The AAAI-17 Workshop on
Human-Machine Collaborative Learning
WS-17-11

Making Robotic Sense of Incomplete Human Instructions in High-level Programming for Industrial Robotic Assembly

Maj Stenmark, Mathias Haage, Elin A. Topp, Jacek Malec
Dept of Computer Science, Faculty of Engineering
Lund University, Sweden
e-mail: {maj.stenmark, mathias.haage, elin_anna.topp, jacek.malec}@cs.lth.se

Abstract

In this paper we describe our NLP supported programming-by-demonstration approach to high-level robot programming that allows users to generate skills and robot program primitives for later refinement and re-use. Our ideas incorporate the identification of common user strategies (interaction patterns) in the programming process, which can be exploited to support a human user in establishing common ground with a robotic system. We have evaluated a prototype implementation of this approach in a user study and use observations from this study to define further research efforts, which we discuss in this short paper.

1 Introduction

Recently, significant research efforts have gone into making industrial robots easier to program, more flexible and ultimately more affordable for smaller companies. Especially when it comes to programming the type of flexible robot targeted for direct collaboration with a user or operator that has recently entered the market, e.g., ABB's YuMi, methods for programming by demonstration, specifically kinesthetic teaching, seem very suitable. However, those methods require in many cases large amounts of data for machine learning methods to be successful (Billard, Calinon, and Dillmann 2016), or produce only trajectory-based motions. To program a two-armed robot like the ABB YuMi to assemble workpieces using complex motions and specific forces in (ideally) a one-shot-demonstration approach, we need not only observation-based demonstrations but kinesthetic teaching that includes force profiles and allows for meaningful semantic annotations of specific movements and forces.

While humans are very well capable of establishing common ground in a conversation with other humans, the commonly used mechanisms for this obviously do not hold in a human-machine communication. If a human with limited understanding of a robot's understanding of the world tries to demonstrate and explain how to handle a certain task, the robot will need to understand the likely points of failure in this explanation and pose the right questions to make the human user understand what the robot needs to understand.

We propose to use observations from recent studies to find patterns in human behaviour with the help of classic ML approaches to support the interpretation of "one-shot" kinesthetic demonstrations in combination with iconic programming and annotations for which we are currently working on a prototype implementation. Our paper is organised as follows: We refer to previous approaches to interpreting behavioural patterns and to representing skills in section 2 and in section 3 we explain our prototype programming interface and the user study conducted with this prototype. Section 4 gives an overview over specific observations from the study that led to further development of the prototype as described in section 5. In section 6 we describe how we plan to work further on the development and integration of more advanced reasoning and learning capabilities of the system.

2 Interaction Patterns and Skills from Demonstrations

Based on our experiences from a series of user studies with mobile service robots, we developed the concept of Interaction Patterns, i.e., reoccurring patterns in the observable interaction (including preparation / positioning of the robot and general movements around the robot), that might correspond to the underlying meaning, conceptual understanding, or even intention the user assumes for her utterances, which would give us at least some means to describe related expectations (and detect violations of those). Our investigations focused so far on patterns that would allow to hypothesise about the conceptual (spatial) category (region, location, or object according to our framework for Human Augemented Mapping (Topp and Christensen 2008)) of an item presented to the robot beyond its label. E.g., while the utterance "this is the office" indicates that a room (or region) is presented, the user behaviour (pointing clearly) suggests that some specific location or large object (the door) is referred to. This should result in a mismatch of expected category and observations, which can further be used to trigger a request for clarification. We confirmed the applicability of this idea in a Bayesian-Network-based evaluation (Martí Carrillo and Topp 2016), and extended the concept of Interaction Patterns to Task Patterns, i.e., pattern-like structures in task sequences that can be categorised and analysed for deviations, again with the help of Bayesian Networks (Kleve 2016).

Further, we have recently developed a programming tool which allows us to generate skill representations according to our previous work from user demonstrations (kinesthetic teaching)(Stenmark and Topp 2016). For our skill representation, we assume two components. The first is a high-level step-by-step instruction how to achieve a goal. In our work, we represent the robot program using a finite state machine. The state machine can have other skills as nested states and the lowest semantically described step comprises atomic states called *primitives*. The primitives must have a mapping to executable code on the robot system, which is the second component of the skill (Stenmark 2015). With the programming tool, the user can provide information (annotations / labeling) of program steps and skill descriptions, similar to the labeling of an environment as assumed for the work with the Interaction Patterns described above.

We assume now that we can transfer the idea of observing common strategies and patterns in the interaction with the system into the realm of industrial robotics by analysing the material collected in the user study described in the following section to find respective structures and patterns in how the different users applied kinesthetic teaching and iconic programming.

3 Prototype System and Study Setup

The prototype design was developed after several case studies where the authors developed assembly applications on the ABB dual-arm robot YuMi. The user interaction is intended to simplify agile online robot programming. Mistakes should be easily corrected with programming and debug modes merged into a single screen to facilitate a quick program modification and execution loop. All available information about the program is retained for use even though it may not be used at the moment. As an example, each time a position is saved, all available information about the position is stored, such as the current joint values, the Cartesian tool position and, if a reference object is selected, relative positions, making it easy to switch between representations. This allows the operator to quickly create a program and work later on creating and applying abstractions such as object references to make the program easier to re-use and adapt.

The current graphical user interface (GUI, see Figure 1) consists of two instruction lanes covering the right half of the screen, each displaying an action sequence for one of the YuMi arms. Each action can be played individually using the play-arrow to the left of the name, the arm pose associated with the action can be updated with a single click, the parameters for each action can be edited, and an expandable icon contains a photo taken from a belly camera of the current operation. To the left, there is a selection of actions. By pressing an action the current position of the YuMi arm is recorded and the action is entered into the instruction lane. Below the action selection pane, there is a pane showing available coordinate systems, referred to as *objects* in the tool. A color marking connects the reference system to the actions in the instruction lane. Below the selection pane is a management pane currently containing functionality for managing objects and actions. It is possible to select groupings of actions in the instruction lane and switch reference object or create a procedure (referred to as skill in the tool). The very bottom features direct actions that will not be entered into an instruction lane.

The available actions were developed for the domain selected in the user study, featuring precise motions (*move*), motions with higher speed and less precision (*via*), and *contact* motions. A contact motion is parameterized as a motion towards a specified point until a certain estimated contact force is reached. The grippers were only controlled by opening and closing the fingers fully. A *locate* action (skill) was added to let the users identify object positions. User-created abstractions, skills, can be created and are added as actions with purple icons.

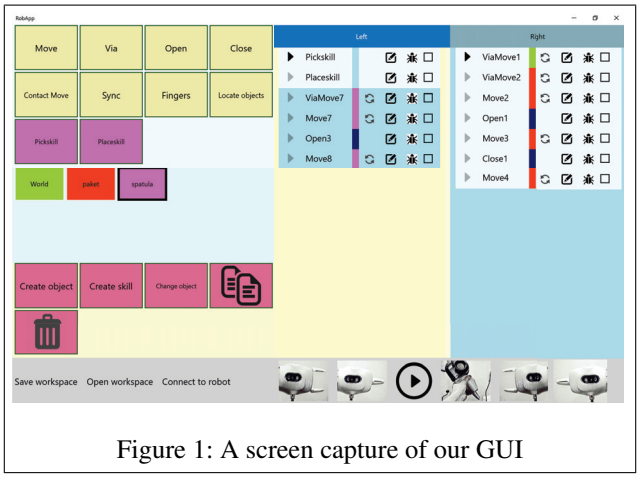

Figure 1: A screen capture of our GUI

Some advanced functionality that was identified as desirable in the initial case studies was not exposed to the users in the study. This included commands for synchronization between the robot arms, recording and replaying trajectories, specifying the finger position and gripping force and writing native code directly. Another problem identified in the case studies were the programming of dual-arm motions in contact situations, were voice commands were desired to save positions without releasing the robot. This functionality is now incorporated in a current implementation, see Section 5, but was not available for the user study described below.

User Study

We evaluated our tool through a user study with 21 non-expert, although technically or mathematically interested and experienced, users in a LEGO building task. Our subjects were given the task to program one arm on a dual-arm ABB YuMi with standard grippers to assemble different types of LEGO Duplo bricks using the robot's lead-through mechanism for kinesthetic teaching of poses in combination with our graphical tool for iconic programming. Figure 2 shows the setup for the study with the robot just finishing one step of the task.

The study participants spent about half an hour with the robot, we recorded their interaction with the robot on video and had them after their trial fill out a short questionnaire.

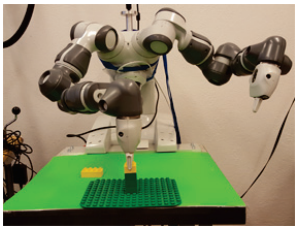

Figure 2: YuMi finishing the insertion of a LEGO piece.

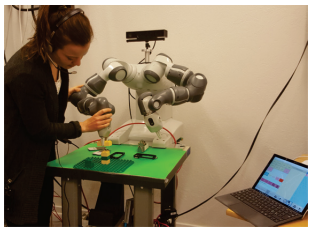

Figure 3: Task demonstration with speech and lead-through.

The overall task was carried out in two phases: The first phase was the same for all participants while the second phase divided the users evenly into three groups according to three different test conditions. During the first phase, the instructions were to program the robot arm to pick up a LEGO Duplo "2x2" piece (referred to as "small LEGO") and insert it on top of another small piece, which was already the top of a little "tower" (task step 1). This program should then be used to create a skill, i.e., a re-usable representation of a sequence of motion primitives. For the second phase, the participants were asked to program the robot arm to carry out the same task another three times, but with a "2x4" piece ("large LEGO"), which should be placed in different poses in relation to the "tower top". For these three steps, the different conditions defined the programming paradigm, i.e., the first group, Group A, was to re-use and refine their own previously created skill, Group B was to re-use an expert-made skill and Group C was a control group in which participants should program each step from scratch without re-use of previously created and saved skills.

We evaluated the trials for each participant according to a number of criteria and could conclude that almost all participants, i.e., 19 of 21, managed to handle the first step with at least partial success. Of those 19, five decided to terminate their trial after this first step, in most cases due to the time limit being (almost) reached, but 14 of 21 participants managed to understand the tool, the robot's movements, and the LEGO bricks' properties (including the challenge of robustly attaching them to each other) sufficiently well to complete at least two steps of the overall task within 30 minutes.

4 Observations from the study

In our study, we also made a number of observations (or received respective explicit comments) regarding the challenges the subjects experienced, which we summarize in the following (most frequently occurring ones first).

- *Precise positioning and fixating the LEGO piece firmly* Many subjects mentioned the placement of the piece as a challenge. Whether they were successful or not seemed to depend a lot on whether the respective subject could determine a good strategy from the LEGO properties. The expert–made skill provided for group B followed a good strategy, and worked very robustly, which helped the group B subjects significantly.
- *Different reference coordinate frames and switching between them* Several subjects struggled to remember when to use which coordinate system.
- *Understanding all functionalities of the tool in the limited time frame, handling the interface* This suggests that we might even have had better results, had the subjects been given more time to "play" with the tool and more expert-made skills to start with.
- *Pose recording vs trajectory recording, understanding robot movement* Some of the subjects struggled to understand the concept of specifying positions to move to rather than trajectories to move along, i.e., they often forgot to integrate so called explicit "via points" in their programs which resulted in the robot moving along seemingly odd paths from one point to another.
- *Robustness, compensating for the LEGO piece sometimes slipping from the gripper* As we did not make use of specifically designed grippers for LEGO-bricks, this was obviously a problem in some cases. It could be compensated for only through a good picking and placing strategy that would be robust to the slipping.

We see several of these challenges pointed out by our subjects as obvious points to work with in the future. We are also aware that these points are not unexpected in any way, but we see them as a concrete confirmation of assumptions we had regarding where and which support for the user by the system would be most suitable to support the user in establishing common ground about the work space and task at hand together with the system. We also assume that we can analyse the material from the study regarding similar patterns as those applied in our work with Interaction Patterns.

5 Ongoing Implementation

In ongoing implementation work on our programming tool, missing advanced features are added, e.g., actions for synchronization and writing native code. A dialogue box for interaction using natural language is added as well, here actions can be dictated using off-the-shelf dictating tools and the commands are evaluated continuously using the online services from previous work. Then the operator does not have to reach the touch interface in order to add instructions as shown in Figure 3. The user can also demonstrate dual-arm contact situations without releasing the hold of the robot. As an example, this enables teaching in situations were the user needs to counteract external forces that would move the robot arm, e.g., caused by the weight of an item in the gripper.

In addition to specifying action sequences, the user can also create skills and objects using voice, e.g., using the

commands *Create a large lego (object)* and *Save the program as a pick skill*. The user can add single instructions by naming them directly (e.g., *Add a via point*) and as soon as a new object or skill is created, the vocabulary is extended to incorporate their names, *Pick the large lego.* etc.

Ongoing work also involves recording, modifying and replaying force-controlled trajectories, as well as interpreting text providing additional information not easily extracted through lead-through.

6 Future ideas

The user study was indented to evaluate difficulties when programming the robot from scratch and to investigate whether or not reusable abstractions simplified the programming. Overall, the test subjects did not have any problems understanding and reusing the expert-made skill when they were allowed to execute it. It was, however, challenging for some participants to get started, hence, a collaborative robot system should provide good template programs and suggestions for the user to lower the initial threshold.

Picking and placing are obvious examples of such template programs, also offered as precoded *Apps* in the Franka robot recently launched by Emika[1]. It is, however, necessary to adapt such general skills to specific tasks, particularly in our targeted assembly scenarios. Our future work involves thus data collection and storing of basic movement sequences as skills on the respective knowledge server to build an extendable library of skill representations, but also investigations of how to best adapt existing skills when reused and how to support the user in this process. Here, we plan to make use of our previously mentioned work regarding Task Patterns (Kleve 2016), i.e., we plan to use Bayesian techniques to find patterns in the instruction sequences of existing skill representations. The system can then suggest the most common action given the current program sequence, e.g., suggesting adding a via point to avoid collision, using a contact motion instead of a fine point motion or changing the reference coordinate system to another object etc.

Using high-level cues from the language used for annotation, the user's intent can be identified even earlier, e.g., if the user starts to add *screws* and *nuts* to the world description, the robot system should offer existing screwing skills from the database as basis for re-use and refinement, since those skills used similar object names, that can be found through synonym resolution offered by our NLP system (Stenmark and Nugues 2013).

If, however, no suitable skill representation to extend or adapt can be found and a new skill has to be generated from scratch, we plan to support the user in annotating specific instructions (motions), by applying the Interaction Pattern idea (Martí Carrillo and Topp 2016) to find patterns in observations from multiple demonstrations and annotations of similar movements, so that it later becomes possible to use the stored movement for matching of task sequences as described above. While our user study material is already a good place to start with such observations, the planned integration of force estimation techniques (as proposed earlier in closely related work (Linderoth et al. 2013)) to obtain force profiles and trajectory representations based on dynamic movement primitives (DMPs, subject to current work, based on approaches earlier presented by, e.g., Ijspeert *et al.*, Kober *et al.*, and Manschitz *et al.* (Ijspeert, Nakanishi, and Schaal 2002; Kober, Gienger, and Steil 2015; Manschitz et al. 2014)) rather than position-based motions will require further studies to gather more insights into how potential users actually demonstrate and annotate specific movements.

We believe that those suggested extensions will enable technically skilled users without extensive robot programming experience to program, or rather teach kinesthetically in combination with natural language annotations, industrial robots in a very intuitive manner. At the same time, the robot system's support will teach the user about the system, which means that we can establish a form of human-robot symbiosis through this type of collaborative learning.

[1] http://www.franka.de (last accessed Dec 12, 2016)

7 Acknowledgements

The research leading to these results has received funding from the European Community's Framework Programme Horizon 2020 under grant agreement No 644938 SARAFun.

References

Billard, A.; Calinon, S.; and Dillmann, R. 2016. Learning from humans. In *Springer Handbook of Robotics, Chapter 74*. Springer. 1995–2014.

Ijspeert, A. J.; Nakanishi, J.; and Schaal, S. 2002. Learning attractor landscapes for learning motor primitives. Technical report.

Kleve, B. 2016. Detecting anomalies when guiding an industrial robot through tasks. Master's thesis, Lund University, Dept of Computer Science.

Kober, J.; Gienger, M.; and Steil, J. J. 2015. Learning movement primitives for force interaction tasks. In *Robotics and Automation (ICRA), 2015 IEEE International Conference on*, 3192–3199. IEEE.

Linderoth, M.; Stolt, A.; Robertsson, A.; and Johansson, R. 2013. Robotic force estimation using motor torques and modeling of low velocity friction disturbances. In *2013 IEEE/RSJ International Conference on Intelligent Robots and Systems*.

Manschitz, S.; Kober, J.; Gienger, M.; and Peters, J. 2014. Learning to sequence movement primitives from demonstrations. In *2014 IEEE/RSJ International Conference on Intelligent Robots and Systems*, 4414–4421.

Martí Carrillo, F., and Topp, E. 2016. Interaction and Task Patterns in Symbiotic, Mixed-Initiative Interaction. In *In Proceedings of the AAAI-16 Workshop on Symbiotic Cognitive Systems*.

Stenmark, M., and Nugues, P. 2013. Natural language programming of industrial robots. In *Robotics (ISR), 2013 44th International Symposium on*, 1–5. IEEE.

Stenmark, M., and Topp, E. 2016. From demonstrations to skills for high-level programming of industrial robots. In *Proceedings of AAAI Fall Symposium on AI for HRI*.

Stenmark, M. 2015. *Instructing Industrial Robots using High-Level Task Descriptions*. Licentiate Thesis, Lund University, Faculty of Engineering.

Topp, E., and Christensen, H. 2008. Detecting structural ambiguities and transitions during a guided tour. In *Proceedings of the IEEE International Conference on Robotics and Automation (ICRA), Pasadena, CA, USA*.