Knowledge acquisition through human demonstration for industrial robotic assembly

Timotej Gašpar, Miha Deniša and Aleš Ude

Jožef Stefan Institute, Jamova Cesta 39, 1000 Ljubljana, Slovenia
{timotej.gaspar, miha.denisa, ales.ude}@ijs.si

Abstract. With the ambition to introduce robots into assembly lines, not suitable for classical automation, the ease of robot programming is becoming more significant then ever. This paper proposes using various approaches for gaining knowledge from human demonstrations. This knowledge is applied to perform assembly tasks in a industrial robotic cell. Real industrial use case is used for evaluation of proposed approaches. It shows their viability and presents different scenarios which call for different approaches of learning and execution of assembly tasks and its subsets.

Keywords: Learning by Demonstration, Motion Generation, Reconfigurable Robotics, Industry 4.0

1 Introduction

As more and more large enterprises adopt robots to automate their production lines, classical automation is reaching its saturation point. In order to further increase the productivity of the industrial sector, automation needs to expand into previously untapped areas. Classical automation is challenging to introduce in small to medium enterprises (SMEs). Main hurdles with automation of production lines in SMEs are: small batches of products, variability of products or a family of products, no experts on site, and funding limits for automation investments. Another aspect of robotics, which is being introduced into industrial environments, is collaborative robotics. To reduce the need for experts and make collaborative robots more viable in industrial setting, the ease of (re)programming a robot for a new task is paramount. To ensure the ease of learning assembly tasks, human demonstration can be used to (re)program the robot. In addition, the need for on-site experts is removed, as relatively small amount of expertise is needed for human demonstration.

A popular approach for acquiring new sensorimotor knowledge is programming by demonstration (PbD) [1–3], where a robot observes a human performing a task. These movements can be observed using a marker based system [4, 5], vision systems with RGB-D [6] or stereo cameras [7]. Alternatively, the human can physically move and guide the robot, using the approach called kinesthetic guidance [8, 9]. If the movement is captured through kinesthetic guidance, it is

already adapted to the robot dynamics and kinematics. The movements, kinesthetically captured in joint space, also preserve the configuration of the robot. Beside classic approaches for point to point motion generation, we also propose using Dynamical Movement Primitives (DMPs) [10–12] to encode a full demonstrated trajectory and store it as a set of parameters.

In the next section we present multiple approaches to extract knowledge from human demonstrations. In addition to kinesthetic guidance, we propose to use an alternative method to ease the process of assembly demonstration. In Section 3 we present various robot trajectory generators. In order to transfer gained knowledge to the robot, several motion generation schemes are used. Section 4 tackles evaluation of proposed approaches in a real industrial use case. A reconfigurable robotic work cell, described in our previous work [13], is used as an evaluation environment. The evaluation serves as a proof of concept and shows the feasibility of proposed approaches.

2 Data acquisition

This section presents proposed approaches for data acquisition. While kinesthetic guidance and joystick control can be used for moving the robot, custom button interface is introduced to ease the process of demonstration.

2.1 Kinesthetic guidance

Traditional robot programming approaches require the user to guide and program the robot with its controlling interface. By handling either a touchscreen or a joystick, the user moves the robot to a desired position and defines the type of motion to that configuration. This method turns out to be relatively slow and requires a certain proficiency in robot handling. For these exact reasons modern robots allow for kinesthetic guidance out of the box. Kinesthetic guidance proves to be a considerably more intuitive approach to move the robot during the programming phase.

Our setup consisted of two Universal Robot UR10 arms where kinesthetic guidance is by default implemented with the so called "Gravity Compensation" mode. This kind of control can be achieved by taking the dynamic model of a robot:

$$\boldsymbol{\tau_u} = \widehat{\mathbf{H}}(\mathbf{q})\mathbf{u} + \widehat{\mathbf{C}}(\mathbf{q}, \dot{\mathbf{q}})\dot{\mathbf{q}} + \widehat{\mathbf{B}}_f(\dot{\mathbf{q}}) + \widehat{\mathbf{g}}(\mathbf{q}) \tag{1}$$

$\boldsymbol{\tau}_u$ represents the controlled torques in the robot joints, $\mathbf{u}$ is the desired control signal to be regulated, $\mathbf{q}$ is the vector of robot joint angles, while $\widehat{\mathbf{H}}(\mathbf{q})$, $\widehat{\mathbf{C}}(\mathbf{q}, \dot{\mathbf{q}})$, $\widehat{\mathbf{B}}_f(\dot{\mathbf{q}})$ and $\widehat{\mathbf{g}}(\mathbf{q})$ represent the estimated inertia, Coriolis, friction and gravity models, respectively. To achieve kinesthetic guidance we set $\mathbf{u}$ to be null. Additionally, because of relatively slow robot motions, we can neglect the $\widehat{\mathbf{C}}(\mathbf{q}, \dot{\mathbf{q}})$ term. This results in the following control strategy

$$\boldsymbol{\tau_u} = \widehat{\mathbf{B}}_f(\dot{\mathbf{q}}) + \widehat{\mathbf{g}}(\mathbf{q}) \tag{2}$$

This mode essentially compensates the effects of gravity by estimating the necessary torques in the robot joints with the dynamic model. By applying external torque $\boldsymbol{\tau}_e$ the user can guide the robot kinesthetically without any additional sensing equipment.

2.2 Wireless joystick control

Due to the UR10 robot lack of sensing equipment in the robot joints, the dynamic model control is based on controlling the torque in the joints by regulating the electrical current to the servo motors. This results in a poor compensation of the friction in the joints which consequently requires significant physical effort to move the robot. It is therefore very challenging to move the robot smoothly and precisely, which is even more notable when kinesthetic teaching of precise trajectory is needed.

To cope with this drawback, we developed an interface that allows the control of the robot in Cartesian space with a wireless joystick. During the development of the joystick control interface we strove to provide an intuitive experience.

The joystick used was a standard Xbox 360 controller, which is widely supported on multiple operating systems.

Intuitive control was achieved by mapping the controller's analogue stick offsets from its default position to Cartesian space velocities. The user can switch from controlling translational velocities to rotational velocities. The user can move the robot with different velocity settings by pressing a button. This ensures the user to have the option between faster and less precise or slower and more precise motions.

Fig. 1: A consumer grade joystick interface that we used to perform precise motions of the robot in Cartesian space.

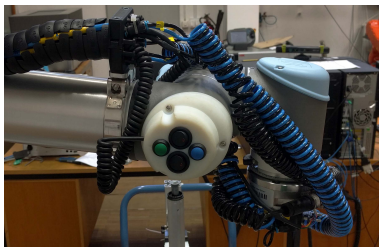

Fig. 2: Custom 3-D printer cover serving as an interface for assembly task teaching.

2.3 Custom button interface

To further increase intuitiveness and accelerate data acquisition we developed an interface with multiple programmable buttons. The interface replaces one of the

covers on the robot and it houses 2 buttons and 2 switches with LEDs (Figure 2). The latter provide a visual reference to the switch's state. The cover was developed to provide buttons that can be programmed for different actions dependent on the needs. In some cases the switches toggle gravity compensation, in others they make an entry of the current joints configuration, pose configuration or whole trajectory into the system database. This allows the user to uninterruptedly record the data needed for the assembly process.

3 Robot Trajectory Generation

Following the acquisition of desired poses and trajectories relevant for an assembly process, the next step is to generate appropriate robot movements. We can generate trajectories between two known configurations or replay a whole trajectory previously demonstrated by kinesthetic guidance.

3.1 Point to point movements

In cases where the the motion performed by the robot is relatively simple i.e. the shortest path from one configuration (joint or Cartesian space) to another, we can use simple trajectory generation algorithms. In our experiments we extensively used two such algorithms:

- *Joint space motion with trapezoidal speed profile*
  This trajectory generation algorithm will generate a trajectory in joint space for the robot to reach a desired final configuration from the initial configuration. In this type of movement, the robot reaches the specified desired maximum speed with constant acceleration.
- *Cartesian space straight line & SLERP trajectories following the minimum jerk velocity profile*
  This trajectory generation algorithm will generate a straight line trajectory in Cartesian space with zero initial and final velocities and accelerations in a manner that minimises the jerk throughout the motion. This algorithm is particularly useful when the path from the initial to the final pose of the end effector should follow a straight line. It the context of industrial assembly, we used this algorithm when approaching objects, picking them up, inserting them into a workpiece, rotating them into the workpiece, etc.

In most cases, a sequence of point to point movements can be defined relative to a single point. Instead of demonstrating multiple points that define point to point movements, a single point can be recorded. The other needed points are defined relative to this one.

3.2 Free from movement representation

With point to point movements we can carry out most steps in an assembly process. However, sometimes it can happen that, due to the robot cell configuration, more point to point movements have to be chained just to avoid possible

collisions. This kind of programming requires the user to have some experience in terms of what poses to chose in order to avoid collisions. To offer the user a more intuitive approach for for collision free motions we implemented an interface to record whole demonstrated trajectories.

Through the use of kinesthetic guidance the user can guide the robot alongside a complex trajectory while avoiding collisions with either the periphery of the cell or the robot structure itself. In order to minimise and parametrise some aspects of the recorded trajectories we encode them using *Dynamic Movement Primitives* [11]. The representation also provides a method of temporal scaling which makes it possible to increase or decrease the speed of the execution of a previously recorded trajectory.

4 Motion strategy evaluation through an industrial use case

The industrial use case, used for evaluation, is a sub set of a glass mounting gripper assembly. It consists of several pick-and-place tasks, (inverse) peg-in-hole actions, positioning of screws, screwing, etc. It was implemented in a modular and reconfigurable work cell [13] consisting of two UR10 robots, tool exchange system, modular beam system, etc. While the robots are controlled by a real-time system using *Matlab Simulink Real-Time (SLRT)* platform [14], the top software layer consists of a *Robot Operating System (ROS)* framework [15], which handles the communication between the modules of the cell. While all the sub-tasks of the studied use case use the proposed approaches, just three of them are presented in this section. They represent different aspects of the whole assembly procedure and cover the full spectrum of all needed tasks.

As described in Section 2, kinesthetic guidance and joystick control were used to move the robot during human demonstrations. Kinesthetic guidance was used to record trajectories in joint space, as well as single robot positions in joint or task space. The custom button interface was used to ease the demonstration process, by using its signals as record events. On the other hand, joystick control was used to record just single poses. Recording whole demonstrated trajectories through joystick control is not beneficial, as point to point generated motions exhibit the same behaviour as moving the robot in task space using the proposed joystick control.

The data acquired was stored in a *MongoDB* server [16]. To ensure saved poses and trajectories are available throughout the ROS system we used a ROS implementation of the MongoDB server.

As mentioned before, three sub-tasks are presented in this section. Each of them displays the usefulness of proposed approaches.

The first sub-task consists of moving the robot to a so called *initial position* in joint space. Moving to this pose is needed when the other robot is executing a task in a shared workspace. To gain an appropriate pose for each robot's *initial position*, the robot is moved kinesthetically. In order to guide the robot to a desired joint space configuration, joystick control would not be suitable.

When the robot is at an appropriate configuration, the button interface is used to store the joint configuration in the database. When the need for an *initial position* arises in the assembly process, the joint configuration is read from the database and *joint space motion with trapezoidal speed profile* is generated (see Section 3.1). Video stills of the robot executing this kind of movement can be seen in Figure 3.

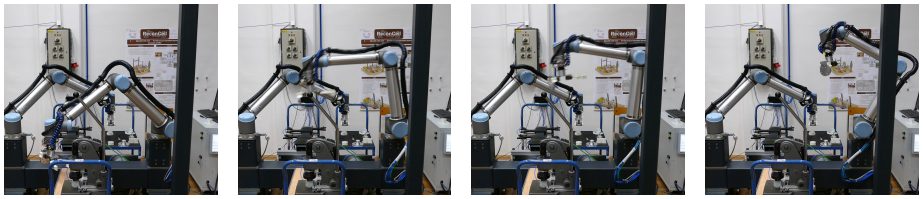

Fig. 3: Sub-task of moving the robot to an *initial position*. While kinesthetic guidance was used to record an joint pose, joint space motion with trapezoidal speed profile was executed to move the robot to the stored pose during assembly.

The second sub-task consist of a precise placement of an object. The robot must place a metal bushing on a metal shaft, which is already in the press (see Fig. 4). As this sub-task includes a constraint setting, contact with the environment, and precise positioning, kinesthetic guidance would not be suitable. Joystick control, presented in Section 2, was used to guide the robot to an appropriate position, where the bushing fit on the base. This demonstrated Cartesian position, stored in the database, was used to define needed relative points. *Cartesian space straight line & SLERP trajectories following the minimum jerk velocity profile* approach was used to generate point to point movements. As it produces a straight line in Cartesian space, appropriate robot trajectories were generated to perform the needed task. Figure 4 shows video stills of the robot successfully executing this sub-task.

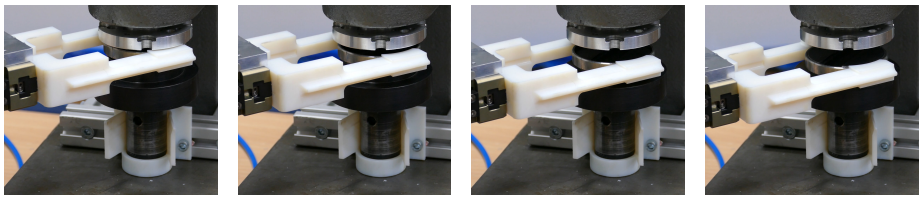

Fig. 4: Sub-task of positioning a bushing on a shaft. Due to a constraint setting, contact with the environment, and precise positioning, joystick control was needed for demonstrating appropriate robot position. Cartesian space straight line & SLERP trajectories following the minimum jerk velocity profile were used to successfully move the robot to the demonstrated point.

The third sub-task consists of activating the press, which is usually done by a human moving a lever. To automate this sub-task, we used one of the robot arms to move the lever. For this purpose, a human demonstrator moved the robot using kinesthetic guidance. With no expert knowledge of the robot and its surroundings, the human was able to move the robot in such a way to activate the press. As the whole trajectory is important, *Dynamic Movement Primitives* were used to encode it (see Section 3.2). Again, the button interface was used to mark the start and end of the desired trajectory, and thus ease the process. The trajectory was then executed during the assembly to effectively use the press. This can be seen on video stills in Figure 5.

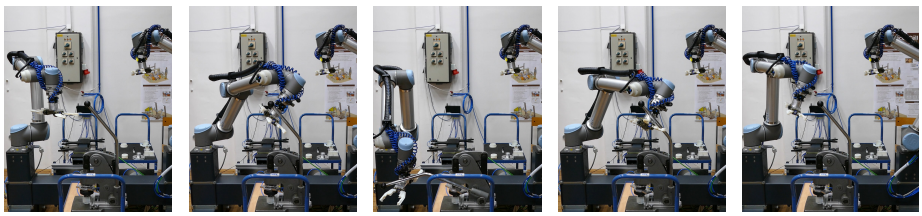

Fig. 5: Sub-task of activating a, usually human operated, press. To successfully move the lever, kinesthetic guidance was used to demonstrate the appropriate robot movement. The joint space trajectory was encoded with DMPs. The figure show the execution of the task by the robot.

All these three example sub-tasks represent different aspect of the chosen industrial assembly task. Combinations of proposed approaches were used throughout the whole industrial use-case implementation. They enabled us to successfully program the cell from (non-expert) human demonstration and to execute the presented assembly task.

5 Conclusion

This paper presents various approaches for knowledge acquisition through human demonstration for the purpose of executing robot assembly tasks. While kinesthetic guidance is used to demonstrate desired robot poses or whole trajectories, a joystick was integrated for the purpose of precise movements when demonstrating robot poses. While point to point movements were generated from demonstrated poses, free form trajectories were encoded as DMPs. Two approaches for point to point movements were presented: joint space motion with trapezoidal speed profile, and linear Cartesian space motion with minimum jerk velocity profile. Evaluation was presented in a form of a real industrial use case study. It showed that appropriately integrated and executed human demonstration can be used to program a new industrial assembly task with little expertise needed. Evaluation showed that different sub sets of the task can require different approaches of human demonstration and motion execution.

Acknowledgment

This work has received funding from the EU's Horizon 2020 IA ReconCell (GA no. 680431) and from GOSTOP programme C3330-16-529000 co-financed by Slovenia and EU under ERDF.

References

1. C. Breazeal and B. Scassellati, "Robots that imitate humans," *Trends Cogn. Sci.*, vol. 6, no. 11, pp. 481–487, 2002.
2. R. Dillmann, "Teaching and learning of robot tasks via observation of human performance," *Robot. Auton. Syst.*, vol. 47, no. 2, pp. 109–116, 2004.
3. A. Billard, S. Calinon, R. Dillmann, and S. Schaal, "Robot programming by demonstration," in *Handbook of Robotics* (B. Siciliano and O. Khatib, eds.), pp. 1371–1394, Secaucus, NJ, USA: Springer, 2008.
4. A. Ude, C. G. Atkeson, and M. Riley, "Programming full-body movements for humanoid robots by observation," *Robot. Auton. Syst.*, vol. 47, no. 2, pp. 93–108, 2004.
5. N. S. Pollard, J. K. Hodgins, M. J. Riley, and C. G. Atkeson, "Adapting human motion for the control of a humanoid robot," in *Proc. IEEE Int. Conf. Robotics and Automation (ICRA)*, (Washington, DC, USA), pp. 1390–1397, 2002.
6. J. Shotton, T. Sharp, A. Kipman, A. Fitzgibbon, M. Finocchio, A. Blake, M. Cook, and R. Moore, "Real-time human pose recognition in parts from single depth images," *Commun. ACM*, vol. 56, no. 1, pp. 116–124, 2013.
7. T. B. Moeslund, A. Hilton, and V. Krüger, "A survey of advances in vision-based human motion capture and analysis," *Comput. Vis. Image Und.*, vol. 104, no. 2, pp. 90–126, 2006.
8. M. Hersch, F. Guenter, S. Calinon, and A. Billard, "Dynamical system modulation for robot learning via kinesthetic demonstrations," *IEEE Trans. Robot.*, vol. 24, no. 6, pp. 1463–1467, 2008.
9. M. Deniša and A. Ude, "Synthesis of new dynamic movement primitives through search in a hierarchical database of example movements," *International Journal of Advanced Robotic Systems*, vol. 12, no. 10, p. 137, 2015.
10. A. Ude, B. Nemec, T. Petrič, and J. Morimoto, "Orientation in cartesian space dynamic movement primitives," in *IEEE International Conference on Robotics and Automation (ICRA)*, (Hong Kong), pp. 2997–3004, 2014.
11. S. Schaal, P. Mohajerian, and A. Ijspeert, "Dynamics systems vs. optimal control–a unifying view," *Prog. Brain Res.*, vol. 165, pp. 425–445, 2007.
12. A. J. Ijspeert, J. Nakanishi, H. Hoffmann, P. Pastor, and S. Schaal, "Dynamical movement primitives: learning attractor models for motor behaviors," *Neural Comput.*, vol. 25, no. 2, pp. 328–373, 2013.
13. T. Gašpar, B. Ridge, R. Bevec, M. Bem, I. Kovač, A. Ude, and Z. Gosar, "Rapid hardware and software reconfiguration in a robotic workcell," in *18th International Conference on Advanced Robotics (ICAR)*, pp. 229–236, 2017.
14. "Simulink Real-Time - Simulink - MATLAB & Simulink." https://www.mathworks.com/products/simulink-real-time.html. Accessed: 2017-03-16.
15. M. Quigley, K. Conley, B. Gerkey, J. Faust, T. Foote, J. Leibs, R. Wheeler, and A. Y. Ng, "Ros: an open-source robot operating system," in *ICRA Workshop on Open Source Software*, (Kobe, Japan), 2009.
16. "MongoDB." https://www.mongodb.com/. Accessed: 2019-02-4.