The GiftWrapper: Programming a Dual-Arm Robot With Lead-through

Stenmark, Maj; Stolt, Andreas; Topp, Elin A.; Haage, Mathias; Robertsson, Anders; Nilsson, Klas; Johansson, Rolf

2016

The GiftWrapper: Programming a Dual-Arm Robot with Lead-Through

Maj Stenmark[1], Andreas Stolt[2], Elin A. Topp[1], Mathias Haage[3],
Anders Robertsson[4], Klas Nilsson[3], and Rolf Johansson[5]

I. INTRODUCTION

In previous work we presented a force-controlled lead-through mechanism[6] that has made its way into the recently launched dual-arm ABB robot YuMi [2], [4]. This mechanism supports close physical interaction with an industrial robot arm during setup and programming. With this paper we intend to share our case-study experiences from programming an ABB YuMi with standard equipment (i.e., basic electric grippers, no external sensors) as shown in Fig. 1 to wrap gift boxes, using and thus evaluating mainly the lead-through programming functionality in combination with the respective "YuMi Online" app. We would consider the programming team as semi-experts in terms of industrial robot programming, as we are quite familiar with the robot and its tools from a research perspective, but have not too much experience in programming robots for real shop-floor conditions in the role of system integrators. The work was commissioned as a Christmas commercial for a consumer electronics retailer in Sweden, with a robot offering gift wrapping services in the shop. The customers would place their purchased items in a cardboard box of previously specified and thus standardized measurements, a sheet of paper would be manually placed in a fixture in front of the robot and then the box was to be put on top of the paper in front of the robot. The robot would wrap the gift as described below and finish by stamping the packet and decorating it with a gift bow chosen and handed over by the currently served customer. Hence, the application was entirely intended as part of an entertaining PR campaign and not to optimize gift wrapping. Consequently, the circumstances (e.g., the allotted time frame, short setup and tear-down cycles) for the development and deployment of the application turned out to be somewhat nontraditional, forcing us to find equally nontraditional solutions. Also, we made use of the robot's specific physical properties, including the fact that the seventh redundant degree of freedom in each arm could be utilized to achieve unusual, but effective joint configurations. In the following, we will describe the circumstances for the application, including the gift-wrapping process as such, and then discuss our experiences from the programming process. Here, we will point out open issues that we identified regarding the usability and support of the programming tools for the dual-arm and 2x7 degrees of freedom properties of the robot, focusing on the physical, lead-through supported kinesthetic teaching.

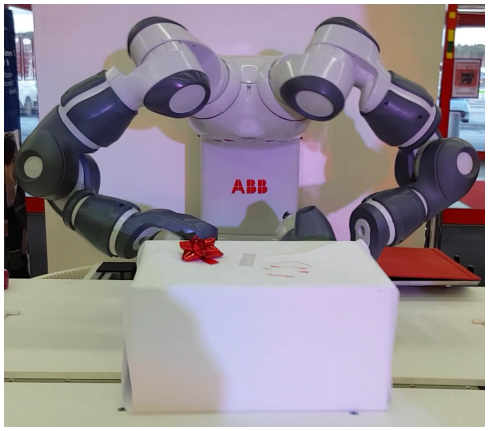

Fig. 1. The gift-wrapping setup.

II. SYSTEM SETUP AND TASK

In this section we will explain the task our application had to handle together with the challenges we encountered. We will also explain a number of choices we had to make during this process that affected the resulting application, many of which were due to the time constraints we faced both for the programming and for the setup on location prior to each deployment.

A. General challenges and issues

The general setup conditions held a number of challenges already. The time frame available for feasibility analysis, design, and implementation of the application was limited to one month. As only one physical instance of the YuMi robot was to be used throughout the month-long campaign covering 18 sites throughout Sweden, each new placement required (almost) daily on-location setup and tear down, which posed additional requirements in terms of setup times and robustness regarding the transport by truck in Swedish winter conditions. These time and robustness constraints (both for the programming but also for the daily setup)

[1] Dept. of Computer Science, Lund University, Lund, Sweden. {Maj.Stenmark, Elin_Anna.Topp}@cs.lth.se
[2] Cognibotics AB, Lund, Sweden. Andreas.Stolt@cognibotics.com
[3] Cognibotics AB and Dept. of Computer Science, Lund University, Lund, Sweden. {Mathias.Haage, Klas}@cognibotics.com
[4] Cognibotics AB and Dept. of Automatic Control; LCCC Linnaeus Center and the eLLIIT Excellence Center at Lund University, Lund, Sweden. Anders.Robertsson@control.lth.se
[5] Dept. of Automatic Control; LCCC Linnaeus Center and eLLIIT Excellence Center at Lund University, Lund, Sweden. Rolf.Johansson@control.lth.se
[6] Finalist for the 2016 euRobotics TechTransfer Award, http://www.erf2016.eu

were the main reason for us not to deploy external sensors (cameras, force-sensitive fixtures, etc.) for the process, as they would have added considerable complexity, given the available programming and system environment.

Additionally, the application as such was challenging due to different types of uncertainties that needed to be handled. First, the weight of the boxes varied and despite careful folding the boxes could slip several centimeters during the wrapping process which required frequent repositioning of the box. Secondly, the boxes could be filled to varying degrees, resulting in the box being more or less rigid (or even bulging outward), which generated some uncertainty regarding the exact actual size of the box. Thirdly, paper is an organic material that folds unexpectedly when deformed and tears when pulled too hard, which in combination with the slight variations in box shape and rigidness generated another level of uncertainty.

B. Lead-through programming and work flow

The above mentioned time and robustness constraints resulted in our decision to go past the offline programming and simulation phase usually assumed and use only the online programming techniques both for programming, testing and debugging. Also, we decided to develop a purely position-based program, as we did not have the time to adapt the available force / torque sensory system to our needs. We settled for very basic error handling capabilities – mainly, the execution would stop in case of a problem being encountered, and could only be resumed after an operator corrected the situation and pressed the resume button. The resulting application makes use of all seven degrees of freedom available in each of the arms, hence, it controls all 14 degrees of freedom. It also makes somewhat unconventional use of all parts of the robot arms, e.g., pushing items using the side of its wrist or fixating a workpiece by pressing down the elbow while performing a folding movement with the same arm. This actually imposed constraints on the box size that was possible to handle, as the robot's torso is rigid and the height of the elbow relative to the table top cannot be arbitrarily adjusted.

C. The resulting application

The resulting packaging process is shown in Figs. 2–4 and a video can be found online [3]. The application required an on-location setup time within the limits of approximately 1–2 hours and a complete run wrapping one gift took 2 min and 40 s, proceeding as described in the following. Initially the robot kept the arms in a perpendicular position and used the lower parts of the arms to push the box into a predefined start position. While fixating the box using the padded part of its right wrist, the robot lifted the paper around the box with the left gripper, making use of an incision in the table that let one of the fingers slide under the paper and grip the edge of the sheet. Now, while the right gripper was used to fixate the paper edge on top of the box, the left hand picked up a piece of pre-cut tape. The pieces of tape had a tendency to stick to the fingers, hence the taping procedure needed to first attach the tape to the paper and then slide back and forth to attach it to the top of the box. After attaching the right side in a similar manner, the package was rotated by 90° and the position was adjusted using the palms and lower parts of the arm, as the fingers turned out being too sensitive for the rather high torques needed to reposition the box. Now, both sides of the package were taken care of running the following process twice in a mirrored way. Using the grippers, the paper was pinched into the side flaps typically shaped in a gift wrapping process (Fig. 3C). These side flaps were folded around the box using two plastic spatulas depicted in Fig. 3(R). One arm fixated the box using three contact points, the elbow, the finger tip and the spatula tip, while the other arm used the other spatula to fold and straighten the side flap. While one arm fixated (with the spatula) both side flaps to the side of the box, the second arm lifted the lower flap (making use of the incision in the table top) that resulted from folding the side flaps inward, took over the fixation and the other arm picked tape which was then placed and finally firmly attached with the second arm fixating the box from the opposite side. The packaging was ended with a stamp and a gift bow. Finally, the finished gift was pushed towards the customer using the soft pads on the wrists.

III. POSITION BASED ONLINE PROGRAMMING

In this section we will describe the development and programming process for the gift wrapping application, focusing on the opportunities and challenges we were facing regarding robot and programming tool properties as well as task related constraints.

As seen from the task perspective, one obvious advantage of the robot is the redundancy in degrees of freedom, which we exploited fully. Another advantage is the inherent safety system, which allowed us to use the physical robot for programming, debugging and testing, as well as we could be sure being on the safe side regarding the customers who were expected to be around (and part of the process) during execution. This means, we would probably not have been able to achieve a result as we did in the given time frame with a robotic system not offering these properties. During the programming and also during execution, however, we faced some challenges in using the provided tools, which we will explain in the following.

A. Physical programming

The robot flexpendant comes with a joystick and the joints, and target coordinates or the elbow angle can be adjusted separately. In order to teach the positions, it is, however, faster to use lead-through programming. In this application, the robot was in contact with the box with the purpose of applying small forces to fixate the box or tape the paper. When using lead-through programming in such a contact situation, the robot arm had to be held in place by the user while the position was saved using the GUI. With multiple contact points, each contact point had to be fixated by the user and often two hands were needed to position the robot correctly. Actually, pressing the button to save the position

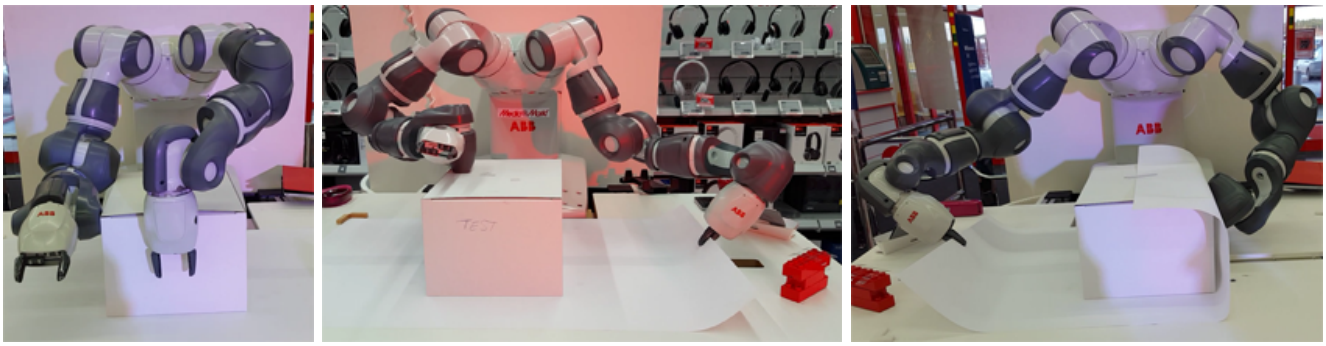

Fig. 2. L: The robot uses multiple contact points along the lower arms and the tool to push the box into its initial position; C: The table has an incision to let the robot fingers grip the paper without creasing it; R: One hand fixates the box while the other one folds and tapes the side.

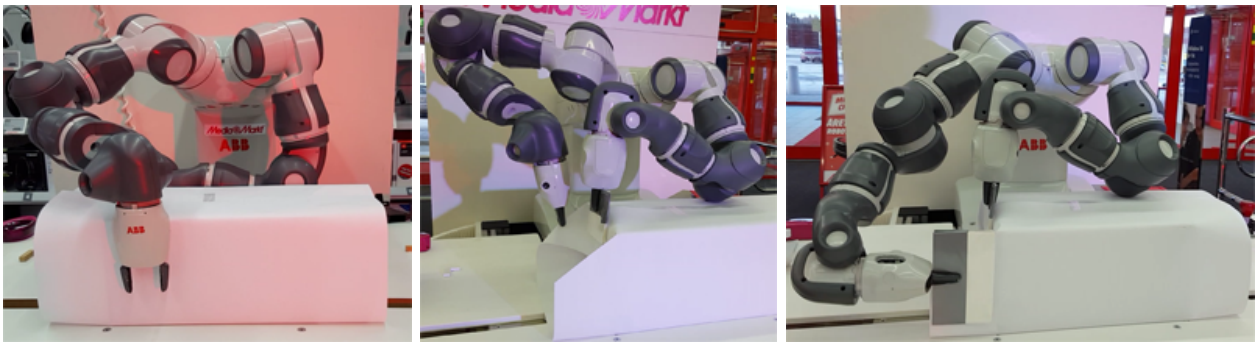

Fig. 3. L: The box is rotated using the side of the grippers and one of the arms as a fixture; C: The paper corners are folded using the fingers; R: The robot uses pieces of cardboard to fold and straighten the paper around the corners of the box.

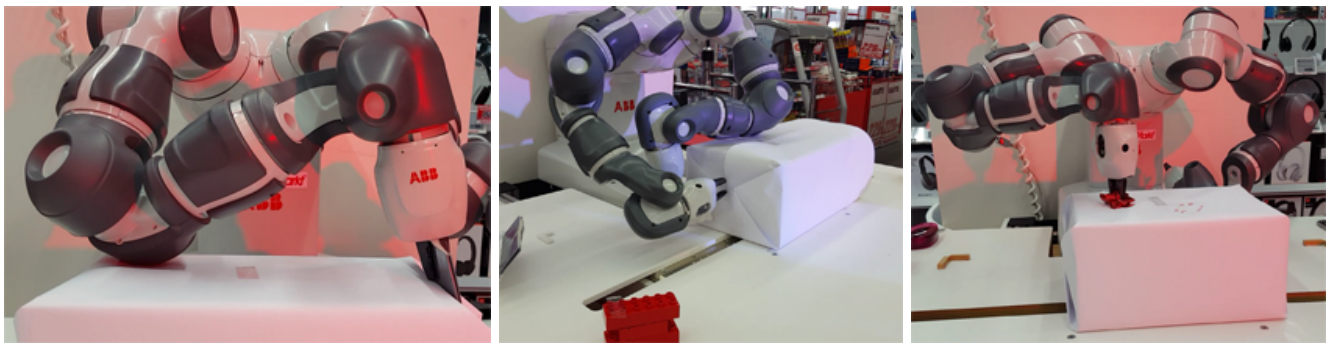

Fig. 4. L: The elbow fixates the box while the finger tip push down the paper and the cardboard piece folds the side; C: When taping the side, one hand holds the paper flaps in place while the other attaches the tape. The other hand will then support the box while the taping hand can rub the tape to secure it; R: The packaging process ends when the customer hands the robot a bow and the robot attaches it to the wrapped gift.

may thus become a physical challenge. The same shortage of appendage arises when releasing the brake of an arm to move it, since the brake button needs to be continuously pressed. Dual-arm lead-through programming of contact situations required three operators or an iterative approach.

B. RAPID prototyping

Since the physical system was used to prototype and test different solutions, the cycle time for testing became a bottleneck. This was partly due to the application itself and partly in the system software. The application added debugging complexity since complete gift wrapping can be seen as one single long assembly operation. Execution of the task itself deformed the paper and during the debugging process, large parts of the program had to be executed in sequence to take the paper folding into account. As soon as some error was detected or a change was introduced, it had to be corrected and debugged before the subsequent parts of the program could be tested. Many of the positions were dependent, so if a grip position changed so would not only influence the following place positions but also the paper could fold differently in the end of the process. The complete cycle to update and test a program was therefore very long. Also, since the programming and testing was carried out on the same hardware, the programming process could not be parallelized. In a typical use case, an application can be tested using simulation software, but paper was not

available in the simulation tool (and would have been too time-consuming to model and add), so only few parts were optimized using offline tools. Other challenges were created by the safety system and the fact that two robot tasks (both arms' controlled concurrently) had to be debugged together (see more on this below). In a more typical scenario, the spatial coordinates (Cartesian coordinates + rotation) of the tool are the relevant features of each position and when objects are moved in the environment, the target positions can easily be offset.

In our case, the relevant positions were often implicit; e.g., the wrist or the elbow was in contact with the box but the targets were still expressed using the tool point. This meant that even small changes in object position would require reprogramming.

C. Safety first

The tools assumed the same safety requirements as for a classic, unsafe robot, hence the execution and debugging modes required the user to specifically change controller states between Auto, Manual and Debug modes or different execution modes (step-over/continuous), and the allowed operations were restricted (e.g., debugging is not allowed in Auto mode). For each debugging operation, there was additional dialogue or required steps that the programmer had to take in order to guarantee safe execution. Failing to take these steps would throw an exception, which in turn had to be handled by the user before (s)he could attempt to carry out the originally intended action in a correct manner. Many of the safety features implemented in the system software seem redundant when the hardware itself is safe, thus counteracting our intentions.

D. Dual-arm robots

Each robot arm has a single point of execution (a program pointer), and during debugging, the user will have to manually move the position of the program pointer back and forth to the correct line of code. As the current system design expects each robot arm to run its own single threaded program, synchronization between arms is done using synchronization points or by specifying pairs of synchronized concurrent motions. During debugging, both program pointers had to be monitored so that the arms would not get out of sync. Synchronized motions often came in sequences and could only be executed in the correct pairs using forward-execution. Therefore, it required a careful procedure to update a position or recover from a collision in synchronized steps.

E. Contact point or collision?

The robot has a collision detection system that stops the robot when too large a torque is needed to reach a target position (assuming a collision with a human co-worker). This would render a conflict when, e.g., a somewhat heavier box had to be pushed into place and held with parts of the arms not being the tool center point, or when a box was very crammed and hence bulging, so that the contact point for taping was reached earlier than expected. We did, however, not have enough time to evaluate the applicability of a specific parameter to resolve this issue. Another solution to this problem would have been a more specific estimation of contact forces for each arm segment and resulting adaptation of the position for taping or turning. This is possible given access to the low-level control signals to evaluate the motor currents as we showed with a research prototype of the robot [1]. The methods for estimating forces in the commercial YuMi are currently under further development.

IV. CONCLUSIONS

From our experiences with the GiftWrapper application we can state that the robot properties and tools for online programming, specifically the lead-through programming, made it possible for us to achieve a reasonable result under rather pressed conditions, especially considering our semi-expertise in terms of production line compatible robot programming. Nevertheless, we faced a number of challenges in the programming process, indicating that the currently provided support tools and programming paradigms do not yet fully support the full exploitation of the robot's properties, and the programming requires still some level of familiarity with the system internals, to achieve the necessary level of robustness and stability. Although the robot supports physical interactions during programming and setup, but also during execution due to its safety system, the interface for instructing the robot is not yet as intuitively and easily to apply as these physical interaction properties suggest. Last but not least, although seemingly a toy problem, the task of wrapping gifts with a general purpose robot turned out to be quite challenging, and it yields many properties of applications we assume to become target applications for this type of robot, e.g., assembly of flexible parts or general uncertainties regarding part properties.

ACKNOWLEDGMENT

The GiftWrapper application was initially suggested for (and then supported by) MediaMarkt Sweden to promote sales before Christmas, by the marketing company Lowe Brindfors in Stockholm. ABB Robotics provided a YuMi robot on loan for the work, plus valuable support for the specialties of YuMi programming. The application was developed and exhibited by Cognibotics AB as a system integrator. The authors acknowledge the persons involved in those companies for valuable comments and discussions.

REFERENCES

[1] Magnus Linderoth, Andreas Stolt, Anders Robertsson, and Rolf Johansson. Robotic force estimation using motor torques and modeling of low velocity friction disturbances. In *2013 IEEE/RSJ International Conference on Intelligent Robots and Systems*, Tokyo, Japan, November 2013.

[2] Andreas Stolt, Fredrik Bagge Carlson, Mahdi Ghazaei, Ivan Lundberg, Anders Robertsson, and Rolf Johansson. Sensorless friction-compensated passive lead-through programming for industrial robots. In *Intelligent Robots and Systems (IROS), 2015 IEEE/RSJ International Conference on*, 2015.

[3] Andreas Stolt and Maj Stenmark. YuMi wraps Christmas gift. https://youtu.be/ASEtz2M1RiY, 2016. Accessed: 2016-01-20.

[4] ABB YuMi. http://new.abb.com/products/robotics/yumi. Accessed: 2016-03-18.