and easy error correction. We represented object assemblies in the form of different types of constraints between different objects. We showed that we could exploit non-rigid constraints to execution by allowing more solutions to the planing problem. Then we illustrated our approach with simulated assemblies of a chair and a bench whose assembling plan and constraints have been extracted from demonstrations. In these simulations we controlled the robot by trajectory planning but an extraction of motor primitives from the demonstrations is also possible.

We also illustrated the notion of task transfer by assembling a bench thanks to the demonstrations of the assembly of a chair, the user being asked to adapt the constraints since they are slightly different for these two objects. Despite the limitations of our work we showed that we could simplify a lot assembly tasks by combining both demonstrations and user corrections.

Concerning the GUI itself, it could also be used to modify the assembly plan by drag-and-drop in the keyframes list and edit the trajectories through editable key points when we use motor primitives instead of planned trajectories. As another extension we could tap into all other motions of the demonstrations that do not end up to a new constraint but are key motions in the assembly, for instance returning the seating to assemble the back or moving objects closer to the robot in case the assembly still leads to an unfeasible trajectory. These key motions are often the same over all demonstrations, and we should be able to detect them when demonstrations are performed by a single user but also when we record human-human cooperation, in order to assemble more complex objects. Detecting these key motions will be a compulsory step before executing the assembly tabletop with a real Baxter robot.

We should also benefit of all available arms of our robot to speed up and complexity the tasks that we are able to perform by executing some actions in parallel. Executing a disassembly for instance will require both arms and synchronized parallel motions.

References

[1] B. Akgun, M. Cakmak, K. Jiang, and A.L. Thomaz. Keyframe-based learning from demonstration. *Inter. Journal of Social Robotics*, pages 1–13, 2012.

[2] B. Akgun, K. Jiang, M. Cakmak, and A. Thomaz. Learning tasks and skills together from a human teacher. *AAAI Conf. on Artificial Intelligence*, pages 1868–1869, 2011.

[3] Sonya Alexandrova, Maya Cakmak, Kaijen Hsiao, and Leila Takayama. Robot programming by demonstra-

tion with interactive action visualizations. In *Proceedings of Robotics: Science and Systems*, Berkeley, USA, July 2014.

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

[5] Andrea Baisero, Yoan Mollard, Manuel Lopes, Marc Toussaint, and Ingo Lutkebohle. Temporal segmentation of pair-wise interaction phases in sequential manipulation demonstrations. In *Submitted to: Proceedings of 2015 IEEE International Conference on Robotics and Automation (ICRA)*, 2015.

[6] Maya Cakmak and M Lopes. Algorithmic and Human Teaching of Sequential Decision Tasks. *AAAI*, pages 1536–1542, 2012.

[7] S. Chernova and M. Veloso. Interactive policy learning through confidence-based autonomy. *J. Artificial Intelligence Research*, 34:1–25, 2009.

[8] S. Chiappa and J. Peters. Movement extraction by detecting dynamics switches and repetitions. *Advances in neural information processing systems*, 23:388–396, 2010.

[9] Jonathan Grizou, Inaki Iturrate, Luis Montesano, Pierre-Yves Oudeyer, and Manuel Lopes. Interactive learning from unlabeled instructions. *Conf. on Uncertainty in Artificial Intelligence (UAI'14)*, pages 1–8, 2014.

[10] Jonathan Grizou, Manuel Lopes, and Pierre-Yves Oudeyer. Robot Learning Simultaneously a Task and How to Interpret Human Instructions. In *IEEE Inter. Conf. on Development and Learning and on Epigenetic Robotics (ICDL-EpiRob)*, Osaka, Japan, 2013.

[11] Daniel H Grollman and Aude Billard. Donut as i do: Learning from failed demonstrations. In *Robotics and Automation (ICRA), 2011 IEEE International Conference on*, pages 3804–3809. IEEE, 2011.

[12] Auke Jan Ijspeert, Jun Nakanishi, Heiko Hoffmann, Peter Pastor, and Stefan Schaal. Dynamical movement primitives: learning attractor models for motor behaviors. *Neural computation*, 25(2):328–73, February 2013.

[13] Johannes Kulick, Marc Toussaint, Tobias Lang, and Manuel Lopes. Active learning for teaching a robot grounded relational symbols. In *Inter. Joint Conf. on Artificial Intelligence (IJCAI'13)*, Beijing, China, 2013.

[14] Manuel Lopes, Francisco S. Melo, and Luis Montesano. Active learning for reward estimation in inverse reinforcement learning. In *Machine Learning and Knowledge Discovery in Databases (ECML/PKDD'09)*, 2009.

[15] David Martinez, Guillem Alenya, Pablo Jimenez, Carme Torras, Jurgen Rossmann, Nils Wantia, Eren Erdal Aksoy, Simon Haller, and Justus Piater. Active learning of manipulation sequences. *2014 IEEE International Conference on Robotics and Automation (ICRA)*, pages 5671–5678, May 2014.

[16] Martin Mason and Manuel Lopes. Robot self-initiative and personalization by learning through repeated interactions. In *6th ACM/IEEE Inter. Conf. on Human-Robot Interaction (HRI'11)*, 2011.

[17] Thibaut Munzer, Bilal Piot, Mathieu Geist, Olivier Pietquin, and Manuel Lopes. Inverse reinforcement learning in relational domains. In *submitted to AAAI'15*, 2015.

[18] Scott Niekum and Sachin Chitta. Incremental Semantically Grounded Learning from Demonstration. *Robotics: Science and Systems*, 2013.

[19] Scott Niekum and Sarah Osentoski. Learning and generalization of complex tasks from unstructured demonstrations. *Intelligent Robots and Systems*, 2012.

[20] Peter Pastor, Heiko Hoffmann, Tamim Asfour, and Stefan Schaal. Learning and generalization of motor skills by learning from demonstration. *2009 IEEE International Conference on Robotics and Automation*, pages 763–768, May 2009.

[21] S. Schaal. Is imitation learning the route to humanoid robots. *Trends in Cognitive Sciences*, 3(6):233–242, 1999.

Inverse Reinforcement Learning in Relational Domains

Thibaut Munzer
Inria, France
thibaut.munzer@inria.fr

Bilal Piot
Supelec

Mathieu Geist
Supelec

Olivier Pietquin
Univ. Lille

Manuel Lopes
Inria, France
manuel.lopes@inria.fr

Abstract

In this work, we introduce the first approach to the Inverse Reinforcement Learning (IRL) problem in relational domains. IRL has been used to recover a more compact representation of the expert policy leading to better generalize among different contexts. Relational learning allows one to represent problems with a varying number of objects (potentially infinite), thus providing more generalizable representations of problems and skills. We show how these different formalisms can be combined by modifying an IRL algorithm (Cascaded Supervised IRL) such that it handles relational domains. Our results indicate that we can recover rewards from expert data using only partial knowledge about the dynamics of the environment. We evaluate our algorithm in several tasks and study the impact of several experimental conditions such as: the number of demonstrations, knowledge about the dynamics, transfer among varying dimensions of a problem, and changing dynamics.

Introduction

Learning policies from observations is an intuitive way to learn complex skills (Schaal, 1999; Argall, Chernova, and Veloso, 2009; Khardon, 1999). When learning from observing another agent, we can aim at learning directly the behavior or, instead, the criteria behind such behavior. The former approach is usually called *Imitation Learning* (IL) while the latter is called *Inverse Reinforcement Learning* (IRL). The choice of one of the two hypothesis might be motivated by the application, the search for an explicit explanation for the behavior, the representation compactnesses or the capability of generalization among different problems. These two mechanisms can be used to model and explain many different social learning behaviors in animals (Lopes et al., 2009).

If an explanation is found for the behavior, as a reward function for instance, it is expected that the agent can generate the correct behavior in different situations, as it can plan its actions in those new situations. As a drawback, such approaches require some knowledge on both the original and the new domain. Another difference between IL and IRL is the compactnesses of the representation. Even in the same domain, there are tasks that are more compactly described as a reward function while others can be better described as a policy. For instance, in a typical blocks world domain, the task of making a tower with all objects requires a non-trivial policy but can be described with a simple reward, see Džeroski, De Raedt, and Driessens (2001).

Another approach for generalizing to different problems is to use representations that are more expressive. The use of relational learning (also known as logical learning) allows one to generalize between worlds with different numbers of objects and agents. Learning to act from demonstration in relational domains have been a research prolblem for a long time (Segre and DeJong, 1985; Shavlik and DeJong, 1987; Khardon, 1999; Yoon, Fern, and Givan, 2002). The use of relational representations is attracting again more attention due to new algorithmic developments, new problems that are inherently relational and the possibility of learning the representations from real-world data, including robotic domains (Lang and Toussaint, 2010; Lang, Toussaint, and Kersting, 2012).

Even though no approach to relational IRL has been proposed, this work is close to TBRIL (Natarajan et al., 2011). The authors propose the use of gradient-tree boosting (Friedman, 2001) to achieve imitation learning in relational domains.

In this work, we generalize a previous approach for IRL, namely Cascaded Supervised IRL (CSI) (Klein et al., 2013), to handle relational representation. The first section presents the Markov Decision Processes (MDP) framework and its extension for relational data. A second section reviews IRL in general and presents the CSI algorithm. We then generalize this method to the relational domain in the third section. Finally, we present our results and the conclusions in the two last sections.

Relational Learning and Markov Decision Processes

This section introduces the notation and the formalism for MDPs and its extension to the relational domain.

Markov Decision Process

The proposed approach for learning by demonstrations relies on the MDP framework. MDP models the interactions of an agent evolving in a dynamic environment. It can be represented by a tuple $M_R = \{S, A, R, P, \gamma\}$ where $S = \{s_i\}_{1 \leq i \leq N_S}$ is the state space, $A = \{a_i\}_{1 \leq i \leq N_A}$ is the action space, $R \in \mathbb{R}^{S \times A}$ is the reward function, $\gamma \in]0, 1[$ is a discount factor and $P \in \Delta_S^{S \times A}$ is the Markovian dynamics which gives the probability, $P(s'|s, a)$, to reach s' by choosing the action a in the state s. A policy π maps each state to an action, so it is an element of A^S and defines the behavior of an agent.

The quality function $Q_R^\pi \in \mathbb{R}^{S \times A}$ for a given state-action couple (s, a) is defined as the cumulative discounted reward for starting in state s by doing the action a and following the policy π afterwards. More formally, $Q_R^\pi(s, a) = \mathbb{E}_{s,a}^\pi[\sum_{t=0}^{+\infty} \gamma^t R(s_t, a_t)]$, where $\mathbb{E}_{s,a}^\pi$ is the expectation over the distribution of the admissible trajectories $(s_0, a_0, s_1, \pi(s_1), \dots)$ obtained by executing the policy π starting from $s_0 = s$ and $a_0 = a$. Moreover, the function $Q_R^* \in \mathbb{R}^{S \times A}$ defined as: $Q_R^* = \max_{\pi \in A^S} Q_R^\pi$ is called the optimal quality function (the optimal policy π^* is greedy resp. to it).

Relational MDP

One can derive a relational MDP that generalizes the reinforcement learning formalism for high level representations (Džeroski, De Raedt, and Driessens, 2001). Solutions to the planning and learning problems have already been developed (Kersting, Otterlo, and De Raedt, 2004; Lang and Toussaint, 2010). Relational representations generalize the commonly used representations in MDPs and machine learning. Instead of representing data as *attribute-value* pairs, it relies on first order logical formulas. Given all objects and a set of possible relations (or predicates), data is represented by the set of logical rules that are true under a particular assignment (grounding).

Under this representation, the state of the environment is the set of predicates that are true for the current grounding of the formulas. The state of the environment can change when (relational) actions are applied. Actions have a different semantics than in finite MDPs. A relational action is abstract and so its application depends on choosing an object to apply it, *i.e.* grounding the action. For instance, the action *take(object)* can be applied to different objects: *take(cup)* or *take(box)*, and so leading to different effects. As in classical systems such as STRIPS rules, actions are defined as preconditions c, that must be fulfilled so that the action can be applied. More formally, an abstract action a is available, in a given state s if there exists a grounding σ such that $c\sigma \in s$, e.g. for that particular grounding of the action the precondition is true in that state. Upon applying such action, the state evolves to another state. This is defined as outcomes o. Formally, if there is a substitution σ such that $c_i\sigma \in s_t$ then applying $a\sigma$ in s_t will lead to $s_{t+1} = (s_t \setminus c_i\sigma) \cup o_i\sigma$ with $(c_i, o_i) \in \mathcal{R}_a$, the set of rules of action a.

Example Domain : Blocks world

We present a simple example domain that is going to be used to validate our approach. This is a classical domain that has several interesting and challenging characteristics such as possibility of defining different tasks, easy to visualize, possibility of defining problems with a changing number of objects. Being a relational domain the blocks world can be defined by a set of objects, a set of predicates and a set of actions. The objects are the blocks and $floor$, a surface where the blocks lies. There are six predicates: $on(X, Y)$ (is true if object X is on object Y), $clear(X)$ (is true if X is a cube and no object lies on him), $cube(X)$, $floor(X)$, $red(X)$ and $blue(X)$.

The blocks world offers two abstract actions $move(X, Y)$ and $wait()$. The $wait()$ abstract action is available in every state and does not modify the state. The $move(X, Y)$ abstract action is defined with three rules that describe when the actions can be applied and what becomes the next state: e.g. can only be applied when the object that is to be grasped has no object on top and the object where we will put it also has no object on top.

We can define multiple tasks in this domain but, to simplify discussion, we will discuss only two rewards: the first, called *stack*, is 1 when all blocks are stacked and 0 otherwise, the second one, called *unstack*, is 1 when all blocks are on the floor and 0 otherwise.

IRL and CSI

In the learning from observations paradigm, an apprentice tries to learn a behavior from demonstrations of an expert agent. An interesting way to address the problem, using MDPs is the IRL framework (Ng, Russell, and others, 2000; Russell, 1998). IRL is a method that tries to find a reward function $\hat{R}$ that could explain the expert policy π_E. More formally, an IRL algorithm receives as inputs a set D_E of expert sampled transitions $D_E = (s_k, a_k, s'_k)_{1 \leq k \leq N_E}$ where $s_k \in S$, $a_k \sim \pi_E(.|s_k)$, and $s'_k \sim P(.|s_k, a_k)$ and a set of non expert sampled transitions $D_{NE} = (s_l, a_l, s'_l)_{1 \leq l \leq N_{NE}}$ where $s_l \in S$, $a_l \in A$, and $s'_l \sim P(.|s_l, a_l)$. Then the algorithm outputs a reward $\hat{R}$ such that any optimal policy $\pi^*_{\hat{R}}$ with respect to $\hat{R}$ is also optimal with respect to the unknown reward R or at least as good as the expert policy with respect to R. In order to realize that, one can search for a reward such that the expert policy is optimal for this reward. But, as for the null reward every policy is optimal, we have to use a bias. For exemple, a reward for which only the expert actions are optimal.

Most of IRL algorithms (see the work of Neu and Szepesvári (2009) for a survey) can be encompassed in a unifying framework called the trajectory matching framework and defined by Neu and Szepesvári (2009). These algorithms try to find a reward function such that trajectories that one obtains by following the optimal policy with respect to this reward function become

close to the observed expert trajectories. Each step of the minimization requires an MDP to be solved. These algorithms are for instance Policy Matching (Neu and Szepesvári, 2007) where the objective is to minimize directly the *distance* between the obtained policy and the expert policy and Maximum Entropy IRL (Ziebart et al., 2008) where the objective is to minimize the Kullback-Leibler (KL) divergence between the distribution of trajectories. Those algorithms are incremental by nature and must solved several MDPs.

Others IRL algorithms such as Structured Classification for IRL (SCIRL) (Klein et al., 2012) and Cascaded Supervised IRL (CSI) (Klein et al., 2013) avoid resolving recursively MDPs. The main idea used in those algorithms is the link which can be made between a score function of a classifier and the quality function (Geist et al., 2013). SCIRL requires a linear parameterization but CSI is able to avoid this drawback. Indeed, CSI can be seen as a first step of classification followed by a step of regression which are two well know general Supervised Learning (SL) algorithms. Thus, if we use a first step of non-parametric classification followed by a non-parametric regression, CSI becomes a non-parametric IRL algorithm which provides a reward function, avoiding the choice of features. This seems to be a good candidate for an IRL algorithm adapted to the relational paradigm.

CSI

The idea behind CSI lies in the link between a score function of a multi-class classifier and the optimal quality function. Indeed, given a data set $D_C = (s_i, \pi_E(s_i) = a_i)_{1 \leq i \leq N_C}$ (the states s_i are seen as inputs and the actions a_i as labels), a classification algorithm outputs a decision rule $\pi_C \in A^S$ which generalizes the relation between inputs and labels observed in the data set D_C. One can remark, that it is quite easy to extract the set D_C from the set D_E which is the usual input of an IRL algorithm. In the case of a Score Based Classification (SBC) algorithm, the decision rule π_C is obtained via a score function $q_C \in \mathbb{R}^{S \times A}$ such that:

$$\forall s \in S, \pi_C(s) \in \operatorname*{argmax}_{a \in A} q_C(s, a). \quad (1)$$

A good classifier provides a policy π_C which chooses often the same action as π_E and the classification error is defined as:

$$\epsilon_C = \sum_{s \in S} \mathbf{1}_{\{\pi_E(s) \neq \pi_C(s)\}}. \quad (2)$$

If the expert is considered optimal with respect to the unknown reward R, π_E satisfies:

$$\forall s \in S, \pi_E(s) \in \operatorname*{argmax}_{a \in A} Q^*_R(s, a). \quad (3)$$

Thus, if the classification has a good performance (ϵ_C is small), q_C can be seen as a near-optimal quality function for the expert policy.

Moreover it is quite easy to find the reward R_C such that $q_C = Q^*_{R_C}$ by inverting the Bellman equation, $\forall (s, a) \in S \times A$:

$$R_C(s, a) = q_C(s, a) - \gamma \mathbb{E}_{P(s'|s,a)}[\max_{a' \in A} q_C(s', a')]. \quad (4)$$

So, R_C is a reward function for which the expert policy π_E is near-optimal if the classification error ϵ_C is small. It is possible to avoid trivial rewards by imposing that $\operatorname{argmax}_{a \in A} q_C(s, a)$ is a singleton: in that case R_C is a reward for which most of the expert actions are the only optimal actions if ϵ_C is small. For a sound proof of the performance of CSI, see the work of (Klein et al., 2013).

However, R_C can be computed exactly only if the dynamics P is provided. If not, we can still estimate R_C by regression. For this, we assume that we have a set of samples from the dynamics of the environment, called *non-expert* D_{NE}, and so we can easily form the set $D_{RL} = D_E \cup D_{NE}$. Our regression data set $D_R = \{(s_i, a_i), \hat{r}_i\}_{1 \leq i \leq N_{RL}}$ is then constructed from $D_{RL} = (s_i, a_i, s'_i)_{1 \leq i \leq N_{RL}}$ where $\hat{r}_i = q_C(s_i, a_i) - \gamma \max_{a \in A} q_C(s'_i, a)$. The regression can be done by a regression tree (Breiman, 1993; Geurts, Ernst, and Wehenkel, 2006) or by a least squares method for instance. The output of the regression algorithm is an estimate $\hat{R}_C$ of the target reward R_C.

Reward shaping The CSI algorithm has the appealing property of allowing one to learn a reward without relying on a simulator by casting the IRL problem as two supervised learning tasks. The two learning tasks are very different due to the relations between the demonstrations, quality functions and reward spaces. There is an infinity of quality functions that explains one set of demonstrations and the goal of the first step is to find one of them. On the other hand, there is only one reward for a given quality function and the goal of the second step is to approximate it.

We can assume, given the demonstration set, that some rewards are easier to approximate. However, the first step is totally agnostic of how hard it will be to approximate the reward when choosing the quality function.

For any function $f \in \mathbb{R}^S$, $q'_C(s, a) = q_C(s, a) + f(s)$ and q_C have the same optimal policy (Ng, Harada, and Russell, 1999). So, we propose to add an intermediary step to CSI where a shaping function, f, is found such that it makes the second step easier. Under the assumption that a function with low entropy is easier to learn with decision trees (as fewer leaves are needed), we use a stochastic optimization algorithm (a simplified version of CMA-ES (Hansen, Müller, and Koumoutsakos, 2003)) to find f such that when building D_R with q'_C the set of reward values has a low entropy.

Relational IRL

This section shows how the CSI algorithm can be lifted to the relational setting. In order to do so, we will rely on relational trees (Blockeel and De Raedt, 1998) as our relational regression algorithm and will introduce the use of a boosting approach to use in as the score-based classifier.

TILDE

TILDE is an algorithm designed to do classification and regression over relational data. It is a decision tree learner similar to C4.5 (Quinlan, 1993). It follows the principle of Top Down Induction of Decision Tree where the dataset is recursively split by building a tree according to some criteria until all data point in one subset share the same label. The only change made to handle relational data is to have first order logic test in each node. These tests are logical formula of one atom with free variables.

TILDE can be used for regression if one allows leaf to contain real numbers.

Boosted Large Margin Method

In order to obtain a SBC that handles relational data, we propose to use the boosted large margin method (Ratliff, Bagnell, and Srinivasa, 2007). The idea behind this algorithm is to optimize a function q_C such that it minimizes the following functional:

$$J_C(q) = \frac{1}{N_C} \sum_{i=1}^{N_C} \max_{a \in A} \{q(s_i, a) + l(s_i, a_i, a)\} - q(s_i, a_i), \tag{5}$$

where $l \in \mathbb{R}_+^{S \times A \times A}$ is called the margin function. If it is zero, minimizing $J_C(q)$ attempts to find a score function q_C for which the example labels are scored higher than all other labels. Choosing a nonzero margin function improves generalization (Ratliff, Bagnell, and Srinivasa, 2007). Instead of requiring only that the demonstrated label is scored higher than all other labels, it requires it to be better by an amount given by the margin function.

The functional J_C is optimized by functional gradient descent, a method really close to classical gradient descent except that the gradient is a function. No parametrization of the function q_C is available, yet it should have generalization properties, so one can compute the empirical gradient and use function approximation (regression) algorithm to approximate it (Grubb and Bagnell, 2011).

Learning the gradient can be seen as a projection of the gradient to K, a set of allowable directions. In boosting literature, this restriction set corresponds directly to the set of hypotheses generated by a weak learner. The nearest direction k^*, which is the projection of the gradient $\partial_q J_C$, is defined by:

Algorithm 1 Boosted Classification

Require: $q_0 \equiv 0$, $i = 0$, $T \in \mathbb{N}^*$ and $(\xi_j)_{\{j \in \mathbb{N}\}}$ a family of learning rates.
1: While $i < T$ do
2: Calculate $\partial_{q_i} J_C$.
3: Find k_i^* for the gradient $\partial_{q_i} J_C$.
4: $q_{i+1} = q_i - \xi_i k_i^*$, $i = i + 1$
5: end While, output $q_T = q_C$

$$k^* = \operatorname*{argmax}_{k \in K} \frac{\langle k, \partial_q J_C \rangle}{\|k\|}, \tag{6}$$

The boosted classification algorithm, which is a functional projected gradient descent, is summarized in the Algo. 1. We detail more how k_i^* is calculated. In order to find k^* for the gradient $\partial_q J_C$, we choose the restriction set K to be classification trees (Breiman, 1993) from $\mathbb{R}^{S \times A}$ to $\{-1, 1\}$ which is a particular choice of weak learners. In particular, $S \times A$ being logical formula we choose logical classification trees (Blockeel and De Raedt, 1998). Thus each $k \in K$ has the same norm. It is sufficient to find $k^* = \operatorname{argmax}_{k \in K} \langle k, \partial_q J_C \rangle$. To do that, we calculate $\partial_q J_C$ for a given $q \in \mathbb{R}^{S \times A}$:

$$\partial_q \max_{a \in A} \{q(s_i, a) + l(s_i, a_i, a)\} = \delta_{(s_i, a_i^*)}, \tag{7}$$

$$\partial_q q(s_i, a_i) = \delta_{(s_i, a_i)}, \tag{8}$$

$$\partial_q J_C = \frac{1}{N_C} \sum_{1 \leq i \leq N_C} \delta_{(s_i, a_i^*)} - \delta_{(s_i, a_i)}. \tag{9}$$

where $a_i^* = \operatorname{argmax}_{a \in A} [q(s_i, a_i) + l(s_i, a_i, a)]$. Then, we calculate $\langle k, \partial_q J_C \rangle$:

$$\langle k, \partial_q J_C \rangle = \frac{1}{N_C} \sum_{i=1}^{N_C} k(s_i, a_i^*) - k(s_i, a_i). \tag{10}$$

To maximize $\langle k, \partial_q J_C \rangle$, we have to find a classifier k such that $k(s_i, a_i^*) = 1$ and $k(s_i, a_i) = -1$ for a maximum of inputs. Thus, in order to obtain k^*, we train a classification tree with the following training set:

$$D_C = (((s_i, a_i), -1) \cup ((s_i, a_i^*), 1))_{1 \leq i \leq N_C} \tag{11}$$

Here the couples (s_i, a_i) and (s_i, a_i^*) are the inputs, and 1 and -1 are the labels. Finally, the output $q_T = q_C$ is a weighted addition of T classification trees.

Experiments

To validate the proposed approach, experiments have been run to 1) confirm Relational CSI (RCSI) can learn a relational reward from demonstrations and 2) study the influence of the different parameters.

Blocks world

Experimental setup To test RCSI in a quantitative way we use the following setup. From a target reward R^*, we compute the optimal policy π^*. The algorithm is given, as expert demonstrations, N_{expert} trajectories starting from a random state and ending when the (first) *wait* action is selected. As random demonstrations, the algorithm is given N_{random} one-step trajectories starting from random states. The optimal policy $\hat{\pi}$ of the learned reward $\hat{R}$ is then computed. As proposed by Klein et al. (2013) the expert dataset is added to the random one to ensure that it contains important (state, action, next-state) triplets such as (goal-state, wait, goal-state). Each experiment is repeated 100 times and the results are averaged.

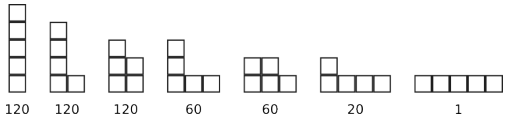

Figure 1: Different abstract states and the corresponding number of grounded states

To sample the random dataset we use the following distribution, P_{state} : we first draw uniformly from different relational configurations and then, for each one, uniformly from the possible groundings. Fig. 1 shows the different configurations and the number of grounded states for 5 blocks: there are 501 states and only one where all blocks are on the floor.

The main parameters are set as follows: 40 trees of maximum depth 4 are learned during the boosting step, the reward is learned with a tree of depth 4 which acts as a regularization parameter.

Performance measure To evaluate the proposed solution we define a performance measure, perf, that measures the ratio between the value of the learned policy (optimal policy derived from the learned reward) and the optimal one.

$$\text{perf}(\hat{r}) = \frac{1}{1000}\sum_{i=0}^{1000}\frac{V_{r^*}^{\hat{\pi}}(s_i)}{V_{r^*}^{\pi^*}(s_i)}, s_i \sim P_{state}$$

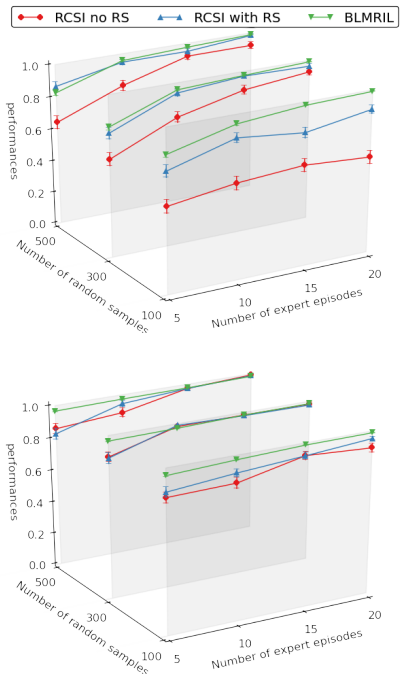

Figure 2: Performance for different amounts of training data on the stack (Top) and unstack (Bottom) task. Error bars represent Standard Error

Relation to TBRIL The first step of RCSI have the same algorithmic structure than TBRIL (Natarajan et al., 2011). The large margin method is a gradient-tree boosting algorithm, so both rely on gradient-tree boosting to do imitation learning. In fact, TBRIL could replace the first step of RCSI. The main difference between them is the functional that is optimized. In TBRIL the likelihood of expert actions is maximized. TBRIL and RCSI have very different goals, different assumptions, and thus must not be compared directly. However it is an interesting indicator of what to expect from RCSI. We use the first step of RCSI as this indicator under the name Boosted Large Margin Relational Imitation Learning (BLMRIL).

Sensibility to dataset sizes Figure 2 shows the results of using RCSI to learn the stack and unstack reward of the blocks world domain. RCSI is capable of learning the reward with enough expert and random training points. This graphs also shows that the reward shaping step increases the performance of the algorithm.

The set of parameters consisting of 300 random episodes and 15 experts episodes gives the best results and so we will use it in for the following experiments.

Noise robustness A important property of IRL is its robustness to noise in the demonstrations. Table. 1 presents the performance of RCSI given a percentage of noise in the demonstrations. Both rewards can still be learned with acceptable accuracy when noise percentage is equal or under 20%. Because we stop the expert episodes when the action *wait* is selected, states that have few actions have a higher probability to be stopping state.

Table 1: Impact of noise. N percent means that the expert chooses a random action $N\%$ of the time.

reward	0	10	20	50
stack	0.97	0.93	0.86	0.71
unstack	0.98	0.94	0.84	0.49

Transfer performance The main claim of relational learning is the capability of transferring among domains sizes. Fig. 3 shows the performance measures for varying the number of blocks, for training the algorithm and for evaluating it. For the stack reward, the graphs show almost no loss of performances due to a changing number of blocks. On the other hand, for the unstack reward, results are clearly worse when using 4, 5 or 6 blocks for training. By looking at the learned rewards, we have observed that, in most cases, one of the two following rewards is learned: One where the value is high when the number of *clear* is 4, 5, or 6 (depending on the number of blocks in the training set) and the a second where high rewards are given when the pattern $on(X, Y)$ and $block(Y)$ can not be matched. Both solutions are correct for a given domain size, as long as the

number of blocks is the same. However, if we change the number of blocks, an ambiguity appears and only the second one stays correct. Yet, there is no reason to prefer one over the other and it is not possible to make a system that could learn both rewards since the expert demonstrations would be similar.

One way to counter this phenomenon is to use a varying number of blocks during learning. This results is shown in the last column of Fig. 3 where a reward learned with a dataset mixing demonstrations with 4 and 5 blocks successfully transfers over to 6 blocks problems. One can also observe that learning the reward does performs better than directly learning the policy. This results would be surprising in a propositional domain and shows how relational representations allow to easily transfer among tasks. The advantage of IRL is shown in the next experiment.

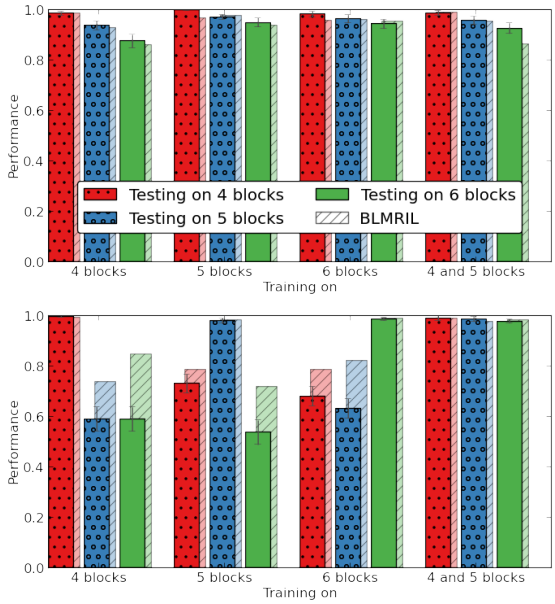

Figure 3: Performance of RCSI (and BLMRIL) when different number of blocks are used for training and testing the stack (Top) and unstack (Bottom) task.

Dynamics changes Learning the reward rather than directly the policy of the expert leads to more robust results even to large modifications of the dynamics of the environment. To evaluate this capability, the dynamics of the blocks world is modified between learning and evaluation. One of blocks has been made unmovable so in order to build a tower one must stack them on top of this fixed block. The results are shown in Table 2, as expected, in this setup, learning the reward leads to significantly better performance after changes in the dynamics.

Chair world

We tried our algorithm on another domain that models several different industrial tasks. In this domain there

Table 2: Impact on the performance of BLMRIL and RCSI when changing the domain dynamics.

Algorithm	constant	changed	**impact**
BLMRIL	0.98	0.57	**-0.41**
RCSI	0.95	0.89	**-0.06**

are two kinds of objects : parts and connectors. For instance a chair can be assembled with the parts : legs, sitting and back. Legs and backs have a male connector, while the sitting has five female connectors, 4 of them are compatible with legs and one is compatible with backs. The agent can *connect* or *unconnect* two connectors (if compatible) and *wait*. The *build* reward is 1 when every connector is connected and 0 otherwise.

This domain increases the difficulty for the algorithm because it violates some of its assumptions. The functional (5) that is minimized in the first step of RCSI enforces that only the expert actions are optimal, corresponding to assuming that the expert policy is deterministic. Nevertheless, we are still able to learn the correct reward achieving an average performance of 0.97, using a tree depth (for the reward) of 6 and 1000 random episodes.

Conclusions

In this paper, we presented the first approach to IRL for relational domains. We showed how the IRL algorithm CSI (Klein et al., 2013) can be generalized to the relational domain. The results indicate that it is possible to learn a relational reward that explains the expert behavior and derive from it a relational policy that achieves a performance similar to the expert. Besides generalizing the classification and regression steps in CSI, we introduced a reward shaping step to reduce the entropy of the reward function to estimate, and a new trans-dimensional perspective on data collection where we increase robustness to domain size changes by including in the training set demonstrations with different sizes.

IRL has the advantage of more compact explanations of behavior and more robustness to changes in the environment dynamics. The use of relational representations allowed for generalizing policies for changing the number of objects in a given domain. This shows one great strength of relational representations, and such result would not be possible in propositional or factored domains even with special feature design. Nevertheless, when the dynamics changes, we can see the interest of inferring the reward that allows the system to reevaluate the expected behavior in the new conditions.

In the future, we plan to apply these algorithms to more complex problems, relaxing the assumption that the expert has a deterministic policy, and considering the generalization of other IRL algorithms. Interesting generalizations are the active (Lopes, Melo, and Montesano, 2009) and interactive settings, multi-agent domains and consider the problem of simultaneously learn the symbolic representation and the task.

References

Argall, B.; Chernova, S.; and Veloso, M. 2009. A survey of robot learning from demonstration. *Robotics and Autonomous Systems* 57(5):469–483.

Blockeel, H., and De Raedt, L. 1998. Top-down induction of first-order logical decision trees. *Artificial intelligence* 101(1):285–297.

Breiman, L. 1993. *Classification and regression trees*. CRC press.

Džeroski, S.; De Raedt, L.; and Driessens, K. 2001. Relational reinforcement learning. *Machine learning* 43(1):7–52.

Friedman, J. H. 2001. Greedy function approximation: a gradient boosting machine. *Annals of Statistics* 1189–1232.

Geist, M.; Klein, E.; Piot, B.; Guermeur, Y.; and Pietquin, O. 2013. Around Inverse Reinforcement Learning and Score-based Classification. In *1st Multidisciplinary Conference on Reinforcement Learning and Decision Making (RLDM 2013)*.

Geurts, P.; Ernst, D.; and Wehenkel, L. 2006. Extremely randomized trees. *Machine learning*.

Grubb, A., and Bagnell, J. 2011. Generalized boosting algorithms for convex optimization. In *Proc. of ICML*.

Hansen, N.; Müller, S.; and Koumoutsakos, P. 2003. Reducing the time complexity of the derandomized evolution strategy with covariance matrix adaptation (cma-es). *Evolutionary Computation* 11(1):1–18.

Kersting, K.; Otterlo, M. V.; and De Raedt, L. 2004. Bellman goes relational. In *Proceedings of the twenty-first international conference on Machine learning*, 59. ACM.

Khardon, R. 1999. Learning action strategies for planning domains. *Artificial Intelligence* 113(1):125–148.

Klein, E.; Geist, M.; Piot, B.; and Pietquin, O. 2012. Inverse reinforcement learning through structured classification. In *Proc. of NIPS*.

Klein, E.; Piot, B.; Geist, M.; and Pietquin, O. 2013. A cascaded supervised learning approach to inverse reinforcement learning. In *Proc. of ECML*. Springer.

Lang, T., and Toussaint, M. 2010. Planning with noisy probabilistic relational rules. *Journal of Artificial Intelligence Research* 39(1):1–49.

Lang, T.; Toussaint, M.; and Kersting, K. 2012. Exploration in relational domains for model-based reinforcement learning. *Journal of Machine Learning Research* 13:3691–3734.

Lopes, M.; Melo, F.; Kenward, B.; and Santos-Victor, J. 2009. A computational model of social-learning mechanisms. *Adaptive Behavior* 467(17).

Lopes, M.; Melo, F. S.; and Montesano, L. 2009. Active learning for reward estimation in inverse reinforcement learning. In *Machine Learning and Knowledge Discovery in Databases (ECML/PKDD'09)*.

Natarajan, S.; Joshi, S.; Tadepalli, P.; Kersting, K.; and Shavlik, J. 2011. Imitation learning in relational domains: A functional-gradient boosting approach. In *Inter. Joint Conf. on Artificial Intelligence (IJCAI)*, 1414–1420.

Neu, G., and Szepesvári, C. 2007. Apprenticeship learning using inverse reinforcement learning and gradient methods. In *Proc. of UAI*.

Neu, G., and Szepesvári, C. 2009. Training parsers by inverse reinforcement learning. *Machine learning* 77(2).

Ng, A. Y.; Harada, D.; and Russell, S. 1999. Policy invariance under reward transformations: Theory and application to reward shaping. In *ICML*, volume 99, 278–287.

Ng, A.; Russell, S.; et al. 2000. Algorithms for inverse reinforcement learning. In *Proc. of ICML*.

Quinlan, J. R. 1993. *C4. 5: programs for machine learning*, volume 1. Morgan kaufmann.

Ratliff, N.; Bagnell, J.; and Srinivasa, S. 2007. Imitation learning for locomotion and manipulation. In *Proc. of IEEE-RAS International Conference on Humanoid Robots*.

Russell, S. 1998. Learning agents for uncertain environments. In *Proc. of COLT*.

Schaal, S. 1999. Is imitation learning the route to humanoid robots. *Trends in Cognitive Sciences* 3(6):233–242.

Segre, A., and DeJong, G. 1985. Explanation-based manipulator learning: Acquisition of planning ability through observation. In *Robotics and Automation. Proceedings. 1985 IEEE International Conference on*, volume 2, 555–560. IEEE.

Shavlik, J. W., and DeJong, G. 1987. Bagger: An ebl system that extends and generalizes explanations. In *AAAI*, 516–520.

Yoon, S.; Fern, A.; and Givan, R. 2002. Inductive policy selection for first-order mdps. In *Proceedings of the Eighteenth conference on Uncertainty in artificial intelligence*, 568–576. Morgan Kaufmann Publishers Inc.

Ziebart, B. D.; Maas, A. L.; Bagnell, J. A.; and Dey, A. K. 2008. Maximum entropy inverse reinforcement learning. In *AAAI*, 1433–1438.

Temporal Segmentation of Pair-Wise Interaction Phases in Sequential Manipulation Demonstrations*

Andrea Baisero[1] Yoan Mollard[2] Manuel Lopes[2] Marc Toussaint[1] Ingo Lütkebohle[1]

***Abstract*— We consider how to learn a representation from bimanual assembly tasks from demonstration. We propose to analyze the demonstration in terms of the (potentially concurrent) interaction phases between any pair of involved bodies (hands, tools, objects). These interaction phases are the key to extract a more abstract description of the demonstration. In particular one may assume that the goal of each interaction phase is to achieve a certain geometric constraint. This generalized previous approaches on LfD to consider not just the motion of the end-effector but also the relational properties of the motion of the objects. We present an approach to train a Conditional Random Field to detect the pair-wise interaction phases and based on this labeling analyze the geometric constraints that are established. In this way we extract a higher level task oriented description of the demonstrated sequential manipulation. We test our system using data from a person assembling a toolbox of 5 parts using a screwdriver and two hands. We consider how to learn a representation from bimanual assembly tasks from demonstration.**

I. INTRODUCTION

A major challenge in robotics is to provide intuitive and easy ways for non-experts to instruct and work with robotic systems. This is true in household domains, where consumers demand more flexibility and personalized functionalities, as well as in new industries that are required to be much more flexible in their production lines and switch between products to meet this new demand. Learning from Demonstration (LfD) is a teaching paradigm through which the robotic system learns to perform new tasks directly by observing the tasks themselves, which are showcased by a demonstrator through either kinesthetic teaching or his own body [**?**], [**?**]. While such approaches are intuitive for the user, they are limited in several respects: with few exceptions ([**?**], see below) previous LfD work has focused on the instruction of single robot movements rather than the instruction of complex sequential manipulation tasks, such as the assembly of furniture using tools. Furthermore, such type of instruction is rather low level, providing data for the motion primitives itself, whereas instructing complex and sequential tasks seems more efficient and intuitive on a more abstract level.

We approach LfD under the assumption that sequential manipulation can well be understood as a sequence of interaction phases, where the goal of each interaction phase is to move objects into a fixed geometric configuration, which we call *constraint*. If there are multiple manipulators (hands or tools), multiple such manipulator-object (or manipulator-tool, tool-object) interactions can concurrently be active. Under this view, instructing a robot to perform a sequential manipulation (potentially involving tools) implies to demonstrate which interactions should be performed in which order and what their desired outcomes are in terms of the created geometric constraints. A LfD algorithm should therefore automatically analyze demonstrations by detecting these interaction phases and extracting the achieved geometric constraints. This approach leads to a much more abstract understanding of the task from demonstrations rather than modeling the low-level robot or manipulator motions themselves.

We propose to train a Conditional Random Field (CRF) to detect the interaction phases between any pair of moving bodies (manipulator, tool or object). Roughly, the CRF learns to exploit features that indicate a temporarily rigid transformation between two moving bodies. The trained CRF will automatically segment a sequential manipulation demonstration into its (concurrent) interaction phases. Based on this, we analyze in a second stage the outcome of these interaction phases w.r.t. the geometric constraints that have been established by the interaction, for instance that a screwdriver has been positioned right above a screw. This gives a task-space description of the goal of each interaction phase and thereby a goal-oriented analysis of the demonstration.

In Section II we first review the related work. We then describe the CRF formalism, its features and training methods in Section III. We briefly discuss, in Section III-E, the possibility of extracting constraints based on the CRF-guided segmentation. Finally, we demonstrate our approach on data recorded from a human assembling a toolbox in Section IV, and conclude in Section V.

II. RELATED WORK

Learning from Demonstration algorithms [**?**] have mostly focused on forms of demonstration where any hierarchical or sequential aspect is explicitly described by the teacher. The notion of key-frame demonstration was introduced in [**?**], [**?**], where users are not asked to provide full demonstrations, but rather only the key aspects by presenting the most important via-points. Recent research started to consider how a complex demonstration can be represented and decomposed in simpler parts.

Niekum et al. [**?**], [**?**] presented an integrated approach to segment manipulation demonstrations into actions, each represented by a Dynamic Motion Primitive (DMP). The

*Work supported by 3rdHand

[1]Machine Learning and Robotics Lab, University of Stuttgart, Germany. firstname.lastname@ipvs.uni-stuttgart.de

[2]Flowers Team, French Institute for Research in Computer Science and Automation (Inria), France. firstname.lastname@inria.fr

segmentation of demonstrations is realized with a Beta Process autoregressive HMM (BP-AR-HMM) [**?**], which not only produces a segmentation but also a corresponding labeling, i.e. an association with latent variable values. This approach is promising to identify segments of robot motion and compile these into DMPs. However, our aim is to identify the crucial interaction phases between manipulators and objects that signify the initial and final moments of an object manipulation. Therefore, in contrast to the problem setting of Niekum et al., we aim at a binary labeling of pair-wise interactions.

Pieropan et al. [**?**] model human activity while also annotating different segments based on the object-hand relation. While their aim of characterizing the affordances of objects differs from ours, the idea to base this characterization on the time sequence of object-hand (or object-object) relation is similar to our approach. However, their method of labeling the object-hand relations is heuristic, based on proximity and approach directions in the 2D video and not suited for our problem.

A series of works [**?**], [**?**], [**?**], [**?**] formulate integrated probabilistic models of sequential or superimposed motion primitives which, when fitted to data, imply a segmentation of motions. Again, the goal of these approaches is to extract specific motion primitives from data rather than to identify interaction phases. Barbic et al. [**?**] provide a good discussion of traditional methods for segmenting motion capture data, including the detection of zero crossings of angular velocities [**?**], and their own PCA approach. Incremental creation of motor primitives has also been used to create a dictionary of full-body motions relying on zero-acceleration heuristics for segmentation [**?**]. Another approach relied on clustering to segment low-level sequences to learn motor primitives and grammar at an high level [**?**]. However, in all these approaches the problem setting focuses on the extraction of elemental motion primitives rather than analyzing pair-wise interaction phases between manipulators, tools and objects.

In conclusion, we are not aware of previous work that explicitly aimed at training a segmentation algorithm to identify atomic manipulations in the midst of a more complex demonstration—as a prerequisite to describe such complex manipulations as a sequence of interactions which generate specific object relations.

III. CONDITIONAL RANDOM FIELDS TO DETECT INTERACTION PHASES

A generic CRF models a conditional probability distribution as a normalized product of (typically log-linear) potentials:

$$p(\boldsymbol{y} \mid \boldsymbol{x}) = \mathcal{Z}(\boldsymbol{x})^{-1} \prod_k \psi_k(\boldsymbol{y}, \boldsymbol{x}) \tag{1}$$

where the potentials may establish arbitrary correlations between any subset of the latent variables $\boldsymbol{y}$ and the observed variables $\boldsymbol{x}$, and the partition function $\mathcal{Z}(\boldsymbol{x}) = \sum_{\tilde{y}} \prod_k \psi_k(\tilde{\boldsymbol{y}}, \boldsymbol{x})$ acts as a normalizing constant.

In a linear-chain CRF, the latent variables form a sequence which can be referenced by an index, and the potentials only couple directly adjacent latent variables.

A linear-chain CRF describes the following distribution:

$$p(\boldsymbol{y} \mid \boldsymbol{x}) = \mathcal{Z}(\boldsymbol{x})^{-1} \prod_{t=1}^{T} \psi_t(\boldsymbol{y}, \boldsymbol{x}) \tag{2}$$

$$\psi_t(\boldsymbol{y}, \boldsymbol{x}) = \exp\left(\phi_t(y_t, y_{t-1}, \boldsymbol{x})^\intercal \theta\right) . \tag{3}$$

where ϕ_t is a *feature vector* that may include features that couple two consecutive latent variable y_{t-1} and y_t as well as features that couple the latent variables to the observed sequence $\boldsymbol{x}$.

The contents of the feature vector are discussed in Sections III-B and III-C[1].

A. Training

The model parameters θ are learned by applying the Maximum Likelihood (ML) estimator to a set of labeled demonstrations. The neg-log likelihood of a linear-chain CRF model is a convex function of θ which means that it has only one optimal point, and that it is a relatively simple problem to optimize, and that a variety of first- and second-order iterative algorithms already exist.

Given a set of labeled sequences $\mathcal{D} = \{(\boldsymbol{y}^{(i)}, \boldsymbol{x}^{(i)})\}_{i=1}^{n}$, the neg-log-likelihood associated with it is

$$\mathcal{L}(\theta; \mathcal{D}) = \sum_{i=1}^{n} \mathcal{L}(\theta; \boldsymbol{y}^{(i)}, \boldsymbol{x}^{(i)}) \tag{4}$$

$$\begin{aligned} \mathcal{L}(\theta; \boldsymbol{y}, \boldsymbol{x}) &= -\log p(\boldsymbol{y} \mid \boldsymbol{x}; \theta) \\ &= \log \mathcal{Z}(\boldsymbol{x}; \theta) - \sum_{t=1}^{T} \phi_t(y_t, y_{t-1}, \boldsymbol{x})^\intercal \theta \end{aligned} \tag{5}$$

whereas the respective gradient is

$$\nabla \mathcal{L}(\theta; \mathcal{D}) = \sum_{i=1}^{n} \nabla \mathcal{L}(\theta; \boldsymbol{y}^{(i)}, \boldsymbol{x}^{(i)}) \tag{6}$$

$$\begin{aligned} \nabla \mathcal{L}(\theta; \boldsymbol{y}, \boldsymbol{x}) = &\sum_{t=1}^{T} \sum_{\tilde{y}_t, \tilde{y}_{t-1}} \phi_t(\tilde{y}_t, \tilde{y}_{t-1}, \boldsymbol{x})\ p_t(\tilde{y}_t, \tilde{y}_{t-1} \mid \boldsymbol{x}) \\ &- \sum_{t=1}^{T} \phi_t(y_t, y_{t-1}, \boldsymbol{x}) \end{aligned} \tag{7}$$

The values of the partition function $\mathcal{Z}(\boldsymbol{x}; \theta)$ and of the joint conditional probabilities $p_t(\tilde{y}_t, \tilde{y}_{t-1} \mid \boldsymbol{x})$ can be efficiently computed using the forward-backward algorithm, the details of which we leave out.

B. TRANSITION FEATURES

In our model, objects are characterized exclusively by their trajectories, which are assumed to be fully observable. Given an object A, its trajectory is a sequence of poses $\boldsymbol{\alpha} := (\alpha_t)_{t=1}^{T}$, where $\alpha_t \in \mathrm{SE}(3)$. Positions and orientations are denoted as $\alpha_{\pi,t} \in \mathrm{T}(3)$ and $\alpha_{\rho,t} \in \mathrm{SO}(3)$ respectively.

[1]As a detail we mention that, because there is no latent variable y_0 in our model, the feature vector ϕ_1 needs special consideration: For simplicity of notation, we define all features that depend on y_0 as identically zero.

Given two trajectories $\boldsymbol{\alpha}$ and $\boldsymbol{\beta}$, we model the interaction between the respective objects A and B with a linear-chain CRF. We define the latent variables $\boldsymbol{y}$ of the CRF as a sequence of binary variables which represent the interaction between the objects at each time step; and the observation sequence as the fully observed trajectories of the two objects throughout the whole demonstration $\boldsymbol{x} = (\boldsymbol{\alpha}, \boldsymbol{\beta})$.

Our feature vector ϕ_t is composed of a transition feature vector and a state feature vector, $\phi_t = (\phi_{\tau,t}^\intercal, \phi_{\sigma,t}^\intercal)^\intercal$. As is typical in linear-chain CRF, the transition features $\phi_{\tau,t}$ indicate the discrete latent state transitions and do not depend on the observation sequence $\boldsymbol{x}$ (although in principle, they could):

$$\phi_{\tau,t}(y_t, y_{t-1}) := \begin{pmatrix} \mathbb{I}[y_t = 0]\,\mathbb{I}[y_{t-1} = 0] \\ \mathbb{I}[y_t = 0]\,\mathbb{I}[y_{t-1} = 1] \\ \mathbb{I}[y_t = 1]\,\mathbb{I}[y_{t-1} = 0] \\ \mathbb{I}[y_t = 1]\,\mathbb{I}[y_{t-1} = 1] \end{pmatrix}, \tag{8}$$

where $\mathbb{I}$ is the indicator function.

C. STATE FEATURES

The type of interaction we are trying to model may be described as one which takes place when the same time-varying transformation is applied to both objects. We want to detect whether both objects are moving, and whether the relative transformation between them remains constant throughout the movement.

We choose the state features $\phi_{\sigma,t}$ to capture the individual object movements as well as relative transformation between the objects.

1) OBJECT MOVEMENT: For any given object A, we compute features which vary according to the object's movement at any given time step. Because position and orientation represent ontologically different entities, we compute separate features for transitional movements and rotational movements, respectively $\nu_{\pi,t}(A)$ and $\nu_{\rho,t}(A)$.

A straightforward choice for $\nu_{\pi,t}$ and $\nu_{\rho,t}$ is represented by the instantaneous linear and angular speeds at time t. However, instantaneous speeds may vary wildly between adjacent time steps, and may be relatively high even in situations where the actual movement is neglectible, resulting in noisy features and degraded inference performance.

Instead, we define a discrete window length ω and compute a scalar variance measure from the object's ω most recent absolute positions and rotations:

$$\mu_{\nu_\pi,t}(A) = \omega^{-1} \sum_{t'=t-\omega+1}^{t} \alpha_{\pi,t'} \tag{9}$$

$$\mu_{\nu_\rho,t}(A) = \omega^{-1} \sum_{t'=t-\omega+1}^{t} \alpha_{\rho,t'} \tag{10}$$

$$\nu_{\pi,t}(A) = \omega^{-1} \sum_{t'=t-\omega+1}^{t} \left\|\alpha_{\pi,t'}(A) - \mu_{\nu_\pi,t}(A)\right\|^2 \tag{11}$$

$$\nu_{\rho,t}(A) = \omega^{-1} \sum_{t'=t-\omega+1}^{t} \left\|\alpha_{\rho,t'}(A) - \mu_{\nu_\rho,t}(A)\right\|^2 \tag{12}$$

One practical consideration to correctly compute the above quantities concerns the fact that the space of quaternion is a double-covering group of SO(3). To avoid introducing fictitious variance, all quaternions are adequately inverted such that the scalar products of quaternions belonging to adjacent time-steps are always positive.

2) RELATIVE TRANSFORMATION: A similar approach is used to construct the features of the relative transformation between objects. Given A and B and their respective trajectories $\boldsymbol{\alpha}$ and $\boldsymbol{\beta}$, we define $\boldsymbol{\delta}$ as the trajectory of B w.r.t. the coordinate frame of A,

$$\delta_{\pi,t} = \alpha_{\rho,t}^{-1}(\beta_{\pi,t} - \alpha_{\pi,t}) \tag{13}$$

$$\delta_{\rho,t} = \alpha_{\rho,t}^{-1}\beta_{\rho,t} \tag{14}$$

We finally compute features $\zeta_{\pi,t}(A,B)$ and $\zeta_{\rho,t}(A,B)$ using the same variance measure and the same window-length parameter ω described previously:

$$\mu_{\zeta_\pi,t}(A) = \omega^{-1} \sum_{t'=t-\omega+1}^{t} \delta_{\pi,t'} \tag{15}$$

$$\mu_{\zeta_\rho,t}(A) = \omega^{-1} \sum_{t'=t-\omega+1}^{t} \delta_{\rho,t'} \tag{16}$$

$$\zeta_{\pi,t}(A) = \omega^{-1} \sum_{t'=t-\omega+1}^{t} \left\|\delta_{\pi,t'}(A) - \mu_{\zeta_\pi,t}(A)\right\|^2 \tag{17}$$

$$\zeta_{\rho,t}(A) = \omega^{-1} \sum_{t'=t-\omega+1}^{t} \left\|\delta_{\rho,t'}(A) - \mu_{\zeta_\rho,t}(A)\right\|^2 \tag{18}$$

Our intended goal is to create a model which is agnostic to the identities of the involved objects, and invariant to properties other than the trajectories themselves. There is thus no justification in having different parameters associated with features $\nu_{\pi,t}(A)$ and $\nu_{\pi,t}(B)$; as well as $\nu_{\rho,t}(A)$ and $\nu_{\rho,t}(B)$. To enforce the fact that the above pairs of features share the same pair of parameters, we add the respective values into the resulting state feature vector, which is as follows:

$$\phi_{\sigma,t}(y_t, \boldsymbol{x}) = \begin{pmatrix} \mathbb{I}[y_t = 1]\,(\nu_{\pi,t}(A) + \nu_{\pi,t}(B)) \\ \mathbb{I}[y_t = 1]\,(\nu_{\rho,t}(A) + \nu_{\rho,t}(B)) \\ \mathbb{I}[y_t = 1]\,\zeta_{\pi,t}(A,B) \\ \mathbb{I}[y_t = 1]\,\zeta_{\rho,t}(A,B) \end{pmatrix} \tag{19}$$

D. RELATION WITH LOGISTIC REGRESSION

The described CRF is a generalization of logistic regression, which learns a classification from $\boldsymbol{x}$ to y_t using a discriminative function linear in the state features $\phi_{\sigma,t}(y_t, \boldsymbol{x})$ only. In that sense, the CRF is a smoothed version of such independent logistic regression classifications. We exploit this as follows: We first train a logistic regression classifier using standard Newton methods and then adopt the resulting parameters as an initialization of the CRF training, which greatly speeds up training effort.

In the experimental section we also report on the benefit of using the CRF versus the non-smoothed logistic regression classifications.

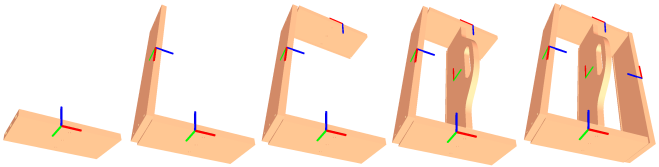

Fig. 1: Representation of a "build toolbox" task description which is extracted from the respective demonstration. Each figure shows the state of the toolbox at points in time which are associated with the end of the manipulation of a wooden piece.

E. CONSTRAINT EXTRACTION

In certain tasks, it is important to be able to analyze different aspects of the motion of the agent and objects separately. For example, an assembly task features several types of behaviors: The agent may act on a single object by grasping it and using it as a tool, or moving it to another location; he may perform an action which results in the assembly of two previously distinct parts; or he may perform other auxiliary actions to deal, for example, with obstacles. The behaviors of the second type which arise during a demonstration provide a transferable description of the performed task which is independent from the executing system. The relevance of this type of task description for the purpose of transferring information from a human execution to a robotic system is clear.

We define an assembly task as a sequence of operations that enforce rigid body constraints between pairs of objects. The CRF model we presented in the previous section is able to detect several segments where pairs of objects display coherent motions. We thus regard some of the transitions between those segments as key points in the demonstration where constraints are being created (or potentially removed) between the respective pair of objects.

Figure 1 shows a sequence of constraints which is extracted using the automated segmentation as a starting point.

IV. EVALUATIONS

A. DATA COLLECTION

Using a motion-capture setup, we recorded a demonstration where a human assembles a toolbox consisting of 5 wooden pieces. The task involves the pick and place of wood pieces, the insertion of screws in appropriate holes, and the use of a screwdriver to screw pieces together.

The object trajectories were recorded using a Polhemus G4 magnetic tracking hardware system, which is able to provide poses at 120Hz. In practice, we have no need to create models which describe the interaction so densely. To soften the load of the model's training and inference, we generate about 10 latent variables for each second of demonstration in our model. Also for practical reasons, only the human's thumb, index and middle fingers and the wooden pieces were being tracked. Object models were constructed for the sole purpose of showcasing results: each object is identified solely by the pose measured by the corresponding sensor.

	left hand / training	right hand / test
Logit Reg	12377.2 (0.4391)	16371.5 (0.5809)
CRF	2464.9 (0.0875)	1912.8 (0.0679)

TABLE I: Neg-log-likelihoods of annotated training and test sequences for the logistic regression and CRF segmentation. Sequences involving the left hand are used for training, and the ones involving the right hand are not. In parentheses, the average neg-log-likelihood per latent variable.

The demonstration was hand-labeled using professional software developed to provide natural language annotations to videos[2], which we used to directly annotate one of the video-recordings. The resulting annotations indicate, for each hand-object pair, whether the hand is actively manipulating the object at any given time during the demonstration. The labels of each finger is assumed to coincide with the corresponding hand's labels.

B. RESULTS

We use the labeled motions for the left hand as training data for the CRF model, and the motions relative to the right hand as test data (although we also show the outputs relative to the left hand for completeness). Table I contains the neg-log-likelihoods computed by the logistic regression and the CRF model. Figure 2 contains screen-shots of the segmentation when applied to the toolbox assembly task. Figure 3 shows the individual output sequences for a various finger-object pairs.

V. CONCLUSIONS

We have considered how to analyze complex sequential demonstrations of bimanual assembly tasks. In contrast with classical LfD approaches, whose main concern is often that of being able to replicate the performer's motions, we focused our attention on detecting higher-level interactions between objects in the scene, which represent the real motivation behind each of the performer's motions.

We defined a graphical model with which to detect such interaction phases which only requires the estimated trajectories of the involved objects to be specified. The model was used to segment an assembly demonstration and to extract the individual manipulation actions which were performed. Furthermore, an elaboration was provided on how to use such segmentation to generate an assembly task description which is independent on the actual execution.

[2]ELAN 4.7.2, Max Planck Institute for Psycholinguistics, The Language Archive, [**?**].

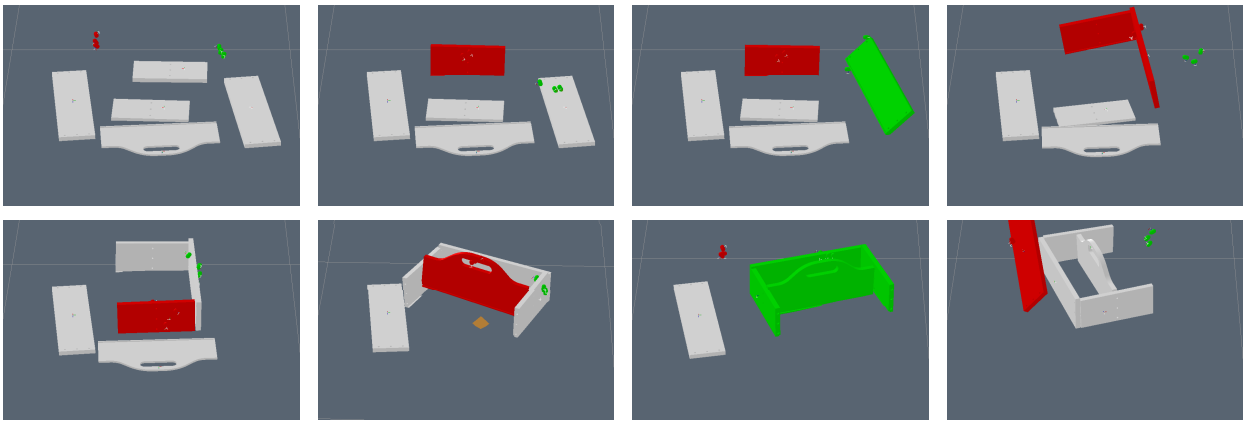

Fig. 2: Reconstruction of the toolbox assembly demonstration, with segmentation output shown by color-coding the objects. The fingers are shown as colored rounded cylinders; right hand in red and left hand in green (top-left frame). The model's segmentation is shown by coloring each object according to a hand's color, if at least two of the outputs of the hand's fingers indicate that an interaction is taking place.

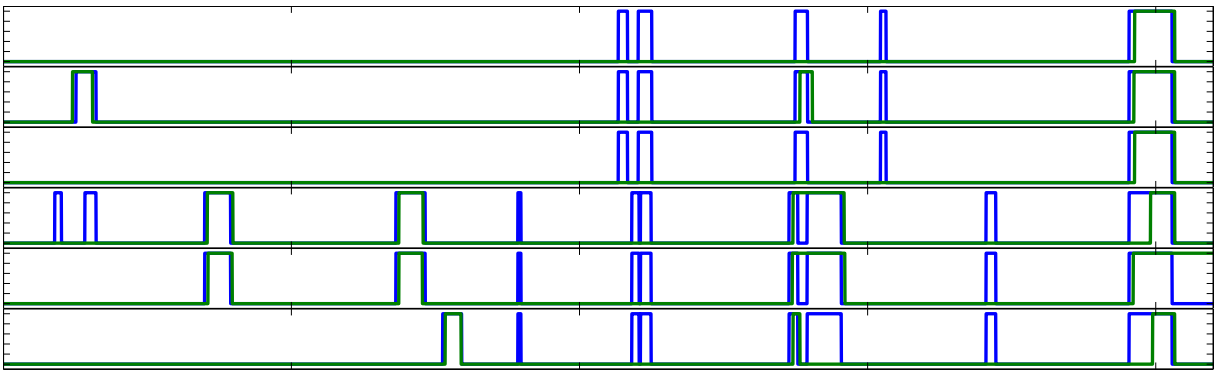

Fig. 3: Each of the above plots contains the labeled ground truth interaction sequence (in blue) and the CRF model's most likely sequence as computed by the Viterbi algorithm (in green) for the whole toolbox assembly demonstration. From top to bottom, the plots refer to the following pairs: Left hand index with toolbox front side; left hand middle finger with toolbox left side; left hand thumb with toolbox middle piece (handle); right hand index with toolbox front side; right hand middle finger with toolbox left side; right hand thumb with toolbox middle piece (handle);

DRAFT
A Parser for Constructing Movement Primitive Libraries

Rudolf Lioutikov[1], Guilherme Maeda[1], Jan Peters[1,2] and Gerhard Neumann[1]

***Abstract*— Movement primitives are a well established approach for encoding and executing movements. While the primitives themselves have been extensively researched, the concept of movement primitive libraries has not received as much attention. The goal of this work is to learn a primitive library from unsegmented demonstrations. The learned library is used to parse previously unseen demonstration into a sequence of recurring primitives. At the same time the library provides a set of primitives which can be sequenced in order to solve previously undemonstrated tasks. Current approaches separate the segmentation of the demonstration from the learning of the primitives. This separation neglects the mutual dependency between the found segments and the learned primitives. However, this dependency can be used to improve the quality of the found segments as well as the learned primitives. Therefore we propose a novel method which, in contrast to previous work, respects this dependency. Based on probabilistic inference our approach is able to learn the segmentation and the primitives simultaneously. We compare our work to state-of-art methods and show the advantages of such a combined approach. Experiments on a complex bimanual robot platform demonstrate the applicability of our method in real world robot tasks.**

I. INTRODUCTION

A key goal of modern robotics is to provide robots with the ability to learn new tasks. A commonly followed concept in order to achieve such behavior is imitation-learning [1]. The idea is to provide the robot with one or more demonstrations of the tasks, which the robot subsequently applies and improves. This idea was applied successfully on a variety of tasks, e.g. the ball-in-a-cup task or even playing table tennis [1]. In both examples the entire task consisted of a single motion, that the robot needed to learn. A common way to encode such single motions or actions are Movement Primitives.

However, representing more complex, multi-step tasks as single Movement Primitives implies a great loss of generality. Considering such tasks as a sequence of Primitives offers multiple advantages, e.g. generalization at the points between the primitives is simpler, the same set of primitives can be used to execute different tasks and the task can be adapted by replacing one primitive of the sequence by a different one. At the same time problems arising from such a sequence of primitives lead to the question how to obtain these primitives and how to subsequently sequence them. In this paper we address the former problem by proposing a framework for the automated segmentation of Probabilistic Movement Primitives [2] given demonstrations of the entire task.

[1]Technische Universität Darmstadt, Intelligent Autonomous Systems Group, Darmstadt, Germany {lioutikov, maeda, peters, neumann}@ias.tu-darmstadt.de

[2]Max Planck Institute for Intelligent Systems, Department Empirical Inference, Tübingen, Germany jan.peters@tuebingen.mpg.de

II. Related Work

Algorithms for automatic segmentation has long attracted the attention of researchers, especially in regards to the classification of motion. For example, in the work of Brand and Kettnaker [3] video images are analyzed to train a HMM to classify if a person is walking, running or crouching. Segmentation has been used to group motions hierarchically, where higher level representations of symbols can then be used to orchestrate and generate low level robot movements [4]. More recently, Kulic et al. [5] proposes on-line segmentation based on HMMs for robot movement generation. The method creates a tree of primitives; the lower nodes representing detailed movements with generality increasing towards the root.

In contrast, here we propose an algorithm that directly encodes trajectories as Dynamical Movement Primitives (DMP) [6] or Probabilistic Movement Primitives (ProMPs) [2]. The use of either DMPs or ProMPs brings several advantages. Segmentation becomes invariant to movements with different velocities due to the use of a phase variable, allowing the generation of compact libraries of movement. Algorithms for segmentation can take advantage of a much lower dimensional weight space representation rather than the robot trajectories. The output of the method is a continuous low-level primitive that can be used to directly control the robot.

From the movement primitive perspective, our algorithm relates to the work of [7], and [8] where DMPs have been used in different ways. In the first, a library of primitives assumed given, while in our work we design our algorithm to start from an empty set. Compared to the work of [8] where segmentation and learning of primitives are independent processes, here we propose an integrating segmentation and primitive learning as an iterative process, which should, therefor improve the final quality of the solution.

A usual problem of movement segmentation is the use of a heuristic to inform the initial segmentation. Perhaps the most natural and intuitive criterion is the zero crossing velocity (ZCV) which has been used by several authors [9], [10]. Here, we also use ZCV as a

heuristic to identify potential candidates. In the context of movement primitives, however, zero crossing velocities usually leads to over-segmentation, especially when the robot trajectories move at low speed. As it will be shown this will require from our method the capability to merge and decrease the number of segments.

III. METHOD

We will consider trajectories τ_i with several possible cutting points (candidate cuts) c_{ij}. The task is to find a probabilistic movement primitive representation (as a mixture of MPs) and an appropriate segmentation for the trajectories. This is an hidden variable problem that can be solved with probabilistic inference. We need two EM algorithms, the inner loop runs the EM on the mixture models and the outer loop on the segments. Lets for now assume that we have a generative model $p(S|\theta)$ for segments, where $S = [q_{t1}, q_{t2}, ..., q_{tN}]$ is a segment and θ are the parameters of the model. For example, the generative model could be a mixture model of ProMPs, i.e.,

$$p(S|\vec{\theta}) = \sum_k \pi_k p(q_{t_1:t_N}|k),$$

where $p(q_{t_1:t_N}|k)$ is the probability of a trajectory given a single ProMP k. We assume that we can train the model by maximum likelihood, or, more generally, by a weighted maximium likelihood estimate. Given several segments $\vec{S} = [S_1, \dots S_N]$, the weighted maximum likelihood estimate max is given by

$$L_1(\vec{S}, \vec{u}) = \sum_i u_i \log p(S_i|\vec{\theta}),$$

where u_i is the weighting for the $i-$th segment. The weighted log-likelihood can be easily optimized by EM for Gaussian mixture models. However, the segments are not given as we do not know which cuts are active or not. Hence, we need to optimize the following log-likelihood (only given for a single trajectory)

$$L_2(\tau) = \log \sum_{j=1}^{2^M} p(D_j) \prod_{S_h \in D_j} p(S_h|\vec{\theta}),$$

where D_j is one of the possible segmentations (out of 2^M segmentations). In this formulation, we marginalized out all possible segmentations D_j. $p(D_j)$ is a prior over a certain segmentation. Typically, we can set this prior uninformative, i.e., $p(D_j) = 1/2^M$

A. Outer EM loop

This model is very hard to optimize. However, if we treat the segmentations as hidden variables we can use EM to find a locally optimal model.

The EM on the possible segmentations can be formalized as follows.

a) E-Step:: In the E-Step, we have to compute the probability of each segmentation D_j given our current model for the segments $p(S|\vec{\theta})$, i.e., the unnormalized responsibilities are given by

$$\tilde{\gamma}_j = p(D_j) \prod_{S_h \in D_j} p(S_h|\vec{\theta})$$

and the normalized therefore by

$$\gamma_j = p(D_j|\vec{\theta}, \tau) = \frac{\tilde{\gamma}_j}{\sum_l \tilde{\gamma}_l}.$$

b) M-Step:: In the M-step, we need to optimize the complete-data log likelihood which is given by

$$\begin{aligned} Q(\vec{\theta}, \vec{\theta}_{\text{old}}) &= \sum_{j=1}^{2^M} p(D_j|\vec{\theta}_{\text{old}}, \tau) \log \left(p(D_j) \prod_{S_h \in D_j} p(S_h|\vec{\theta}) \right) \quad (1) \\ &= \sum_{j=1}^{2^M} \gamma_j \sum_{S_h \in D_j} \log p(S_h|\vec{\theta}), \quad (2) \end{aligned}$$

where we neglect the prior $p(D)$ by assuming it is fixed. We can formulate Eq. 2 more efficiently by summing over all possible segments instead of all possible segmentations.

$$Q(\vec{\theta}, \vec{\theta}_{\text{old}}) = \sum_{h=1}^{|S|} \left(\sum_{j=1}^{2^M} \gamma_j \mathbb{I}(S_h \in D_j) \right) \log p(S_h|\vec{\theta}) \quad (3)$$

I.e., for each segment, we need to sum all responsibilities of segmentations, where the segment is part of the segmentation. We can see that Equation 3 is of the same form as the log-likelihood L_1 for the inner loop with

$$u_j = \sum_{i=1}^{2^M} \gamma_i \mathbb{I}(S_j \in D_i).$$

It can therefore be optimized by an inner loop EM that optimizes the *weighted* log likelihood of all possible segments.

B. Inner EM loop

The inner EM loop is given by optimizing L_1, which is done by running a weighted EM on the segments for the mixture modell.

It is important to note that, in order to properly evaluate the log-likelihood, the segmentation is not allowed to change the number of data points for a single trajectory. Consequently, each segment will have a different number of data points.

C. Assumption

The observations feed into our framework are multiple endeffector trajectories of the same or different tasks. We assume that each trajectory is a combination of various segments, where each segment was produced by a ProMP representing a point-to-point movement. A point-to-point movement is defined as a trajectory which starts

and ends with zero velocity. Given the observations the framework now aims to find the set of ProMPs, which could have produced the observed trajectories.

D. Finding the Segments

By assuming point-to-point movements we can define the beginning or the end of a segment each time the trajectory hits zero velocity. Applying the framework to the real world means dealing with noise. Noisy observations might lead to false zero crossings in the velocity. We counteract this effect by smoothing the velocities with an averaging low-pass-filter and introducing an absolut threshold instead of zero.

Given the assumption that each segment was generated by a ProMP, it needs to be considered, that the same MP might have produced segments with different durations. Therefore a temporal scaling of each segment is necessary. So far we apply a simple linear scaling, while in the future non-linear scaling methods might be beneficial.

Since the order of the ProMPs is not fixed there will be similar segments which significantly differ in absolute values due to translational and rotational transformations. In order to still be able to find the underlying MPs the framework translates all segments such that they start at $[0,0,0]^T$. Subsequently a rotation around the z-axis is performed which aligns the vector from the start to the end point of the segment with the x-axis of the coordinate frame. Note that these transformations are necessary for the detection of similar segments, but at the same time the found ProMPs will have 0 variance at the start point and also 0 variance for the y-dimension of the end point. While these effects should not be problematic for the detection of segments in unseen trajectories, they first appear to be significat limitations for a subsequent sampling of segments from the found ProMPs.

However, considering the premise that the obsrvations are results of a sequence of MPs, the variance for the start point of each segment is not important, since it has to start where the previous segment ended. The lost variance along the y-axis can be circumvented by adding the angle of the rotation around the z-axis as and additional variable in the primitive.

Finally the segments are projected into the weightspace of the primitives, which has the benefit of reducing the dimensionality of the segments.

IV. EXPERIMENTS

In order to illustrate our approach we evaluated the algorithm on a one dimensional data set. The data set consist of multiple trajectories representing sine waves which always start with a value of zero and a negative tendency and end with a value of zero and a positive tendency. As Illustrated in Figure 1 there are multiple zero velocity cuts, marked in red, in such a trajectory, but the two green segments will always occur together. There fore it makes sense to assume that such trajectories could have been generated by only three underlying skills. The three skills found by our approach are shown. In Figure 2, where the skills are represented by random samples drawn form the primitives.

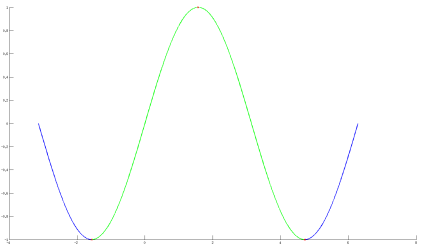

Fig. 1. Example of a training trajectory.

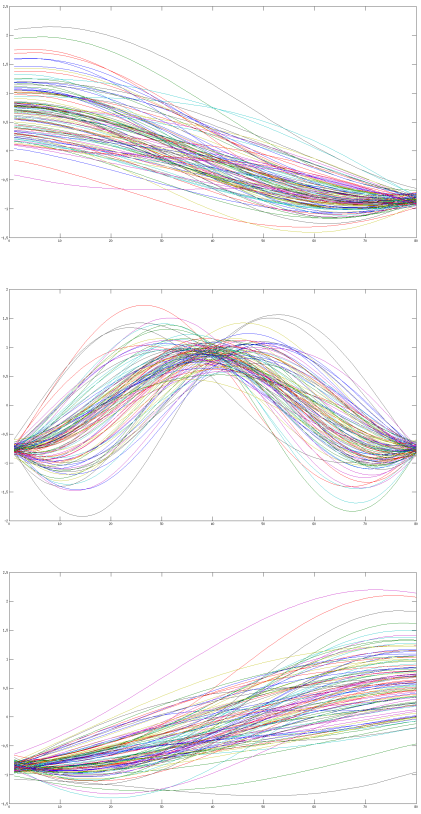

Fig. 2. Learned skills

V. CONCLUSION

Currently, for the sake of simplicity the presented method relied on the assumption that cuts are initially given by instants of zero velocity, which usually oversegments trajectories. However out method is able to compensate for this oversegmentation and finds skills which can produce trajectories with zero velocities.

ACKNOWLEDGMENT

The research leading to these results has received funding from the European Community's Seventh Framework Programme (FP7-ICT-2013-10) under grant agreement 610878 (3rdHand), and (FP7-ICT-2009-6) under grant agreement 270327 (ComPLACS).

References

[1] K. Mülling, J. Kober, and J. Peters, "Learning Table Tennis with a Mixture of Motor Primitives," in *IEEE/RAS International Conference and Humanoid Robotics*, pp. 411–416, 2010.

[2] A. Paraschos, C. Daniel, J. Peters, and G. Neumann, "Probabilistic movement primitives," in *Advances in Neural Information Processing Systems (NIPS), Cambridge, MA: MIT Press.*, 2013.

[3] M. Brand and V. Kettnaker, "Discovery and segmentation of activities in video," *Pattern Analysis and Machine Intelligence, IEEE Transactions on*, vol. 22, no. 8, pp. 844–851, 2000.

[4] W. Takano and Y. Nakamura, "Humanoid robot's autonomous acquisition of proto-symbols through motion segmentation," in *Humanoid Robots, 2006 6th IEEE-RAS International Conference on*, pp. 425–431, IEEE, 2006.

[5] D. Kulic, W. Takano, and Y. Nakamura, "Combining automated on-line segmentation and incremental clustering for whole body motions," in *Robotics and Automation, 2008. ICRA 2008. IEEE International Conference on*, pp. 2591–2598, IEEE, 2008.

[6] S. Schaal, J. Peters, J. Nakanishi, and A. Ijspeert, "Learning movement primitives," in *Robotics Research*, pp. 561–572, Springer, 2005.

[7] E. Meier, F.and Theodorou, F. Stulp, and S. Schaal, "Movement Segmentation using a Primitive Library," in *2011 IEEE/RSJ International Conference on Intelligent Robots and Systems*, pp. 3407–3412, 2011.

[8] S. Niekum, S. Chitta, B. Marthi, S. Osentoski, and A. G. Barto, "Incremental semantically grounded learning from demonstration," in *Robotics: Science and Systems*, vol. 9, 2013.

[9] J. Barbič, A. Safonova, J.-Y. Pan, C. Faloutsos, J. K. Hodgins, and N. S. Pollard, "Segmenting motion capture data into distinct behaviors," in *Proceedings of the 2004 Graphics Interface Conference*, pp. 185–194, Canadian Human-Computer Communications Society, 2004.

[10] A. Fod, M. J. Matarić, and O. C. Jenkins, "Automated derivation of primitives for movement classification," *Autonomous robots*, vol. 12, no. 1, pp. 39–54, 2002.