Proc. 2015 IEEE Int. Conf. on Robotics and Biomimetics, pp. 1195-1201, Zhuhai, China, Dec. 2015

A Programming by Demonstration System for Human-Robot Collaborative Assembly Tasks

Takuma Hamabe, Hiraki Goto, and Jun Miura
Department of Computer Science and Engineering
Toyohashi University of Technology

Abstract— **Programming by demonstration (PbD) has been one of the promising approaches to robot programming, where a robot learns how to operate by observing human demonstrations. While most of existing PbD research dealt with single-agent tasks, this paper deals with collaborative tasks, in which a robot and a human collaborate to assemble a structure by coordinating their operations. The system has three parts: demonstration observation parts, task modeling parts, and task execution parts. A scene recognition system is developed for reliable action recognition in human demonstrations. The developed PbD system has successfully been applied to a collaborative assembly tasks with more than twenty assembly steps.**

Index Terms— **Programming by demonstration, collaborative assembly, humanoid.**

I. Introduction

Service robots are expected to support people in various scenes of their everyday life in factories, in offices, and at home. People often collaborate to achieve tasks such as jointly carrying a heavy item and constructing a large structure together. Such a collaboration capability is one of the necessary functions of future robots.

One of the keys to a smooth collaboration between a human and a robot is a common understanding of the task to perform. For a robot to collaborate effectively, such an understanding needs to be represented explicitly as a *task model*. Kimura et al. [1] used a rule-based representation in which preconditions, operations, and expected results of assembly steps are described. Referring to the visual recognition result of the current state, an appropriate rule is selected for the robot's assistive action generation. Lens et al. [2] used a similar rule-based representation. Hanai et al. [3] used a state transition-based task model for predicting human actions. The robot can generate an efficient and collision-free hand motion based on the prediction.

We have previously developed a robot system that can assist a human to do an assembly task [4]. We proposed to use an FSM (finite state machine)-based task model, which is suitable for representing tasks with multiple feasible action sequences. Referring to the task model, our robot was able to perform assistive actions in a timely fashion. In this work, however, the task model was constructed manually and given in advance to the robot.

One promising way of teaching knowledge to robots is *programming by demonstration* (PbD) or *teaching by showing*, in which a human teacher just demonstrates a task and a robot observes it to make a task model [5]. Previous works can be divided into task-based [6], [7], [8] and motion-based [9], [10], [11]. The former assumes an observation system for extracting symbolic descriptions of the scene and represents a task model as a sequence of symbolic primitives, while the latter directly records motion trajectories of human or robot and/or generates models for robot control. Ogawara et al. [12] took a hybrid approach that both symbolic and trajectory-level representation are combined. These works deal with PbD for single-agent tasks.

Dominey et al. [13] developed a framework of interactive task model construction and usage for anticipating human actions in a collaborative assembly task. A speech-controlled humanoid incrementally learns the sequences of verbal orders from the human and his actions. The learned task is represented as a simple list of actions. In [14], they extended the approach to task learning not by explicit verbal orders but by observing a collaborative task execution by humans. Rozo et al. [15] developed a PbD system for impedance model acquisition in collaborative object handling. These works deal with a simple assembly task or motion-level learning.

The goal of this research is to develop a system which learns task knowledge from human demonstrations of collaborative assembly and reproduces the same task in a human-robot collaboration scenario. The task we deal with is an assembly of a small chair model; this task includes more than twenty assembly steps. Our robot assists a human when he executes steps which are not easy to do only by himself. The task knowledge is, therefore, for the robot to recognize the current situation and to judge if an assistive action is necessary. Visual scene recognition routines are developed, which are used both for human demonstration recognition and for scene recognition by the robot in a human-robot collaborative assembly.

The rest of the paper is organized as follows. Sec. II describes an overview of the system. Sec. III explains an FSM-based representation of collaborative tasks. Sec. IV describes a setup and functions of a vision-based human demonstration observation system. Sec. V explains the process of generating a task models from a set of observed data of human demonstrations. Sec. VI shows experimental results of human-robot collaborative assembly based on acquired task models. Sec. VII discusses remaining issues and future work. Sec. VIII summarizes the paper.

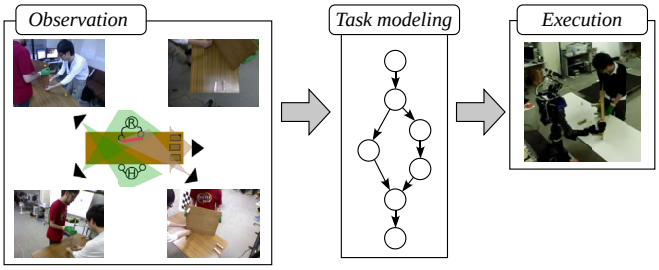


Fig. 1. Configuration of a programming-by-demonstration (PbD) system for collaborative assembly.

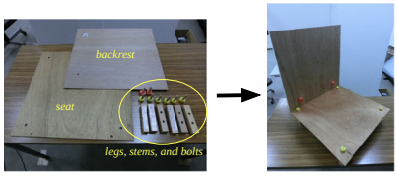


Fig. 2. The chair assembly task.

II. System Overview

Most programming by demonstration (PbD) systems have three main parts (see Fig. 1). The *observation* part (left) observes human demonstrations and recognizes what actions are performed in what order. The *task modeling* part (center) examines the recognition results and organizes them into *task models*. The *execution* part (right) executes the same task using the acquired task model. A task model is usually an abstracted description of the task, which can cope with the difference of the scene between the demonstration and the execution phase (e.g., different locations of objects) as well as the difference in configuration of a human and a robot.

The human demonstration observation system is equipped with four RGB-D cameras (MS Kinect), two of which observe the objects in the scene while the others mainly observe human action. This system generates a sequence of states and actions in the demonstration, which is then fed to the task modeling part.

The task modeling part gets a set of demonstration sequences and merges them into an FSM (finite state machine)-based task model, which allows multiple possible assembly sequences for a task. The task model is for the robot, not for the human; the task model gives the robot how the assembly steps proceed and how the human does actions in these step, thereby enabling the robot to know when and what assistive actions it should do.

The task execution system realizes collaborative assembly tasks by a human and a humanoid robot. The robot observes human actions and automatically determines when and what to do, with referring to the generated task model.

Fig. 2 shows the assembly task treated in this paper. The task includes steps for fastening a bolts with a leg or a stem. We usually use both hands for such a step and another hand is needed for holding the part to which the leg or the stem is attached. Since our robot currently has a limitation on executable operations in the assembly tasks, that is, it can perform only pick-and-place and holding operations, the roles of two persons in demonstration are fixed in advance; *PersonR* for the robot role does only holding actions to be executed by the robot in the execution phase and *PersonH* for the human role does the rest.

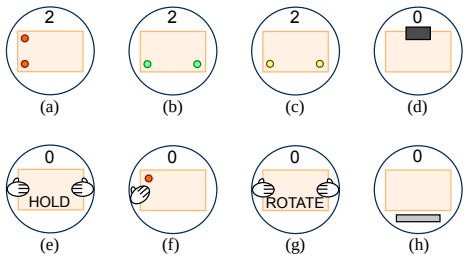


Fig. 3. Symbolic explanation of state descriptors. (a) The number of attached legs and positions. (b) The number of attached stems and positions. (c) The number of attached bolts for fixing the backrest. The numbers on the top indicates the number of attached corresponding parts. (d) Whether the robot holds the seat/backrest. (e) Whether the human holds the seat. (f) Whether the human is attaching a leg. (g) Whether the human is manipulating the seat/backrest. (h) Whether the robot is estimating the poses of the legs on the work table.

III. Task Representation

Finite state machine (FSM) is defined by a set of states and state transitions and suitable for representing flow with multiple paths. We, as a representation of our task model, use a type of FSM, whose output values are determined both by its current state and the detected event. We adopted this type of FSM in our previous work on realizing a human-robot collaborative assembly task [4]. This section briefly explains the representation. In this work, we added a few descriptors to the previous ones to cover a larger variety of assembled parts.

A task execution proceeds with state changes caused by human and/or robot actions. Fig. 3 shows the state descriptors used in this paper. There are five descriptors for *static* states, that is, the states those which are related to parts status: (a)(b)(c) how many legs/stems/bolts are attached and where, (d) whether the robot holds the seat/backrest, and (e) whether the human holds the seat/backrest.

There are also three descriptors for describing *dynamic* states, that is, the states those which are for representing ongoing human actions: (f) whether the human is attaching a leg/stem/bolt, (g) whether the human is rotating the seat, and (h) whether the robot is estimating the pose of legs/stems on the work table. Explicitly representing such dynamic states is necessary because the states of some assembled parts are sometimes hard to observe due to occlusion by hands, and detecting the end of an action may provide a good timing for calling the routine for recognizing the result of the action (i.e., a change of a static state).

Fig. 4 shows a part of the obtained task model for chair assembly, illustrating a sequence of state transitions shown by the images on the left. A state transition is specified by the detection of an event with the corresponding hand and recognition action.

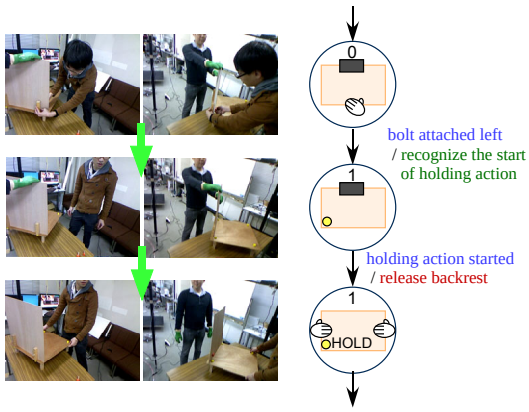


Fig. 4. A part of the chair assembly task model. Inputs (blue) to the FSM are detected static or dynamic states. Outputs are the start of hand action (red) or recognition action (green).

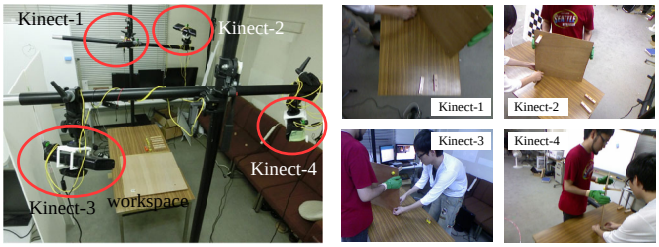


Fig. 5. Observation system with four Kinects.

IV. Observing Human Demonstrations

Observing human demonstrations is composed of (1) recognition of object and human actions and (2) generation of state transition descriptions. This section describes our observation system and these recognition functions.

A. Observation system

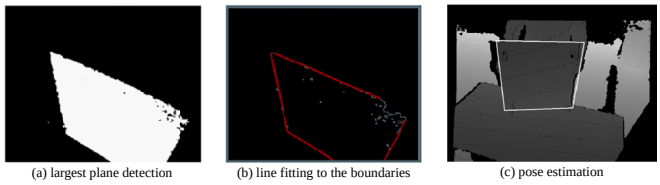
Fig. 5 illustrates the observation system, which is composed of four RGB-D cameras (MS Kinect). Two of the cameras are for observing human actions while the others for detecting objects and estimating their poses. The figure also shows images of a collaborative assembly captured by the four cameras. We use the Shake'n Sense technology [16] to eliminate interference between cameras.

We use two PCs, each of which handles two cameras. Time stamp information is added to data from each camera, which comes from the connected PC. To synchronize the two PCs, we manually detect the images capturing an event (e.g., turning on an LED light) and examine the difference between time stamps of respective PCs. Geometric calibration is also done using a standard OpenCV calibration routine using a checker board.

We have designed image processing routines for recognizing *states*, which are described by object states and hand states. The following subsections explain these routines.

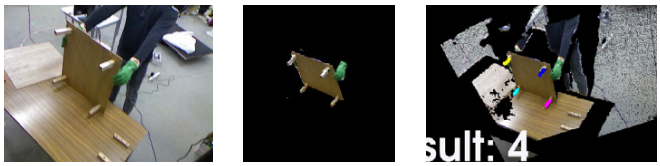
B. Object detection and pose estimation

The objects treated in this paper are large flat boards (seats and backrests), small long blocks (legs and stems), and bolts



(a) largest plane detection (b) line fitting to the boundaries (c) pose estimation

Fig. 6. Pose estimation of a seat.



(a) input image. (b) extraction of point cloud near the tabletop. (c) four detected legs (colored point clouds).

Fig. 7. Detection of legs attached to the seat.

(see Fig. 2). We prepare routines for detecting respective objects. The detection is based on a plane extraction and a size filtering. We also prepare a routine for estimating pose of a seat or a backrest because their poses are useful for setting a region of interest (ROI) for detecting attached legs, stems, and bolts. The models of all objects are given in advance.

Fig. 6 shows the process of detection and pose estimation of a seat: detection of the largest plane, fitting of boundary lines, and the estimation of the seat. Fig. 7 shows the process of detection of four legs attached to the seat: detection of the seat, extracting point clouds nearby, and plane detection and matching with the leg model.

C. Human motion detection

Human motion is detected using the skeleton tracking function of the Kinect Windows SDK. From the skeleton data, hand position is extracted and used for interpreting human actions, as explained below. The position data is converted to the world coordinate system on the workspace using the calibration data.

D. Generating state transition descriptions

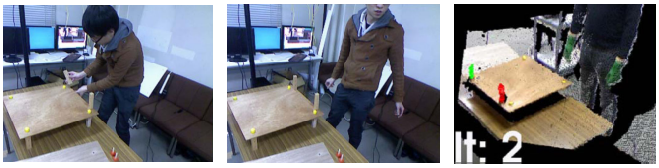
1) Event sequence-based action recognition: The task of observation system is to generate state transition descriptions from RGB-D and skeleton data. Dynamic states and state transitions are associated with the corresponding human actions. Since it is sometimes difficult to continuously track hands and objects and/or recognize actions due to frequent occlusions, we develop a specialized routine for each dynamic state or each state transition; each routine verifies a predefined sequence of human action events and object detection events. We here present two examples of such routines. Routines for other actions are prepared in a similar way.

Action "*PersonH picks up a leg*" is detected by verifying events in the following three steps (see Fig. 8):

1) The hand of PersonH approaches a leg position on the work table and then moves away from it.

(a) Two legs before action. (b) Hand approaches and moves away from legs. (c) One leg after action.

Fig. 8. Pick up a leg on the work table.



(a) Hand with a leg approaches the tabletop. (b) Hand moves away from the tabletop. (c) The number of attached legs increases.

Fig. 9. Detection of legs attached to the seat.

2) Detect the number and the pose of legs on the work table.
3) If the number of legs is smaller than the one before the hand movement, PersonH is considered to have picked up a leg. Otherwise, no action was performed.

Action "*PersonH attaches a leg to the seat*" is detected by verifying events in the following three steps under the condition that PersonH picks up a leg (see Fig. 9):

1) The hand of PersonH (with a leg) approaches the seat and then moves away from it.
2) Detect the number and the pose of legs attached to the seat.
3) If the number of legs is larger than the one before the hand movement, PersonH is considered to have attached a leg to the seat. Otherwise, no action was performed.

2) State transition descriptions: Event detection is performed every five frames. Based on a detected event, the observation system chooses appropriate detection routines and watches possible subsequent events. Once an action is detected, it is recorded on the list, and the final complete list is output for further processing.

V. Task Model Generation from Observed Data

A. Merging multiple sequences into one task model

Observed action sequences may be different from each other in the order of executed actions. When attaching two stems to the seat at different positions, for example, the order of attaching the two legs is not important; either leg can be attached first. Since we do not tell the demonstrators an exact sequence of assembly steps, different sequences may be observed in multiple demonstrations. It is therefore necessary to merge all observed sequences into one task model. Dufay and Latombe [17] deals with such a merging problem by an inductive learning approach. They generated a graph-based task model which uses sensor-level decisions. We here deal with a merging of symbolic level

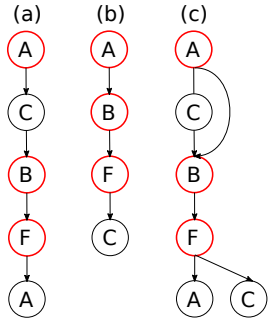


Fig. 10. Merging sequences using LCS. Red nodes are the ones included in the extracted LCS.

action sequences. Nicolescu and Matarić [18] proposed a general framework of combining several natural methods in task learning: demonstration, generation, feedback from teacher, and practice. In the generalization step, they propose to use *longest common subsequence* (LCS) [19]. An LCS is the longest sequence of symbols which appear in the same order in multiple input sequences. This is useful for extracting common symbols which might be merged for a compact representation. Abbas and MacDonald [20] show an algorithm to generate a consistent task graph by combining all demonstrated sequences at once.

We also use LCS for merging observed demonstrations. We, however, additionally examine the generated graph and exclude sequences which have not been observed, since such a sequence might not be feasible.

B. LCS-based merging and modification

Fig. 10 shows an example of sequence merging using LCS. From two input sequences (a) and (b), the LCS: $A \rightarrow B \rightarrow F$ is extracted. The symbols in the LCS in both sequences are merged and the final graph (c) is generated, which includes the two input sequences.

A merged task model often covers sequences which have not been actually observed. Such a generalization might be useful; the model can be applied to unknown, new cases. At the same time, however, it might include an unfeasible sequence. Let us consider the example shown in Fig. 11. From two input sequences (a), a merged model is generated (b). Although both input sequences include actions C and E only once, the merged model allows executing either action twice. If both C and E are necessary for this assembly (e.g., attaching legs to two different positions), the merged model includes feasible action sequences.

To solve this problem, we take an approach that only observed sequences are included in a merged model. Unobserved sequences are excluded as follows:

1) Find a consecutive action subsequence, to which and from which multiple paths exist (e.g., action D in Fig. 11(c)).
2) Collect actions connected to the subsequence (e.g., actions C and E before and after action D in Fig. 11(c)) and check if all possible paths have been observed.
3) If yes, do nothing for this subsequence. If no, make the same subsequence so that the number of the

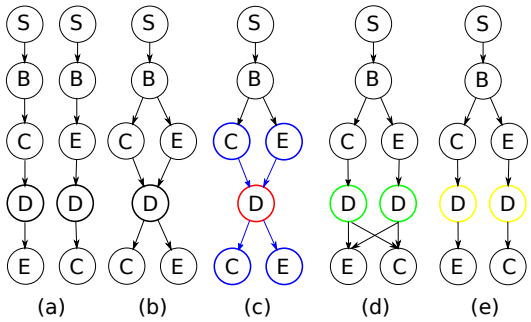


Fig. 11. Excluding unobserved sequences from the merged model.

subsequences is equal to that of incoming paths (e.g., green-colored action D in Fig. 11(d)).

4) Examine the output side of the subsequence and exclude unobserved paths (Fig. 11(e)).

C. Adding a loop for coping with recognition/manipulation failures in execution phase

Task models are generated from human demonstrations. Since humans usually do not fail in scene recognition and object manipulation, the generated models initially do not include so-called error recovery routines. The robot is, however, not as versatile as humans at this moment and some recovery routines are necessary for reliable robot execution. We therefore add feedback loops to retry recognition and motion planning operations which are known, from our experiences, to sometimes fail; we add retry loops at part pose recognition and grasp point calculation.

VI. Human-Robot Collaborative Assembly using Acquired Task Models

We use a HIRO humanoid robot by Kawada Co. for human-robot collaborative assembly. The robot uses one Kinect for recognizing objects and human hand actions. The visual recognition routines are similar to those used in observing human demonstration but some modifications were made considering a different configuration of the robot's vision system (i.e., one Kinect at the head position).

We conducted PbD experiments for the chair task. For each task, two persons demonstrated collaborative assembly twice, and the system learned a merged task model. The obtained task models are then tested using the human-robot collaborative assembly system. We here show only the results for the chair task due to a space limitation.

Fig. 12 shows the merged task model learned from the observation of two demonstrations. Nodes indicate object state and hand states, while edges indicate transitions triggered by detected events and associated hand or recognition actions.

Using the task model, we conducted the human-robot collaborative assembly of chair twice. Table I summarizes the two experiments. The assembly paths for these experiments are different at many places in the task model with multiple options exist. The robot with the acquired task model can cope with such different human choices thanks to the FSM-based task representation. The times for completing the task are different for the two experiments. This is due to a larger number of trials in scene recognition in the first experiment; the robot tries to recognize the scene several times until one of the expected situations is detected, especially in the loops added to cope with recognition failure (see Sec. V-C).

VII. Discussion

The current system is a first step towards more versatile PbD system for collaborative assembly. Many research challenges remain to be tackled.

In the current setting, the roles of robot and human in collaborative assembly are fixed beforehand and, therefore, the human demonstrations are performed considering those roles. This is due to the limitation of robot's manipulation capability. The acquired task model itself can, however, basically be applied to the situation when the current human and the robot roles are exchanged. Moreover, more flexible breakdown of actions would be possible. Implementing various skills on the robot to cope with such flexible collaboration is one of future work. Application to robot-robot collaboration would also be interesting.

Our current system and most of existing ones take a so-called *batch* approach. That is, the demonstration/observation phase, the task modeling phase, and the execution phase are sequential, as shown in Fig. 1. Usual human-to-human teaching scenarios are, however, more interactive. During the demonstration of a teacher, a learner may have a question on some demonstration steps or ask the teacher to show some steps more clearly. Even in the execution phase, the learner may ask questions to make the model be more precise. Such an interaction will enhance the PbD approach more effective.

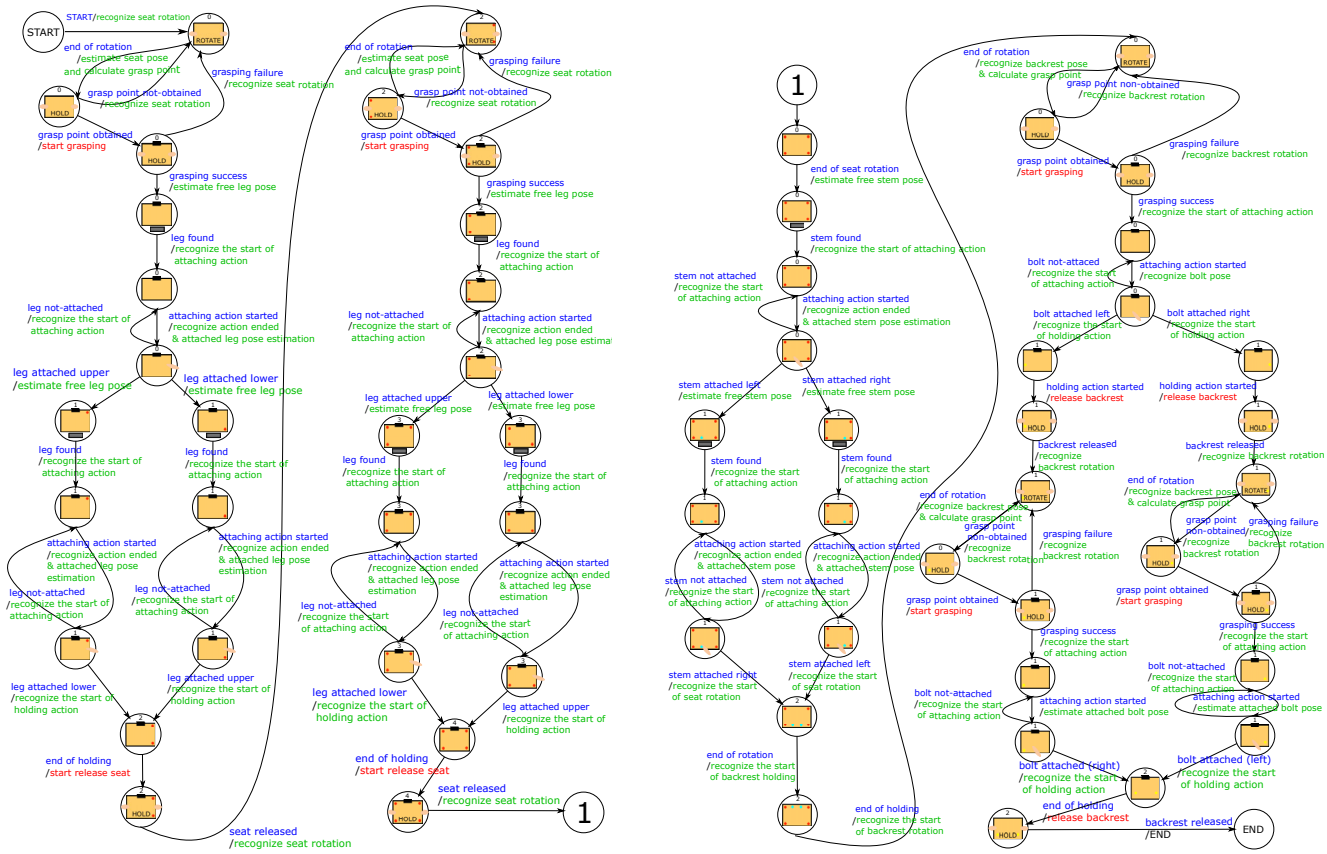
Currently we assume a complete set of recognition routines for generating a symbolic description of the scene and a set of state descriptors are given in advance. This is actually a strong limitation when extending the system to include a more variety of tasks. While improving the recognition ability is certainly necessary, other approaches would also be interesting such as unsupervised learning for action labeling [21].

VIII. Summary

This paper has described a Programming by Demonstration (PbD) system for collaborative assembly. We developed a PbD system which observes human-human collaborative assembly demonstrations, generates task models using an FSM-based representation, and controls the robot to support human-robot collaborative assembly. The system has been successfully applied to assembly tasks with more than twenty action steps. Current limitations and future work are also discussed.

Acknowledgment

This work is supported in part by KAKENHI 15H01616 from JSPS.



(a) Steps for assembling the seat part.

(b) Steps for assembling the backrest part.

Fig. 12. The chair task model learned from two demonstrations.

TABLE I

SUMMARY OF HUMAN-ROBOT COLLABORATIVE ASSEMBLY EXPERIMENTS.

Exp. id	Order of attaching legs	Order of attaching stems	Order of attaching bolts	execution time (sec.)
1	lower right→upper right→upper left→lower left	right→left	right→left	820
2	upper right→lower right→lower left→upper left	left→right	left→right	642

REFERENCES

[1] H. Kimura, T. Horiuchi, and K. Ikeuchi. Task-Model Based Human Robot Cooperation Using Vision. In *Proceedings 1999 IEEE/RSJ Int. Conf. on Intelligent Robots and Systems*, pp. 701–706, 1999.

[2] C. Lenz, S. Nair, M. Rickert, A. Knoll, W. Rösel, J. Gast, A. Bannat, and F. Wallhoff. Joint-action for humans and industrial robots for assembly tasks. In *Proceedings of 17th IEEE Int. Symp. on Robot and Human Interactive Communication*, pp. 130–135, 2008.

[3] R. Hanai, R. Oya, T. Izawa, and M. Inaba. Motion Generation for Human-Robot Collaborative Pick and Place Based on Non-Obstruction Strategy. In *Proceedings of 2011 IEEE Int. Conf. on Robotics and Biomimetics*, pp. 20–25, 2011.

[4] H. Goto, J. Miura, and J. Sugiyama. Human-Robot Collaborative Assembly by On-line Human Action Recognition Based on an FSM Task Model. In *Proceedings of HRI2013 Workshop on Collaborative Manipulation: New Challenges for Robotics and HRI*, 2013.

[5] A. Billard, S. Calinon, R. Dillmann, and S. Schaal. Robot Programming by Demonstration. In B. Siciliano and O. Khatib, editors, *Springer Handbook of Robotics*, chapter 59, pp. 1371–1394. Springer, 2008.

[6] Y. Kuniyoshi, M. Inaba, and H. Inoue. Learning by Watching: Extracting Resuable Task Knowledge from Visual Observation of Human Performance. *IEEE Trans. on Robotics and Automation*, Vol. 10, No. 6, pp. 799–822, 1994.

[7] K. Ikeuchi and T. Suehiro. Toward an Assembly Plan from Observation Part I: Task Recognition with Polyhedral Objects. *IEEE Trans. on Robotics and Automation*, Vol. 10, No. 3, pp. 368–385, 1994.

[8] R. Dillmann. Teaching and Learning of Robot Tasks via Observation of Human Performance. *Robotics and Autonomous Systems*, Vol. 47, pp. 109–116, 2004.

[9] S. Nakaoka, A. Nakazawa, F. Kanehiro, K. Kaneko, M. Morisawa, H. Hirukawa, and K. Ikeuchi. Learning from Observation Paradigm: Leg Task Models for Enabling a Biped Humanoid Robot to Imitate Human Dances. *Int. J. of Robotics Research*, Vol. 26, No. 8, pp. 829–844, 2007.

[10] S. Calinon and A.G. Billard. What is the Teacher's Role in Robot Programming by Demonstration? Toward Benchmarks for Improved Learning. *J. of Interaction Studies*, Vol. 8, No. 3, 2007.

[11] S. Calinon, P. Evrard, E. Gribovskaya, A. Billard, and A. Kheddar. Learning collaborative manipulation tasks by demonstration using a haptic interface. In *Proceedings of the 14th Int. Conf. on Advanced Robotics*, pp. 1–6, 2009.

[12] K. Ogawara, J. Takamatsu, H. Kimura, and K. Ikeuchi. Extraction of Essential Interactions Through Multiple Observations of Human Demonstrations. *IEEE Trans. on Industrial Electronics*, Vol. 50, No. 4, pp. 667–675, 2003.

[13] P.F. Dominey, G. Metta, F. Nori, and L. Natale. Anticipation and Initiative in Human-Humanoid Collaboration. In *Proceedings of the 8th IEEE-RAS Int. Conf. on Humanoid Robots*, 2008.

[14] S. Lallee, F. Warneken, and P.F. Dominey. Learning to Collaborate by Observation. In *Proceedings of Humanoids 2009 Workshop*

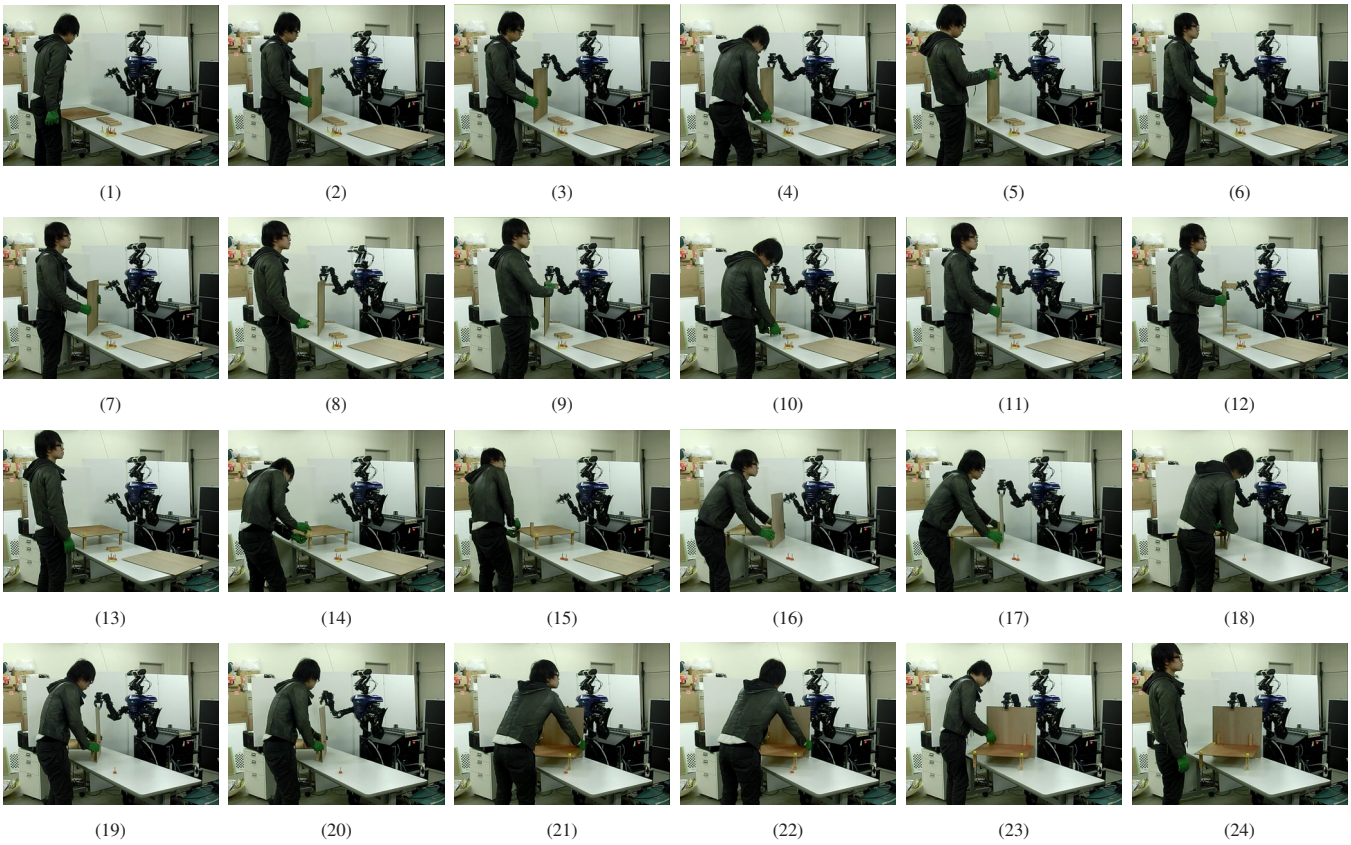


Fig. 13. A collaborative chair assembly sequence. (1) Start of the assembly. (2) Human holds the seat. (3) Robot also holds it. (4) Human release the seat and then attaches a leg. (5) He is attaching another leg. (6) Robot sees Human holds the seat, and (7) releases it so that Human can rotates it. (8) Robot holds the seat again. (9) Human is attaching the third leg and (10) the fourth leg. (11) Human holds the seat with the legs, (12) Robot release it and (13) Human puts it on the work table. (14)(15) Human attaches two stems. (16) Human puts the backrest at a right position and (17) Robot holds it. (18) Human is fixing the backrest with a bolt. (19) Human holds the whole structure, (20) Robot releases it, and (21) Human rotate it. (22) Robot holds it again, and (23) Human put the other bolt for fixing. (24) The chair assembly is completed.

on Developmental Psychology Contributions to Cooperative Human Robot Interaction, 2009.

[15] L.D. Rozo, S. Calinon, D.G. Caldwell, P. Jiménez, and C. Torra. Learning Collaborative Impedence-based Robot Behaviors. In *Proceedings of 2013 AAAI Conf. on Artificial Intelligence*, pp. 1422–1428, 2013.

[16] D.A. Butler, S. Izadi, O. Hilliges, D. Molyneaux, S. Hodges, and D. Kim. Shake'n'sense: reducing interference for overlapping structured light depth cameras. In *Proceedings of the SIGCHI Conf. on Human Factors in Computing Systems (CHI '12)*, pp. 1933–1936, 2012.

[17] B. Dufay and J.C. Latombe. An Approach to Automatic Robot Programming Based on Inductive Learning. *Int. J. of Robotics Research*, Vol. 3, No. 4, pp. 3–20, 1984.

[18] M.N. Nicolescu and M.J. Matarić. Natural Methods for Robot Task Learning: Instructive Demonstrations, Generalization, and Practice. In *Proceedings of the 2nd Int. Joint Conf. on Autonomous Agents and Multi-Agent Systems*, pp. 241–248, 2003.

[19] T.H. Cormen, C.E. Leiserson, R.L. Rivest, and C. Stein. *Introduction to Algorithms, 2nd Edition*. MIT Press, 2001.

[20] T. Abbas and B.A. MacDonald. Generalizing Topological Task Graphs from Multiple Symbolic Demonstrations in Programming by Demonstration (PbD) Processes. In *Proceedings of 2011 IEEE Int. Conf. on Robotics and Automation*, pp. 3816–3821, 2011.

[21] C. Wu, J. Zhang, S. Savarese, and A. Saxena. Watch-n-Patch: Unsupervised Understanding of Actions and Relations. In *Proceedings of 2015 IEEE Conf. on Computer Vision and Pattern Recognition*, pp. 4362–4370, 2015.