2022 IEEE 18th International Conference on Automation Science and Engineering (CASE)
August 20-24, 2022. Mexico City, Mexico

2022 IEEE 18th International Conference on Automation Science and Engineering (CASE) | 978-1-6654-9042-9/22/$31.00 ©2022 IEEE | DOI: 10.1109/CASE49997.2022.9926527

Robot Learning of Assembly Tasks from Non-expert Demonstrations using Functional Object-Oriented Network

Yi Chen, David Paulius, Yu Sun, and Yunyi Jia*

Abstract — **Robot Learning from Demonstration (RLfD) is a research field that focuses on how robots can learn new tasks by observing human performances. Existing RLfD approaches mainly enable robots to repeat the demonstrated tasks by mimicking human activities, which usually requires efficient demonstrations for human experts. This paper proposes a new Function Object-Oriented Network (FOON) based approach to make robots learn and optimize assembly tasks from non-expert demonstrations. It first proposes an assembly FOON construction approach with automatic subgraph creation and merging algorithms to extract information from multiple non-expert demonstrations. It then proposes an assembly task tree retrieving approach with a robot execution optimization process to make the robot learn and generate the best possible task execution plan from the constructed FOON. The proposed approaches are validated through experiments with a dual-arm YuMi robot and the experimental results illustrate the effectiveness and advantages of the proposed approach.**

I. Introduction

Industrial robots have been widely utilized for several decades for different applications in industrial manufacture. In such cases, robots are usually precisely pre-programmed and execute repetitive manipulations in well-constructed environments [1]. However, in many other cases, such as domestic service and collaborative assembly, robots must work on non-repetitively high-mix tasks in a dynamic environment [2]. It is impossible to pre-program all the potential situations and events in these applications, as they require robots that can solve problems on their own after being programmed with the necessary skills. The conventional machine-code level programming methods, which are usually time-consuming, lack flexibility and high expertise demanding, and are not well qualified in these cases. Robot Learning from Demonstration (RLfD) provides an easy and intuitive way to program robot behaviors, potentially reducing development time and costs tremendously [3].

In the RLfD perspective, conveying human demonstrations to robot knowledge is an important aspect of the problem. Many studies have been reported in this field, and the information is commonly transferred by kinematic teaching [4], teleoperation [5], or recording human motion [6] or tool motion directly [7]. Most of these approaches aim to enable robots to either repeat or imitate the demonstrated trajectories. Although these approaches significantly reduced the coding effort and setup time cost for new tasks, robots still lack the knowledge of the insights of their tasks and corresponding environments. The demonstrations for each task are independent of those of other tasks, and robots need to be reprogrammed by new demonstrations when there are tiny changes in some factors of the task. It makes these approaches not efficient enough for applications in assembly tasks. For example, different assembly tasks usually consist of different sequences of manipulations, but when we split those manipulation sequences, they usually share some common one-step manipulations on standard parts or tools, which means some sections of human demonstrations for one assembly task are potentially reused in other assembly tasks. To this end, studies on modeling task-level knowledge (e.g. features and states of objects, task constraints, etc.) and robot manipulations in the same framework become a new frontier in robotics.

Many approaches have been investigated for assembly task modeling based on RLfD, such as vision-based approaches [8], natural-language-based approaches [9], graph-based approaches [10], probabilistic-based approaches [11], machine-learning-based approaches [12], and multimodal-based approaches [13]. Recently Huang and Sun [14] have used Long Short Term Memory networks (LSTM) to learn how to handle the dynamics in pouring. These approaches enable robots to learn knowledge of demonstrated tasks and necessary primitive skills from human demonstrations. With the learned knowledge, robots can achieve a certain level of adaptation to new tasks or environments by taking advantage of the learned skills and a situational awareness ability. Additionally, some approaches are also developed to represents the hierarchical tasks and the corresponding primitive skills, which enable robots to learn hierarchical tasks from human demonstrations [15], [16]. Mohseni-Kabir et al. [17] proposed a framework that makes robots learn the task hierarchy and trajectory-level primitive skills simultaneously. However, most of the existing approaches usually assume that the demonstrations are performed by human experts who can conduct the task in an efficient way in order to achieve efficient robot executions through learning in RLfD, e.g., in assembly tasks, the demonstrations must be conducted by an expert worker in the assembly domain. To enable robots to learn from non-expert demonstration can reduce the time and the cost of human training in industrial

This work was supported by the National Science Foundation under grant IIS-1845779.

Yi Chen is currently with the ABB US Research Center in Raleigh, NC, and he previously performed this work with the Department of Automotive Engineering, Clemson University, Greenville, SC 29607, USA. Yunyi Jia is with the Department of Automotive Engineering, Clemson University, Greenville, SC 29607, USA. {yc4@clemson.edu, yunyij@clemson.edu}. David Paulius and Yu Sun are with the Department of Computer Science and Engineering at the University of South Florida, Tampa, FL 33620, USA.{ paulius.david@gmail.com, yusun@mail.usf.edu}

field and make robots more friendly to ordinary people, which is potentially expand the usage of robots in the daily life.

It is worth mentioning that there are several recent works [18]–[20] which use reinforcement learning to learn from imperfect human demonstrations. However, such works mainly aim to learn the lower-level control policies which are very different from higher-level assembly tasks. Also, these approaches usually require a significantly large amount of demonstrations and usually cannot always guarantee the best solution compared to model-based approaches because of their data-driven heuristic nature.

In this paper, we aim to advance the robot learning from demonstration by reducing the requirement of demonstrations with human experts. Our major motivation is to make robots learn just like humans who can usually learn tasks from others from different perspectives and then synthesize the best way to accomplish the task although the others may not always demonstrate the tasks in efficient ways. Therefore, we propose a new FOON-based approach to address robot learning from non-expert demonstrations in the robotic assembly contexts. We have previously introduced FOON as a graphical knowledge representation of human cooking tasks [21], [22]. In this paper, we extend FOON to learning assembly tasks from non-expert demonstrations. We reconstruct some features of Functional Object-Oriented Network (FOON) to make it suitable for assembly tasks and also develop automatic subgraph creation and merging algorithms for FOON construction from multiple non-expert assembly demonstrations. Furthermore, we also propose an assembly task tree retrieving algorithm with the robot execution optimization process to enable robots to learn and generate the best possible task execution based on the constructed FOON. Because our approach employs models, it requires just a few demonstrations, which is significantly fewer compared to existing data-driven approaches.

II. WEIGHTED FUNCTIONAL OBJECT-ORIENTED NETWORK FOR ASSEMBLY TASKS

A. Structure of FOON

The FOON proposed in our paper is a graphical task representation that includes robot motions, physical interactions between robots and objects in the workspace, and overall assembly task state descriptions. It provides a more intuitive way to represent, analyze, and visualize human demonstrations. More importantly, FOON decouples objects and motion from a holistic view of action. This decoupling allows FOON nodes to have a more granular representation and gives FOON more flexibility than traditional task-step representations. The flexibility created by the motion nodes and object nodes enables more integrated task-tree merging and can generate more optimal task trees.

The constructed FOON for assembly tasks is a bipartite network that contains motion nodes and object state nodes. To make it suitable for assembly task representations, as opposed to the original FOON for cooking tasks, a specific type of object node, the so-called assembly state node, is introduced into the FOON to keep track of the assembly states in human demonstrations. Mathematically, an assembly state node can be either an input node or an output node of a motion node, and each motion node can have at most one assembly state node in its input nodes and at most one assembly node in its output nodes. FOON would only allow the object state nodes and assembly state nodes to be connected to motion nodes, and the motion nodes to be connected to object state nodes and assembly state nodes, which form a bipartite network.

1) Nodes

The nodes in FOON for assembly tasks have three types: object state O, motion M, and assembly state A. An object state node N_O represents a state of an object, which includes the object's identifier, name, and attributes. The attributes of an object include but are not limited to, its position, mass, size, color, and so on. For assembly tasks, a single part in a specific state can be defined as an object state node. When the state of the part is changed, a new object state node is generated. When more than one part is assembled as a component, a new object state node is also generated for this component. An assembly node N_A has the same mathematical properties as an object node. It does not represent the state of a part or component, but rather it represents the state variations of the final assembly product. A motion N_M represents a manipulation related to specific objects. The information in a motion node includes, but is not limited to, the type of action, manipulated objects, start position, goal position, and so on. In a FOON for assembly, each object node and assembly node are unique in their name and attributes. Motion nodes with exactly the same attributes can appear at different locations in the FOON graph.

2) Edges

FOON for assembly is a directed graph. An edge can be drawn from an object state node or an assembly state node to a motion node, or vice-versa. In general, the object state nodes with edges directed to a motion node are regarded as task constraints of the manipulation in the motion node. The object state nodes that have edges that come from a motion node are regarded as the manipulation outcomes. These are analogous to input and output nodes coined in [21].

If there is an assembly node that has an edge directed to a motion node, this assembly node would indicate the state of the final assembly product before the manipulation corresponding to the motion node. If a motion node has an edge directed to an assembly node, this assembly node would indicate the state of the final assembly product after the manipulation corresponding to the motion node. Some motion

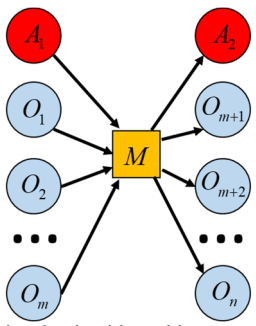

Fig. 1. A functional unit with m object state nodes and one assembly state node as input, (n-m) object state nodes and one assembly state node as output.

nodes may not lead to a change in the final assembly state so that they do not have an assembly node as input or output.

3) Functional Unit

A functional unit in a FOON for assembly tasks consists of a motion node and multiple object state nodes. Some functional units also contain one assembly state node as input and/or output. As shown in Fig. 1, the object state nodes and the assembly node with edges directed to the motion node are the input nodes of the functional unit, and the object state nodes and the assembly state node with edges pointing from the motion node are the output of the functional unit. In assembly tasks, the input assembly state node represents the state of the final assembly product before the manipulation of the motion node. Together, the object state nodes form the task constraints of the manipulation. A functional unit is defined as a minimum learning unit in a FOON for assembly.

B. Integrating Weights into a FOON

In this paper, the weights of FOON reflect both the success rate W_{SR} and average efficiency W_e of a given manipulation. The success rate is mainly related to the capabilities of robots, which corresponds to the accuracy required for executing manipulations. For example, robots have payload limitations for their arms and object size limitations for their grippers. It is nearly impossible for robots to handle manipulations beyond those limitations. Low-accuracy manipulations, such as handover, usually have a relatively higher success rate. On the contrary, the high-accuracy manipulations, such as placing a bolt into a hole, usually have a relatively lower success rate. In terms of average efficiency, it mainly relates to safety and the complexity of manipulations. For instance, robots can move at a higher speed when simply picking up a single part from stock and handing it over to humans. On the contrary, robots may have to run at a slower speed when moving a component, which contains some loose parts on it, from one location to another. Moreover, some manipulations, such as switching the location of two parts, may contain multiple motion steps and require two robot arms to cooperate with each other, which means relatively higher complexities and lower efficiency. The weight of manipulation for the corresponding functional unit can be computed by:

$$W_{FU} = W_{SR} \times W_e \quad (1)$$

where W_{SR} is the success rate of the manipulation and W_e is the weight of efficiency for the manipulation.

The representative success rates of manipulations can be determined empirically. The average efficiency of a specific type of manipulation can be identified from the average time cost of those in human demonstrations via hand tracking and object tracking with the optical tracking system. However, these are not trivial tasks to perform. To simplify this, we assign estimated weights based on our experiences and results of waypoint teaching and trajectory execution time. Manipulations that cannot be executed by a robot were assigned a success rate W_{SR} of 0.01, while other motions would be assigned higher values which varies between 0.8 to 0.95. In addition, single-arm-single-part manipulations were assigned a higher average efficiency weight which varies between 0.9 to 0.95, single-arm-multi-parts manipulations (moving a component) were assigned a medium average efficiency weight which varies between 0.75 to 0.85, and dual-arm-single-component manipulations, which require both robot arms to work on different parts of a single component at the same time, assign a lower average efficiency weight of 0.1.

Algorithm 1: Merging New Subgraph to Universal FOON

for all functional unit FU_i in new subgraph:
  for all existed functional unit FU_j in the universal FOON **do**
    if FU_i is equal to FU_j **then**
      FU_i is already existed in the universal FOON.
      continue to search next functional unit in new subgraph
    else
      Add FU_i to the universal FOON
      Add input object state nodes of FU_i to node list
      Add input assembly state node nodes of FU_i to node list
      Add output object state nodes of FU_i to node list
      Add output assembly state node of FU_i to node list
(The connections are rebuilt in the task retrieval process)

III. FOON CONSTRUCTION FROM NON-EXPERT DEMONSTRATIONS

In assembly tasks, parts and their corresponding attributes, such as mass, color, shape, etc. are usually well-defined. Using object tracking and human hand tracking, the velocities and locations of parts and the sequence of human manipulations can be obtained through human demonstrations. Therefore, FOON for assembly tasks can be learned from human demonstrations. However, the human demonstrations of non-expert end-users can be inefficient. For example, humans may first place a part at an incorrect location and then fix it, which introduces unnecessary manipulations into the demonstration. Similarly, humans may accomplish assembly tasks with an unoptimized manipulation sequence, which increases the usage of manipulations with lower success rates or efficiency. To eventually get an efficient solution, we first need to learn the assembly task representation using FOON based on the non-expert demonstration.

A. Creating Subgraphs

For each assembly task, multiple rounds of human demonstrations are performed by different non-expert users. Each round of human demonstration automatically generates a list of functional units based on object tracking and human hand tracking. The process is also recorded as an instructional video online. The corresponding object state nodes, assembly state nodes for both input and output, and the motion node of each functional unit are manually verified according to the instructional video. These functional units are then connected and combined into a subgraph automatically. The subgraph of FOON is then visualized and verified manually. Each subgraph represents the structured knowledge of an overall process of an assembly task. However, each process may be inefficient since it is demonstrated by a non-expert user.

B. Merging Subgraphs

The FOON for assembly can be expanded by merging new subgraphs generated by different human demonstrations of different assembly tasks. The merging algorithm is described in Algorithm 1. Two functional units are regarded as equal if

and only if the set of nodes in one functional unit is exactly the same as the other functional unit. The FOON is first empty; the subgraph of each round of human demonstration for each assembly task is merged into the universal FOON in sequence. For each functional unit in the subgraph, if it does not exist in the universal FOON, it will be added to the universal FOON. The merged FOON contains the structured knowledge from multiple rounds of non-expert demonstrations for multiple assembly tasks, which gives the potential to robots to find the optimized efficient solution for each assembly task. The functional units in the universal FOON can be further connected according to their input and output nodes. For each output node of a functional unit, it might connect to the same input node of other functional units. The connections are rebuilt in the following task retrieval process.

IV. ASSEMBLY TASK RETRIEVAL AND OPTIMIZATION FROM NON-EXPERT DEMONSTRATIONS

In addition to FOON being a knowledge representation obtained or learned from human demonstrations, a FOON can also be used by robots for problem-solving and process optimization for assembly tasks. Given the goal of an assembly task, robots can search for all possible solutions and then choose the most efficient solution for the assembly task based on all the assembly tasks learned from human demonstrations. As mentioned in the previous section, each subgraph of FOON corresponds to a single round of human demonstration of an assembly task. Robots can at least choose the most efficient subgraph from the original non-expert demonstrations. Additionally, when multiple rounds of demonstrations of multiple assembly tasks are merged as a universal FOON, it is possible to find more efficient subgraphs other than the original demonstrations for assembly tasks.

A. Retrieving Assembly Task Tree

Once multiple non-expert demonstrations for multiple assembly tasks have been conducted, a universal FOON can be established by merging subgraphs generated by each round of human demonstrations using Algorithm 1. In order to find the optimized solution for a task, we need to find all the possible assembly processes based on the universal FOON. Most existing symbolic planning algorithms focus on solving the planning problems that are represented using planning domain definition languages. Since the proposed FOON for assembly tasks uses a different representation, we will need to develop a corresponding planning algorithm for it. Each assembly process is a combination of functional units, which gives the path from an initial condition to the goal of an assembly task. The algorithm to retrieval all possible assembly task processes is shown in Algorithm 2.

First, we give a goal assembly state node N_{goal} to the robot. All the nodes, which contain N_{goal} as an output assembly state node, in the universal FOON are appended to a list of root tree nodes R. Over each root node r_i in R, it is possible to generate multiple task tree paths, which can accomplish the given assembly task. Starting from the root tree node, we iterate for each input object state node of the corresponding functional unit and search for the functional units which can produce it. When we search for the dependencies of a functional unit, we define this functional unit as the *head.* For an input object state node N_{input} of the head, if a functional unit contains it as an output object state node and it is not an ancestor of the head, then this functional unit is regarded as a dependency of the head for the input object node N_{input}. All of the functional units that produce N_{input} are regarded as candidate functional units and are added to the list of candidate functional units L_c, which is then appended to a list of the preliminary tree nodes L_p. This step proceeds until the tree stack is empty; at this point, the list L_p covers the functional units for the dependencies for all the object state inputs. To accomplish the given assembly task, we only need one functional unit to meet the dependency for each input object state node. Thus, we compute the Cartesian product of the list L_p. Each product set of functional units will contain a whole path that meets object state input requirements of the corresponding root. By conducting a breadth-first search $BFS(r_i)$ with respect to the root r_i, we can obtain one assembly process to accomplish the assembly task. By iterating for each product set, we can obtain all possible assembly processes for the given assembly goal from the universal FOON.

Algorithm 2: Retrieval of Assembly Processes

```
Let N_goal be the goal assembly state node.
Let R be the list of root tree nodes.
Let P_all be the list of all possible assembly process.
Initialize R and P_all as empty list.
for all functional units FU_i in the universal FOON do
  if N_goal is the output assembly state node of FU_i then
    Add FU_i to R.
  end if
end for
for all root nodes r_i in R do
  Initialize a tree stack TS.
  Initialize a prelim tree node list L_p
  Append the root node r_i to TS.
  Append the root node r_i to L_p.
  while the tree stack TS is not empty do
    Pop the functional unit FU_h from the right side of TS.
    Set the head of search h to FU_h.
    for all input object state nodes N_input do
      Initialize a list of candidate functional units L_c
      for all functional units FU_i in the universal FOON do
        if N_input is one of the output object state nodes and FU_i is not
        an ancestor of the head h then
          Set FU_i as a child of the head h
          Set h as the parent of the functional unit FU_i
          Append FU_i to candidate list L_c
        end if
      end for
      Append candidate list L_c to prelim tree node list L_p
    end for
  end while
  Let P_d to be the cartesian product of L_p.
  for each dependent tree path d_p in P_d do
    for all task paths p found by breath-first-search BFS(r_i) do
      Append path p to P_all.
    end for
  end for
for all path in P_all do
  Calculate the integrated weight of the path
return optimized task path p*
```

B. Robot Execution Optimization

Once all possible assembly processes are determined, the optimal solution for the given assembly task is determined by the integrated weight of the assembly process. For an assembly process consisting of N functional units and weights W_i (i=1,2,…, N) for each function unit, the integrated weight of the assembly process can be written as

$$W_I = \prod_{i=1}^{N} W_i \quad (2)$$

The optimal assembly process can be determined by

$$p^* = \underset{W_I}{argmax}(P_{all}) \quad (3)$$

where P_{all} is the set of all possible assembly processes of the given assembly task. Once the optimized task-level assembly process p^* is determined using FOON and the corresponding task retrieval algorithm, for each motion in p^*, the robot will search for the trajectory with minimum execution time among all the taught trajectories of the motion based on the situation of the workspace.

V. EXPERIMENTAL RESULTS AND ANALYSIS

To evaluate the proposed approaches, different non-expert demonstrations for different assembly tasks are conducted. In this section, we first present the experimental setup. The assembly tasks and the corresponding non-expert demonstrations used in our experiments are then explained respectively. Based on the demonstrations, the results of robot learning and task retrieval and optimization are discussed.

A. Experimental Setup

The proposed approaches are verified and evaluated on a multi-model human-robot collaborative assembly test platform. The hardware setup of the test platform is illustrated in Fig. 2 (a). In our experiments, we use the ABB Yumi, which is a dual-arm collaborative robot. The human stands face to face with the robot to work in a shared workspace. The Kinect RGB-D sensor offers a top-view point cloud of the workspace for part recognition and tracking. The software of the test platform is developed based on the Robot Operating System (ROS) and visualized through Rviz. The trajectory-level motion planning from point to point is accomplished based on the Open Motion Planning Library (OMPL) via MoveIt! motion planning framework.

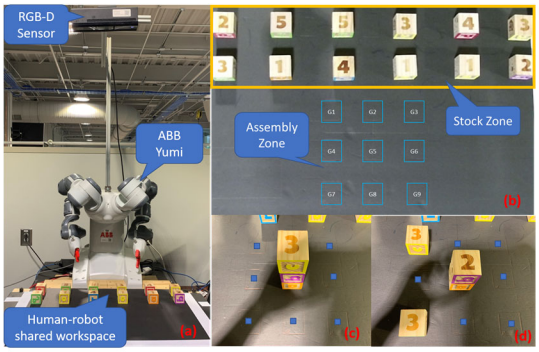

Fig. 2. Experimental setup. (a) The configuration of the robot and RGB-D sensor. (b) The top view of the human-robot shared workspace captured by the RGB-D sensor. (c) The final state of the stacking task. (d) The final state of the shape constructing task.

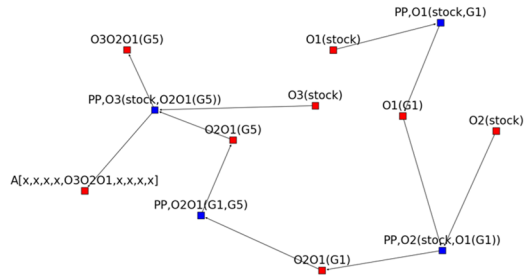

Fig. 3. The subgraph obtained from the first-round demonstration of the stacking task.

B. Subgraphs of Non-expert demonstrations

In our experiments, we use blocks with different numbers to represent different types of parts (O1 to O5). Initially, there are 12 parts in total located at different given locations in the stock zone. A 3 x 3 grid (G1 to G9) is defined as an assembly zone on the workbench. The start positions and the goal positions of pick-place and stacking robot actions between different grids are defined by assembly task in advance and the motion trajectories for the actions are generated via sampling-based motion planning algorithms in MoveIt! motion planning framework. The motions and corresponding object states are learned via human demonstrations. The observed motions are mapped to corresponding elements in the action set according to the corresponding grid locations. The corresponding robot action assignment based on the observed motions is executed by sending the corresponding start position and the goal position and calling the sampling-based motion planner to realize the action. Two types of assembly tasks are performed: the stacking task and the shape constructing task. The final assembly states of both tasks and the indexes of the grid locations are shown in Fig. 2 (c) and Fig. 2 (d). Three non-expert demonstrations are conducted for each task.

For the stacking task, the goal is to build a 3-level stack (O3O2O1) at G5. The process of the first demonstration (Fig. 3) is picking an O1 from stock and placing it at G1; then picking an O2 from stock and placing it onto the O1 at G1; then picking O2O1 together and moving them from G1 to G5; picking an O3 from stock and stacking it on the O2O1 at G5. This demonstration is not efficient because the human stacked the first two parts at an improper location and then fixed it by moving the component to the correct location. The process of the second non-expert demonstration (Fig. 4) is picking an O2 from stock and placing it to G5; then picking an O1 from stock and stacking it on the O2; then switching the positions of O1

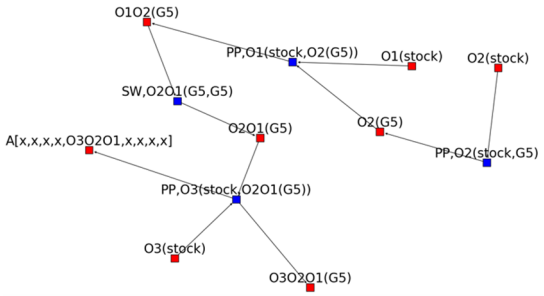

Fig. 4. The subgraph obtained from the second-round demonstration of the stacking task.

and O2; finally picking an O3 from stock and stacking it on the O2 at G5. This demonstration is inefficient since an extra dual-arm manipulation was conducted to fix the order of the stack. The process of the third demonstration (Fig. 6) is picking an O3 from stock and placing it to G8; then picking an O2 from stock and stacking it on the O2 at G8; then switching the positions of O2 and O3; then picking an O1 from stock and placing it to G5; finally picking the component (O3O2) from G8 and stacks it on the O1 at G5. This demonstration is less efficient than the previous two

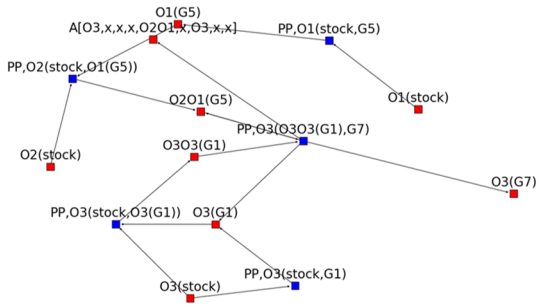

Fig. 5. The subgraph obtained from the first-round demonstration of the shape constructing task.

demonstrations since it contains five manipulations, including

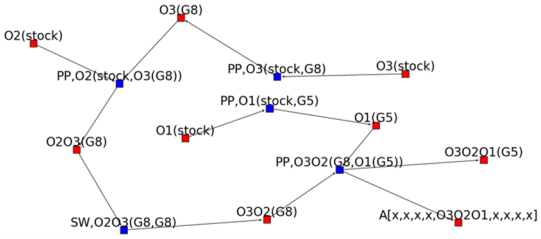

Fig. 6. The subgraph obtained from the third-round demonstration of the stacking task.

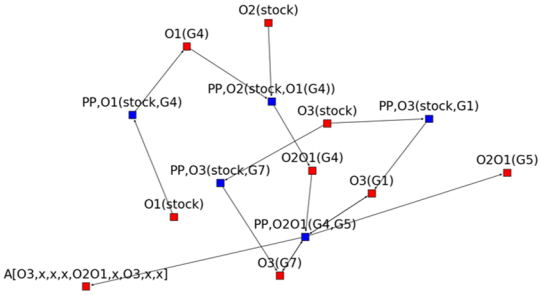

Fig. 7. The subgraph obtained from the second-round demonstration of the shape constructing task.

one dual-arm manipulation, to accomplish a three-parts stacking task.

The shape constructing is to build a specific 3D shape in the 3 x 3 grid. The goal is to have a block O3 at both G1 and G7 and have a block O2 on the top of a block O1 at G5. Similar to the stacking task, we also conducted three non-expert demonstrations for the shape constructing task. The process of the first demonstration (Fig. 5) is stacking two O3 at G1; then picking an O1 from stock and placing it to G5; then picking an O2 from stock and stacking it onto the O1 at G5; finally moving the O3 at the top of the component O3O3 to G7. This demonstration is inefficient because of the error pick-place action in the second step, which is fixed in the last step. The process of the second non-expert demonstration

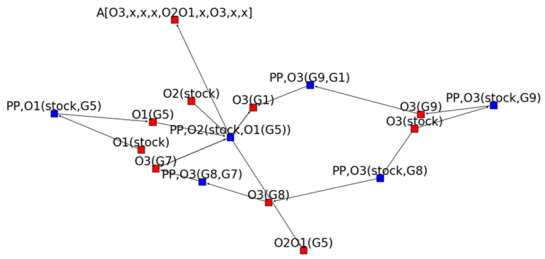

Fig. 8. The subgraph obtained from the third-round demonstration of the shape constructing task.

(Fig. 7) is picking an O3 and placing it to G1; then building the component O2O1 via two pick-place actions at G4; then picking another O3 from stock and placing it to G7; finally moving the O2O1 from G4 to G5. The process of the third non-expert demonstration (Fig. 8) is picking an O3 from stock and placing it to G9; then picking another O3 from stock and

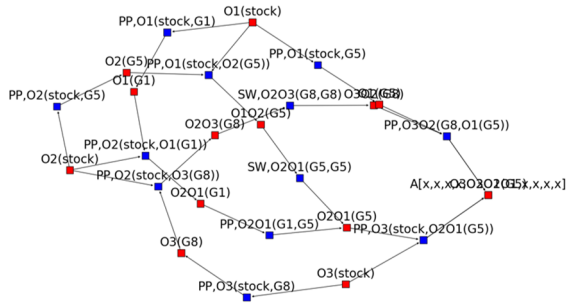

Fig. 9. The merged graph obtained from the three demonstrations of the stacking task.

placing it to G8; then moving them to G1 and G7 respectively; afterward, a block O1 and a block O2 are picked from stock and stack at G5 to accomplish the task.

C. Results of FOON Generation and Optimal Assembly Task Tree Retrieving

To verify the proposed approaches, we merge the subgraphs generated by the non-expert demonstrations progressively to establish the universal FOON. For each stage, we compare the optimal assembly task process we can obtain based on the present situation of the robot knowledge.

In the first stage, the universal FOON was only built based on the three non-expert demonstrations of the stacking task is shown in Fig. 9. It contains 12 functional units. Based on this merged FOON, the assembly process retrieval algorithm finds the same as the three non-expert demonstrations. The integrated weights of these three demonstrations are 0.478, 0.050, and 0.032 respectively. The results indicate that the robot will implement the stacking task by repeating the first human demonstration at this stage of merged FOON. The real robot execution of the stacking task is shown in Fig. 10. Also,

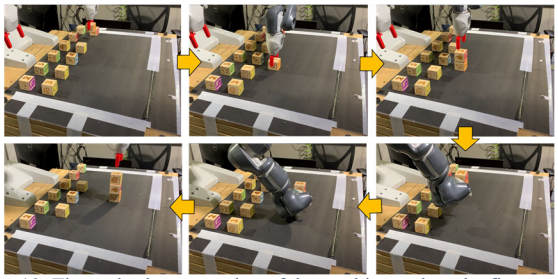

Fig. 10. The real robot execution of the stacking task at the first stage.

the robot cannot find a solution for the shape formatting task since none of its demonstrations has been integrated into the merged FOON yet.

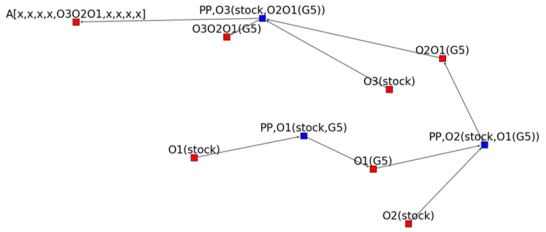

Fig. 11. The optimized result of the stacking task.

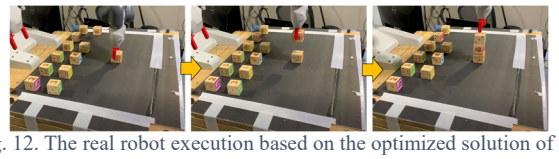

Fig. 12. The real robot execution based on the optimized solution of the stacking task.

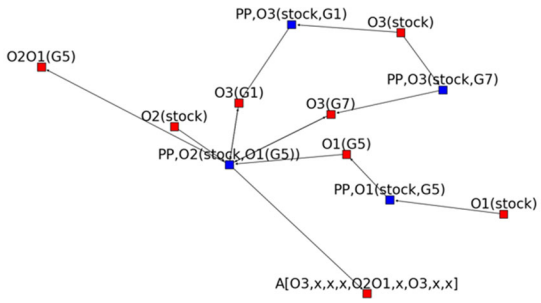

Fig. 13. The optimized solution for the shape constructing task.

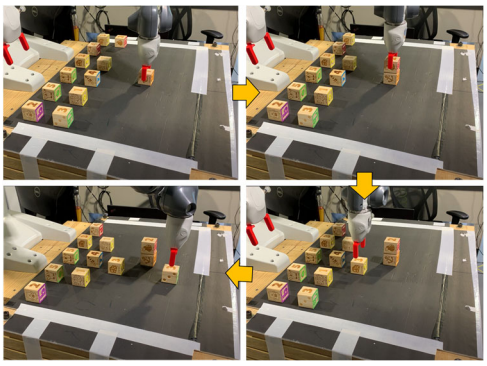

Fig. 14. The real robot execution of the optimized strategy for the shape constructing task.

In the second stage, the universal FOON was built based on the three non-expert demonstrations of the stacking task and the first non-expert demonstration of the shape constructing task. For the stacking task, the robot can find an optimized assembly process with three single-arm-single-part manipulations as shown in Fig. 11, which is to pick up a block O1, a block O2, and a block O3 from stock and stack them at G5 in sequence. The corresponding integrated weight of this assembly process is 0.625, which is higher than all the non-expert demonstrations of the stacking task. The real robot execution of the optimized strategy on the stacking task is shown in Fig. 12. For the shape constructing task, the first-round human demonstration is successfully reconstructed as the solution of the task at this stage of the merged FOON.

In the third stage, we merged all the six non-expert demonstrations for both tasks to construct a universal FOON. The result of the universal FOON contains all six non-expert demonstrations of both the stacking task and shape constructing task. At this stage, the optimized result for the stacking task is also found as same as the three-motions solution shown in Fig. 11. For the shape constructing task, an optimized solution with four single-arm-single-part manipulations is found, which is to directly pick up a block O1, a block O2, and two O3 blocks and place them to their corresponding locations in sequence. The subgraph of the optimized solution of the shape constructing task is shown in Fig. 13, and the robot execution of this optimized strategy is shown in Fig. 14.

D. Evaluations

The summary of the assembly process optimization of the three stages of the universal FOON is illustrated in Table I. The results of the raw non-expert demonstrations and the optimized results of the stacking and the shape constructing task are shown in Table III and Table II respectively. The universal FOON built based on the three demonstrations of the stacking task contains 12 functional units and 13 object state nodes in total. Based on this universal FOON, we can find the optimized solution for the stacking task with an overall success rate of 0.693 and an overall efficient weight of 0.690. By adding a non-expert demonstration of the shape constructing task, the universal FOON contains 16 functional units and 16 object state nodes. With the updated universal FOON with more knowledge of assembly tasks, we can obtain a better-optimized solution for the stacking task with an overall success rate of 0.770 and an overall efficient weight of 0.812. According to the task design, this is already the best solution for the stacking task. Also, the robot successfully learned the shape constructing task via the additional one-round human demonstration, though it is not very efficient. After merging two additional non-expert demonstrations of the shape constructing task, a better solution for the shape constructing task is found from the universal FOON, which is

TABLE I
SUMMARY OF ASSEMBLY PROCESS OPTIMIZATION OF DIFFERENT STAGES OF UNIVERSAL FOON

Stage	Universal FOON			Optimized Stacking Assembly Process				Optimized Shape Constructing Process			
	No. of Object State Node	No. of Motion Node	No. of Functional Unit	No. of Functional Unit	Overall Success Rate	Overall Efficient Weight	Integrated Weight	No. of Functional Unit	Overall Success Rate	Overall Efficient Weight	Integrated Weight
1	13	12	12	4	0.693	0.690	0.478	None	None	None	None
2	16	16	16	3	0.770	0.812	0.625	5	0.694	0.621	0.432
3	18	24	24	3	0.770	0.812	0.625	4	0.772	0.772	0.595

built based on six in-efficient human demonstrations. Based on the proposed task, the optimal solution simply contains four single-arm-single-part manipulations, which should be the most efficient solution based on our experimental setup. Moreover, the most efficient solution for the stacking task, which has already been found in the earlier stage, is also found in this universal FOON.

Furthermore, in this paper, since we use a FOON model-based approach, we only require several rounds of non-expert demonstrations to derive the best possible solution for the task. In contrast, the existing data-driven approaches [18]–[20] usually require humans to demonstrate the task thousands of times to obtain enough training data to learn the task. Therefore, comparing to the existing data-driven approaches, the time cost and the teaching effort of human demonstrations can be significantly reduced using our proposed approach.

TABLE III
RAW DEMONSTRATIONS VS OPTIMIZED SOLUTION OF STACKING TASK

Item	Raw Demonstrations				Optimized
	1st	2nd	3rd	Average	
No. of Actions	4	4	5	4.33	3
Success Rate	0.693	0.616	0.520	0.609	0.770
Efficient Weight	0.690	0.081	0.061	0.280	0.812
Integrated Weight	0.478	0.050	0.032	0.187	0.625

TABLE II
RAW DEMONSTRATIONS VS OPTIMIZED SOLUTION OF SHAPE CONSTRUCTING TASK

Item	Raw Demonstrations				Optimized
	1st	2nd	3rd	Average	
No. of Actions	5	5	6	5.33	4
Success Rate	0.694	0.694	0.696	0.695	0.772
Efficient Weight	0.621	0.656	0.696	0.658	0.772
Integrated Weight	0.432	0.456	0.485	0.457	0.595

VI. CONCLUSION

This paper introduces a graph-based task representation approach based on the Functional Object-Oriented Network (FOON) to represent the knowledge of assembly tasks. It creates algorithms to create a FOON from multiple non-expert assembly demonstrations and also develops an assembly task tree retrieving approach with a robot execution optimization process to generate the best possible task execution plan from the FOON. The results indicate that robots can find the best possible assembly process among multiple rounds of non-expert demonstrations. The evaluation also indicates the effectiveness and advantages of the proposed approach compared to other existing approaches.

REFERENCES

[1] S. Ekvall, "Robot task learning from human demonstration." KTH, 2007.

[2] D. Paulius and Y. Sun, "A survey of knowledge representation in service robotics," *Rob. Auton. Syst.*, vol. 118, pp. 13–30, 2019.

[3] Y. Wu, Y. Su, and Y. Demiris, "A morphable template framework for robot learning by demonstration: Integrating one-shot and incremental learning approaches," *Rob. Auton. Syst.*, vol. 62, no. 10, pp. 1517–1530, 2014.

[4] E. L. Sauser, B. D. Argall, G. Metta, and A. G. Billard, "Iterative learning of grasp adaptation through human corrections," *Rob. Auton. Syst.*, vol. 60, no. 1, pp. 55–71, 2012.

[5] L. Peternel and J. Babič, "Humanoid robot posture-control learning in real-time based on human sensorimotor learning ability," in *2013 IEEE international conference on robotics and automation*, 2013, pp. 5329–5334.

[6] S. Kim, C. Kim, B. You, and S. Oh, "Stable whole-body motion generation for humanoid robots to imitate human motions," in *2009 IEEE/RSJ International Conference on Intelligent Robots and Systems*, 2009, pp. 2518–2524.

[7] Y. Huang and Y. Sun, "A dataset of daily interactive manipulation," *Int. J. Rob. Res.*, vol. 38, no. 8, pp. 879–886, 2019.

[8] H. Cheng and H. Chen, "Learning from demonstration enabled robotic small part assembly," in *2014 9th IEEE Conference on Industrial Electronics and Applications*, 2014, pp. 395–400.

[9] W. Wang, R. Li, Y. Chen, Z. M. Diekel, and Y. Jia, "Facilitating Human-Robot Collaborative Tasks by Teaching-Learning-Collaboration From Human Demonstrations," *IEEE Trans. Autom. Sci. Eng.*, no. 99, pp. 1–14, 2018.

[10] Y. Wang, R. Xiong, L. Shen, K. Sun, J. Zhang, and L. Qi, "Towards learning from demonstration system for parts assembly: A graph based representation for knowledge," *4th Annu. IEEE Int. Conf. Cyber Technol. Autom. Control Intell. Syst. IEEE-CYBER 2014*, pp. 174–179, 2014.

[11] J. Ibrahim and P. Plapper, "Contact-state modeling of robotic assembly tasks using Gaussian mixture models," *Procedia Cirp*, vol. 23, pp. 229–234, 2014.

[12] G. Thomas, M. Chien, A. Tamar, J. A. Ojea, and P. Abbeel, "Learning robotic assembly from cad," in *2018 IEEE International Conference on Robotics and Automation (ICRA)*, 2018, pp. 1–9.

[13] Y. Wang, Y. Jiao, R. Xiong, H. Yu, J. Zhang, and Y. Liu, "MASD: A Multimodal Assembly Skill Decoding System for Robot Programming by Demonstration," *IEEE Trans. Autom. Sci. Eng.*, vol. 15, no. 4, pp. 1722–1734, 2018.

[14] Y. Huang and Y. Sun, "Learning to pour," in *2017 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, 2017, pp. 7005–7010.

[15] A. Garland and N. Lesh, "Learning hierarchical task models by demonstration," *Mitsubishi Electr. Res. Lab. (MERL), USA–(January 2002)*, pp. 51–79, 2003.

[16] P. E. Rybski, K. Yoon, J. Stolarz, and M. M. Veloso, "Interactive robot task training through dialog and demonstration," in *Proceedings of the ACM/IEEE international conference on Human-robot interaction*, 2007, pp. 49–56.

[17] A. Mohseni-Kabir *et al.*, "Simultaneous learning of hierarchy and primitives for complex robot tasks," *Auton. Robots*, vol. 43, no. 4, pp. 859–874, 2019.

[18] Y.-H. Wu, N. Charoenphakdee, H. Bao, V. Tangkaratt, and M. Sugiyama, "Imitation learning from imperfect demonstration," *arXiv Prepr. arXiv1901.09387*, 2019.

[19] Y. Gao, H. Xu, J. Lin, F. Yu, S. Levine, and T. Darrell, "Reinforcement learning from imperfect demonstrations," *arXiv Prepr. arXiv1802.05313*, 2018.

[20] A. Skrynnik, A. Staroverov, E. Aitygulov, K. Aksenov, V. Davydov, and A. I. Panov, "Hierarchical deep q-network from imperfect demonstrations in minecraft," *Cogn. Syst. Res.*, vol. 65, pp. 74–78, 2019.

[21] D. Paulius, Y. Huang, R. Milton, W. D. Buchanan, J. Sam, and Y. Sun, "Functional object-oriented network for manipulation learning," in *2016 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, 2016, pp. 2655–2662.

[22] D. Paulius, A. B. Jelodar, and Y. Sun, "Functional object-oriented network: Construction & expansion," in *2018 IEEE International Conference on Robotics and Automation (ICRA)*, 2018, pp. 1–7.