INVITED PAPER

Learning-Based Automation of Robotic Assembly for Smart Manufacturing

This article proposes an approach to endowing robots with the capability of autoprogramming of assembly tasks with minimal human assistance that is based on "learning from observation" and "robotic embodiment."

By Sanghoon Ji, Sukhan Lee, *Life Fellow IEEE*, Sujeong Yoo, Ilhong Suh, *Fellow IEEE*, Inso Kwon, *Member, IEEE*, Frank C. Park, *Fellow IEEE*, Sanghyoung Lee, and Hongseok Kim

ABSTRACT | For smart manufacturing, an automated robotic assembly system built upon an autoprogramming environment is necessary to reduce setup time and cost for robots that are engaged in frequent task reassignment. This article presents an approach to the autoprogramming of robotic assembly tasks with minimal human assistance. The approach integrates "robotic learning of assembly tasks from observation" and "robotic embodiment of learned assembly tasks in the form of skills." In the former, robots observe human assembly operations to learn a sequence of assembly tasks, which is formalized into a human assembly script. The latter transforms the human assembly script into a robot assembly script in which a sequence of robot-executable assembly tasks are defined based on action planning supported by workspace modeling and simulated retargeting. The assembly tasks, in the form of the robot assembly script, are then implemented via pretrained robot skills. These skills aim to enable robots to execute difficult tasks that involve inherent uncertainties and variations. We validate the proposed approach by building a prototype of the automated robotic assembly system for a power breaker and an electronic set-top box. The results verify that the proposed automated robotic assembly system is not only feasible but also viable, as it is associated with a dramatic reduction in the human effort required for automating robotic assembly.

KEYWORDS | Automated robot programming; learning by observation; smart manufacturing.

I. INTRODUCTION

As the manufacturing industry trends toward smaller batch production, production lines rely more and more on flexible or smart work cells. In such a smart cell-based production line, industrial robots are required to work in an environment, which is more complex, less structured, and subject to more frequent changes than in a standard production line. Furthermore, the robots should be able to collaborate among themselves and with human workers. This situation represents a sharp deviation from a conventional mass production line where robots carry out simple and fixed routines under highly structured workspace environments.

However, there is a key challenge to implementing smart cell-based production lines: the time and effort required to set up the robots engaged in sophisticated tasks, either independently or cooperatively, in such a minimally

Manuscript received January 23, 2021; accepted February 17, 2021. Date of current version March 23, 2021. This work was supported in part by the "Robot Industry Fusion Core Technology Development Project" of the Korea Evaluation Institute of Industrial Technology (KEIT), sponsored by the Korea Ministry of Trade, Industry and Energy (MOTIE), under Grant KEIT 10048320, in part by the AI Graduate School Program under Grant 2019-0-00421, and in part by the ICT Consilience Program of the Institute of Information and Communication Technology Planning & Evaluation (IITP), sponsored by the Korean Ministry of Science and Information Technology (MSIT), under Grant IITP-2020-0-01821. *(Corresponding author: Sukhan Lee.)*

Sanghoon Ji and **Sujeong Yoo** are with the Convergence Research Center, Korea Institute of Industrial Technology, Ansan 15588, South Korea (e-mail: robot91@kitech.re.kr; sjyou21@kitech.re.kr).

Sukhan Lee is with the Department of Information and Communication Engineering, Sungkyunkwan University, Suwon 16419, South Korea (e-mail: lsh1@skku.edu).

Ilhong Suh is with the Department of Electronic and Computer Engineering, Hanyang University, Seoul 04763, South Korea (e-mail: koreaihsuh@hanyang.ac.kr).

Inso Kwon is with the Robotics & Computer Vision Lab, Korea Advanced Institute of Science and Technology, Daejeon 34141, South Korea (e-mail: iskweon77@kaist.ac.kr).

Frank C. Park is with the Department of Mechanical Engineering, Seoul National University, Seoul 08826, South Korea (e-mail: fcp@snu.ac.kr).

Sanghyoung Lee is with the Department of Smart Manufacturing Innovation, Korea Institute of Industrial Technology, Cheonan 31056, South Korea (e-mail: zelog@kitech.re.kr).

Hongseok Kim is with the Department of Electronics Engineering, Kookmin University, Seoul 02707, South Korea (e-mail: kimhongseok@kookmin.ac.kr).

Digital Object Identifier 10.1109/JPROC.2021.3063154

structured smart cell are expected to grow exponentially. This challenge increases the demand for "leveling up" of automation to set up the robots in smart cell-based production lines. This boils down ultimately to the problem of how to automate the programming of the robots in such a way that the robots can carry out the given task automatically with the minimum human input.

For instance, although many cooperative robots have recently been developed and released to support the flexibility of work cells, there is an immediate need for further improvement in the convenience of teaching the robots to accomplish a task in a more direct and intuitive manner. This means that, in order to achieve simultaneous improvements in productivity and flexibility based on robotized smart work cells, it is critical to increase the level of automation in programming and robotic execution. As such, researchers in the field of smart manufacturing are increasingly interested in applying artificial intelligence (AI) to automating robot work planning and control.

The process of robotizing a target process by introducing or adding an industrial robot to a work cell consists of two steps. The first step comprises the preoperations, such as programming (P), teaching (T), and parameter tuning (PT), while the second step comprises the postoperations needed for adaptation to a real environment. The preliminary work (P, T, PT), which is performed by human workers, takes about three to seven months depending on the complexity of the target process. Thus, there exists an urgent need for improvement in the efficiency of the prework. In particular, the current practice of manual work of (P, T, PT) by automation specialists should be replaced by a new paradigm of data-based automation for which AI is expected to play a major role.

Note, however, that, for practical purposes, it may be necessary to consider potential obstacles to the use of AI-based automated systems in real manufacturing practices. For instance, major robot makers may be reluctant to allow their robots to be programed and controlled by external sources. Therefore, this article proposes an AI-based automated assembly system that guides the robots in an assembly smart work cell to automatically create work plans by observing human demonstrations of the work process and content. The work plan includes the assembly order, the sequence of unit assembly operations, the target pose of assembly objects, and the skills required to complete the unit assembly operations. Although challenging, the development of the proposed AI-based automated assembly system is supported by the recent emergence of two significant technological advancements: 1) deep learning (DL)-based real-time segmentation, modeling, and understanding of static and dynamic scenes and 2) an increase in the power of reinforcement learning (RL) for implementing robot skills.

A. Related Work

As a means of improving the prework (P, T, PT) for robotic assembly, the "BAXTER" robot [1] simplified P and T by eliminating P altogether. The process injection is made easier by the use of direct teaching to automate Nemec *et al.* [4] developed a new type of autonomous robot work cell, the "Reconcell," in which the robot learns skills for polishing and grinding processes from human demonstrations. "Reconcell" was designed for both large and small production lines [3]. Niekum *et al.* [6] developed a robot that assembles IKEA desks, for which they define the unit tasks that the robot can perform with the skills the robot learns. The furniture is assembled through a combination of the unit works represented as a designated sequence.

Diankov proposed an AI-based platform, "OpenRAVE" [5], for the development of robot programs. "OpenRAVE" carries out automatic task programming using the minimum available information on tasks and robots, including CAD files, sensor data, movement constraints, grip/contact constraints, an obstacle map, and robot-related information, such as robot kinematics, grippers, bases, sensors, controllers, and H/W interfaces. Dömel *et al.* [30] presented a modular software system that autonomously solves industrial manipulation tasks. A number of modules have been developed to represent knowledge on the state of the world, the objects to be handled, and the assembly processes such that, given the task and environmental requirements, they are flexibly organized through hierarchical flow control to perform autonomous mobile manipulation. Meanwhile, Syddansk Universitet, in the IntellAct project [2], developed a technology that allows robots to automatically learn semantic tasks based on an understanding of scenes and human actions.

Automatic manipulation of industrial objects requires object identification, segmentation, and poses estimation. "Kinema pick" [31] can recognize the shape and pose of various-sized boxes and determine the robotic motions that are necessary to pick them up and place them on a conveyor belt in real time. Deep Sliding Shapes [32] was proposed as a method of simultaneously performing object detection and object pose estimation based on an RGB-D camera. This method consists of DL networks that estimate a 3-D region, identifies the objects in that region, and computes their poses. MaskTrack ConvNet [33] was presented as a method of obtaining a highly accurate representation of object segmentation from videos. In it, a DL network performs segmentation for the current frame based on the segmentation result of the previous frame and the image in the current frame as the input.

B. Problem Statement and Proposed Approach

In this article, we address the following problem: how to reduce the setup time and cost for reconfiguring and reprogramming robots engaged in a minimally structured smart assembly cell, so as to effectively deal with the frequent reassignment of assembly tasks of increasing sophistication. To deal with this problem, we recognize the need

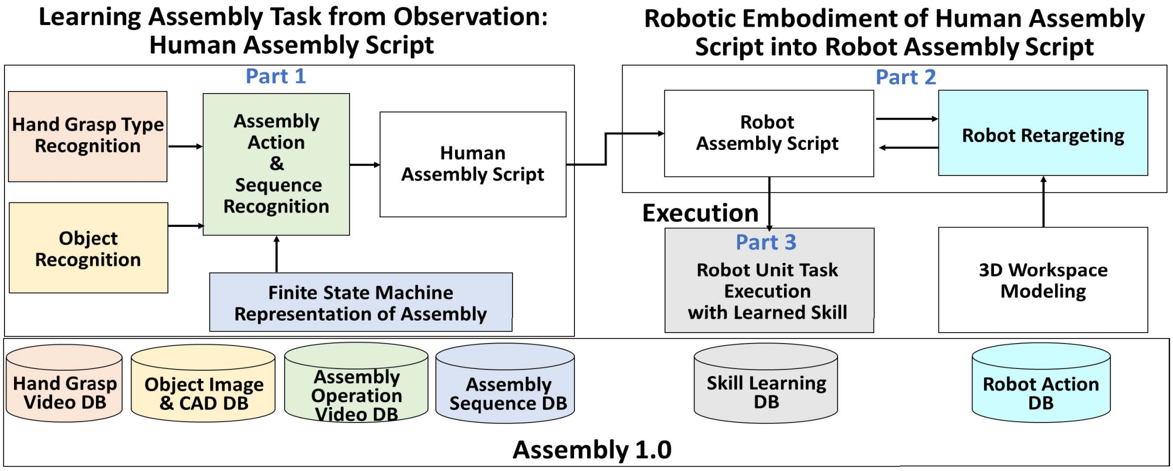

Learning Assembly Task from Observation: Human Assembly Script
Part 1
Hand Grasp Type Recognition
Object Recognition
Assembly Action & Sequence Recognition
Human Assembly Script
Finite State Machine Representation of Assembly
Robotic Embodiment of Human Assembly Script into Robot Assembly Script
Part 2
Robot Assembly Script
Robot Retargeting
Execution
Part 3
Robot Unit Task Execution with Learned Skill
3D Workspace Modeling
Hand Grasp Video DB
Object Image & CAD DB
Assembly Operation Video DB
Assembly Sequence DB
Skill Learning DB
Robot Action DB
Assembly 1.0

Fig. 1. ***Overview of the proposed learning-based automated assembly system with three parts and supporting DB.***

to upgrade the level of automation in the assembly line by endowing robots with the capacity for autoprogramming under minimal human input. To this end, we propose a robot autoprogramming environment that involves the integration of robotic learning of assembly tasks from observing human assembly and robotic embodiment of learned assembly tasks with pretrained skills. Furthermore, we demonstrate how recent advancements in deep and RL can impact the quality of implementation of the proposed approach.

II. SYSTEM OVERVIEW

The proposed approach to building an autoprogramming environment for robots engaged in a smart assembly cell consists of three parts: 1) robotic learning of assembly tasks through observation of the human assembly process, which results in a high-level description of learned assembly tasks as a formal human assembly script; 2) automatic transformation of the human assembly script into a robot assembly script, in which a sequence of robot-executable tasks is specified based on robot action planning supported by workspace modeling and simulated retargeting, referred to here as “robotic embodiment;” and 3) automatic execution of the sequence of robot-executable tasks defined in the robot assembly script with pretrained robot skills. The pretrained skills allow the robots to adapt to the uncertainties and variations present in real assembly environments. Refer to Fig. 1 for an illustration of how the three parts described above function to accomplish the assembly objective of the system.

To define the sequence of assembly tasks from observations, the first part performs DL-based recognition of a sequence of human assembly actions and of the grasping mode of the hand, in addition to the recognition of the objects involved in those assembly actions, from captured video images. To this end, we collect large-scale human assembly data sets to train and test the DL networks, including annotated data sets of human assembly actions with their sequences in the assembly process and the grasping modes involved, from the real assembly of various electromechanical products. The assembly sequences are then organized into finite state machines (FSMs) as generalized representations [34], [35]. An FSM filters a sequence of assembly actions recognized by DL networks in such a way to ensure the robustness of the identified assembly sequences and to describe the assembly states associated with those actions. This process of learning a sequence of assembly tasks from observation is supported by the recognition of the objects and the types of hand grasping motions that help to identify the assembly states associated with the actions.

The learned assembly tasks with their sequences are formalized into a human assembly script in terms of the assembly actions with their orders in the assembly sequence and the state transitions, grasping modes, and objects involved.

The second part, “robotic embodiment,” aims to transform the human assembly script generated in the first part into a robot assembly script that can be executed by robots despite the differences between humans and robots in terms of their physical and perceptual capabilities. The robotic embodiment starts with planning a sequence of robot-executable actions with Planning Domain Definition Language (PDDL) [11], so as to accomplish the intended assembly tasks defined in the human assembly script. To plan the robot actions, we predefine a set of robot actions to be used in the PDDL domain. The planned robot actions are subject to simulation-based verification prior to being entered into a robot assembly script. The verification process assesses whether the planned action sequences are executable based on simulation aided by 3-D workspace modeling. If they fail, the planned action sequences are modified through retargeting and replanning. Retargeting sets up novel assembly states that may allow the robots to accomplish the given assembly tasks even though these states are not specified in a human assembly script. The additional retargeted states are then fed back into PDDL for replanning the necessary robot actions. As an example, when the robot cannot reach its target position due to obstacles, contrary to what is expected from initial planning, retargeting and replanning become necessary. In this case, retargeting and replanning should be automatically invoked to generate, for instance, an action for obstacle removal so that robot can reach the target position.

The third part implements the robot assembly script from the second part in real-world assembly cells. The implementation integrates the planned robot assembly actions with real-time recognition of the 3-D positions of the objects and tools involved. It should be noted, however, that simulation-based retargeting alone may not be sufficient for robots to accomplish the planned assembly tasks. This is especially true when the uncertainties and variations involved in assembly tasks are too great to be specified in the assembly script, as is, often, the case when an assembly task requires sophisticated motion and force trajectories, as well as interactions with objects and environments. To handle this issue, we pretrain the robots on a list of skills from which they can select to carry out real-world assembly operations. Here, such robot skills are represented by deep convolutional neural networks (DCNNs) [10], [20], in which the supervisor is capable of self-improvement through imitation learning (IL) [21]–[23] and RL [18].

A. Assembly1.0: Data Sets Collected for Implementation

For the implementation of the proposed learning-based automated assembly system, large-scale training and testing data sets are required. We have collected and annotated the required data sets into a database referred to here as Assembly1.0. Assembly1.0 consists of a video data set of human assembly action sequences for learning human assembly operations, a video data set of human object grasping actions during assembly operations, and a 3-D point cloud data set of industrial objects and, partly, their CAD files, involved in the assembly. These data sets are collected to represent typical assemblies of industrial products in real manufacturing settings. Specifically, the data set for human assembly action sequences includes such unit actions as “approach,” “reach,” “pickup,” “put,” “flip,” “plug,” “screw,” and “release,” together with the objects involved in the unit actions. On the other hand, the data set for human object grasping actions covers the right and left hands with 33 grasping types in association with eight object categories: “bolt,” “upper body,” “cable,” “copper,” “hexagon wrench,” “iron,” “bottom body,” and “busbar.” Table 1 summarizes the specifications of data sets included Assembly1.0. We plan to open Assembly1.0 publicly accessible for research purposes through the AI-Hub Korea: *https://www.aihub.or.kr/.*

Table 1 Assembly1.0 Data Set Specifications

Dataset	Specification	Size	Reference
Video for human assembly action sequences	RGB-D Kinect v2 RGB, Depth, 30Hz Front view	120 videos, 85,000 frames	Section III
Video for human object grasping actions	RGB-D Kinect v2 RGB, Depth, 30Hz Top view	60 videos, 42,500 frames	Section III
3D point cloud /CAD files of industrial objects	SKKU 3D Camera 0.1 mm Precision	60/440 objects in 15/35 categories for industry/household, 4 categories with CAD	Section IV

III. LEARNING ASSEMBLY TASKS FROM OBSERVATION: HUMAN ASSEMBLY SCRIPT

In the first part of the proposed automated assembly process, the robots learn assembly tasks by observing human workers. The objective is to extract high-level descriptions of assembly action sequences, including action classes, the order of actions in sequence, the state transitions associated with actions, the hand grasp type, and the target objects involved. As shown in Fig. 1, this part entails DL-based recognition of assembly action sequences, which is supported by the recognition of the hand grasp types and the objects involved. The assembly action sequences, thus, recognized are filtered with FSM for robustness. The results are formally represented by a human assembly script, which is then transferred to the second part, robotic embodiment.

A. Recognition of Hand Grasp Type for Assembly Intent

Recognition of hand grasp type plays an important role in identifying assembly states, as grasp types indicate how to manage objects and tools by hand [12], [13] or, in short, the assembly intent. However, it is difficult to correctly recognize the different grasp types by visually observing human hands as the hand motions involved in the assembly are quite diverse and can be obscured when holding an object. When manipulating objects, humans either grasp tools and parts or lift objects. Here, we focus on the grasp types associated with one hand to identify human assembly intent. Feix *et al.* [14] proposed a taxonomy for human grasp types in everyday life by integrating and modifying various existing grasp taxonomies. We adopt their taxonomy of 33 grasp types, of which ten are selected for use based on their strong relevance to assembly: cylindrical, spherical, palmar, tip, lateral, hook, tripod, prismatic finger, disk, and index finger extension. We also select eight categories of objects associated with the ten grasp types.

Fig. 2 illustrates the data flow of the proposed grasp type recognition, while Fig. 3 shows the detailed architecture of the grasp recognition system.

The proposed grasp type recognition system is based on two stages of processing. First, we detect all of the hands and objects in the image. Then, every possible combination

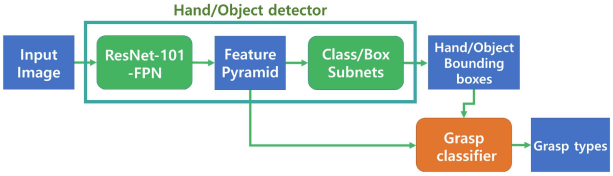


Fig. 2. ***Data flow of our grasp recognition approach.***

of hand and object that could be linked to hand grasp is examined. The detection of hands and objects is implemented by RetinaNet [15], as shown in Fig. 3, while the recognition of grasp types from the detected hands is done by a two-layer convolutional neural network (CNN) [8] with an averaging pooling layer and a fully connected layer. Note that RetinaNet consists of a backbone network of ResNet-101 [16], a feature pyramid network [17], and subnetworks for classification and box regression.

More specifically, to identify a possible combination of hand and object as a candidate for grasp type recognition, we first consider the physical distance between the two due to the fact that the hand and object should be near each other. To measure this physical distance, we scale the overlap between the respective bounding boxes of the hand and object. On the other hand, to classify the grasp type of each candidate, we apply region of interest (RoI) pooling [18] to the respective hand and object bounding boxes to generate input for the grasp classifier. Fig. 4 illustrates the two stages of the proposed grasp type recognition system. Fig. 4(a) shows an input image to be processed for grasp type recognition. Fig. 4(b) represents the result of the first stage, where all hands and objects that are present in the scene are detected with the respective bounding boxes. Note that the bounding boxes are colored to represent the types of both hand and object along with a confidence score for each. Fig. 4(c) shows the result of the second stage, demonstrating the grasp type classification for the selected hand-object candidate. In Fig. 4(c), right and left hands are differentiated using colors, and the classification probabilities of both the grasp type and the object type are marked.

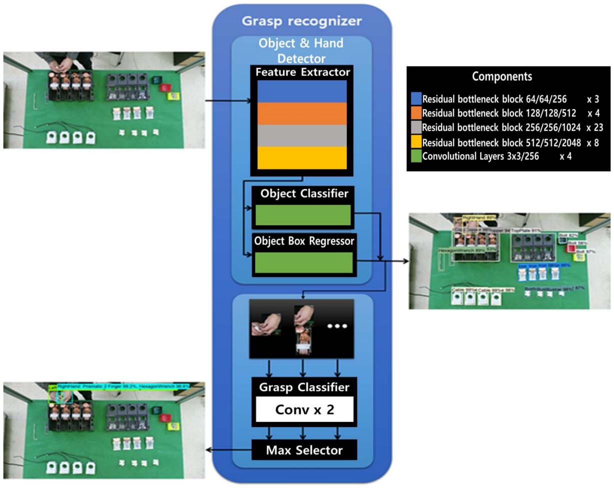

Grasp recognizer
Object & Hand Detector
Feature Extractor
Object Classifier
Object Box Regressor
Grasp Classifier
Conv x 2
Max Selector
Components
🟦 Residual bottleneck block 64/64/256 x 3
🟧 Residual bottleneck block 128/128/512 x 4
⬜ Residual bottleneck block 256/256/1024 x 23
🟨 Residual bottleneck block 512/512/2048 x 8
🟩 Convolutional Layers 3x3/256 x 4

Fig. 3. ***Architecture of the grasp recognition network.***

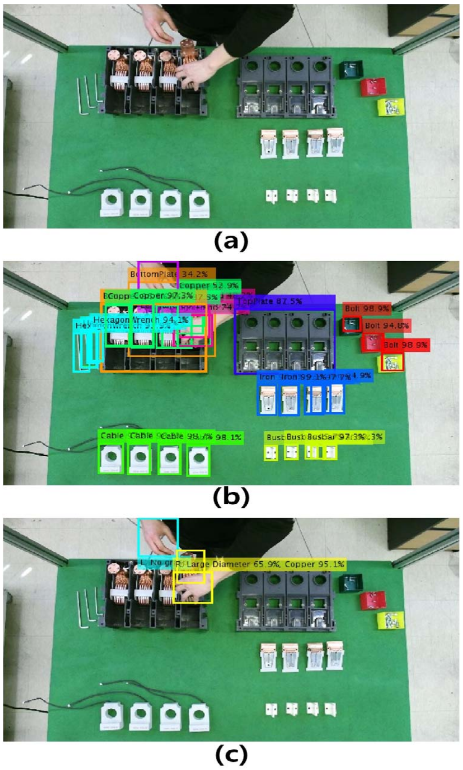


Fig. 4. ***Illustration of the proposed method of grasping mode recognition: (a) input image, (b) output of the hand/object detector, and (c) output of the grasping mode classifier.***

To evaluate the performance of the proposed grasp type recognition algorithm, we collect a hand grasp video data set from real assembly environments, consisting of videos of power breaker/air circuit breaker assembly by human workers. The video data have a resolution of 1920 × 1080 at 30 frames/s. The videos depict 15 workers repeating their assembly tasks, five times each, and each assembly task takes approximately four minutes to complete. Refer to Fig. 4 and Table 1 for the samples of collected data and for the specification of grasping action video data set, respectively. This process results in 7.768 image frames, which are divided into 5.827 frames for training, 0.388 for validation, and 1.549 for testing. The results of the performance evaluation indicate that the proposed system achieves an average accuracy of 98.26% for ten grasp types and eight object types.

B. Assembly Action Sequence Recognition

Assembly action sequence recognition is used to predict human assembly actions in the form of an action sequence based on video input, as shown in Fig. 5. To improve the adaptability of our system to various action recognition scenarios, we adopt a well-established DL model, 3-D CNN with the VGG-M architecture, as the target

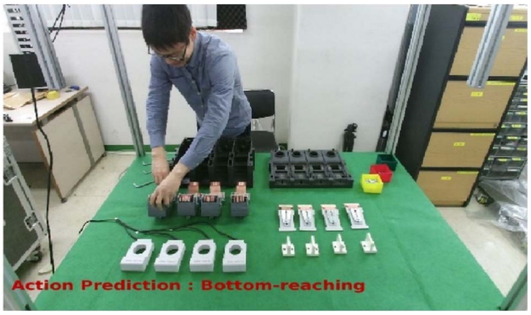


Fig. 5. ***Example of action prediction for a single frame, where the prediction result is "reaching toward the bottom part of the object."***

model. Note that, to make our system as light as possible, we configure the model with a feedforward CNN architecture without a recurrent loop and use only RGB images for action sequence recognition. This provides us with near real-time prediction performance. Furthermore, to improve the robustness of the learning process, we augment the training data by applying a random 2-D projective transformation or homography to the collected video frames. Note that the action recognition system takes the hand grasp types and the recognized objects of interest as an additional input to heighten accuracy. Accordingly, the resulting action recognition system is highly robust to various scenes from different viewpoints.

It is imperative that our DL model is both effective and efficient. One way to improve the power of the DL model is to use a deeper architecture; however, this comes at the expense of computational cost. To achieve satisfactory performance while maintaining the speed of forward processing as close to real time as possible, we leverage a machine learning paradigm called "knowledge distillation," by which the knowledge of a larger model is transferred to a lighter model. In this scenario, the larger model is referred to as the teacher network, while the lighter model is the student network. We use ResNet152 [9] pretrained on the ImageNet data set as the teacher network and 3-D CNN with the VGG-M architecture as the student network. The student network gains knowledge from the teacher network by mimicking the input–output mapping function of the teacher network. For example, the student network mimics the intermediate loss of the teacher network by backpropagating the L2 distance to the loss of the teacher network for training, as illustrated in Fig. 6. As a result, we are able to train the student network to have a high prediction accuracy while sustaining high speed. Specifically, the accuracy of action sequence prediction is improved from 91.67% without knowledge distillation to 94.43% with knowledge distillation when testing with the human assembly action sequence data set shown in Table 1. In addition, FSM helps further improve the accuracy from 94.43% to 94.72%. Knowledge distillation offers not only improvement in accuracy but also efficiency in computation. For instance, we achieve about 0.1 s or ten frames/s to process a video clip consisting of ten consecutive frames. In comparison, it takes about 0.3 s or 3.33 frames/s for the teacher network, ResNet152, to process the same video clip.

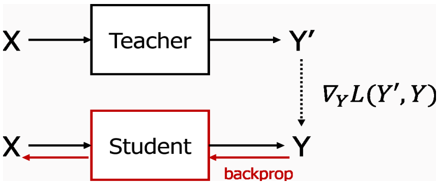


Fig. 6. ***Description of the knowledge distillation technique. The student model (our target model) learns information from the teacher model (larger model with more knowledge). Thus, our target model is trained more effectively without sacrificing computational cost.***

As stated in the system overview section, human assembly sequences can be organized into FSM as generalized representations. An FSM representation of an assembly sequence is useful for filtering the predicted action sequences in a postprocessing step. In an FSM, the current state changes to another state based on the action predicted by the action recognition system, as illustrated in Fig. 7. An FSM can be constructed from assembly sequences through learning in the form of either probabilistic or deterministic automata. The FSM is represented by a transition matrix, each element of which indicates what the next state should be either probabilistically or deterministically, given the current state and the action predicted by the DL model. For the sake of simplicity, we predefine a deterministic FSM by grammatically structuring the collected assembly sequences. The FSM offers a concise form of our prior knowledge about the possible order of the action sequences. For example, we know that, if the current state is "reaching bus-bar," the next state cannot be "pickup upper part." Fig. 7 shows an example of FSM filtering of the predicted action sequence, where the state changes only when the action prediction involves "put plate," while the prediction of "other actions" maintains the present state. In particular, to prevent an "illegal transition" from happening, we set 0 for the elements in the transition matrix that correspond to illegal transition so that FSM does not predict a wrong action sequence. In summary, FSM-based filtering of action prediction and the associated state transition help to sustain temporal consistency, alleviating the temporal flickering problem in action sequence prediction.

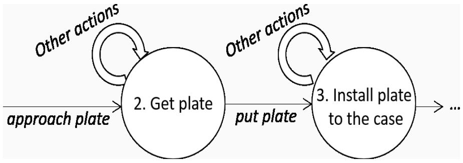
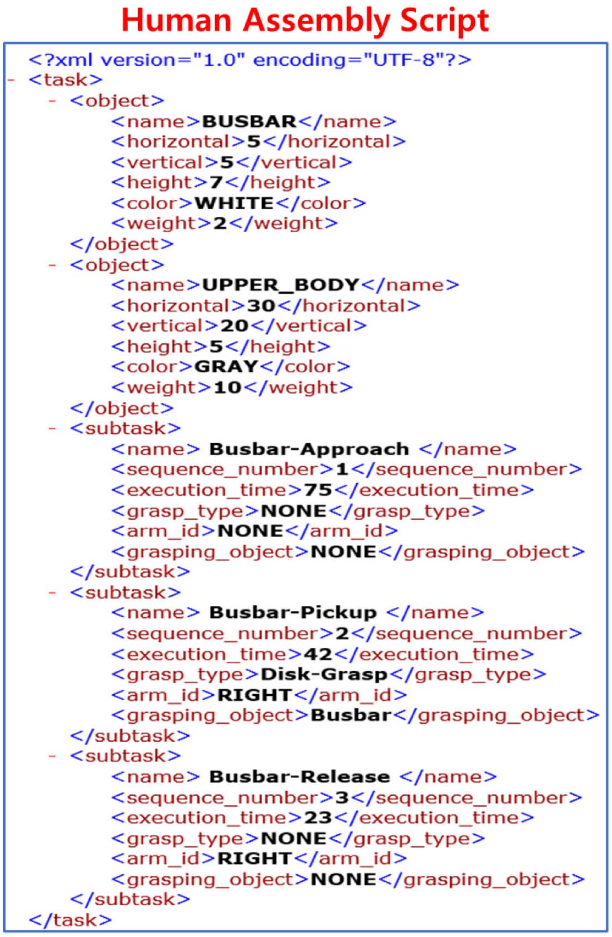


Fig. 7. ***FSM designed for refining action sequence prediction. By taking the information from the previous time step into account, our action sequence prediction becomes more robust to temporal flickering.***



Human Assembly Script

```
  <?xml version="1.0" encoding="UTF-8"?>
- <task>
  - <object>
        <name>BUSBAR</name>
        <horizontal>5</horizontal>
        <vertical>5</vertical>
        <height>7</height>
        <color>WHITE</color>
        <weight>2</weight>
    </object>
  - <object>
        <name>UPPER_BODY</name>
        <horizontal>30</horizontal>
        <vertical>20</vertical>
        <height>5</height>
        <color>GRAY</color>
        <weight>10</weight>
    </object>
  - <subtask>
        <name> Busbar-Approach </name>
        <sequence_number>1</sequence_number>
        <execution_time>75</execution_time>
        <grasp_type>NONE</grasp_type>
        <arm_id>NONE</arm_id>
        <grasping_object>NONE</grasping_object>
    </subtask>
  - <subtask>
        <name> Busbar-Pickup </name>
        <sequence_number>2</sequence_number>
        <execution_time>42</execution_time>
        <grasp_type>Disk-Grasp</grasp_type>
        <arm_id>RIGHT</arm_id>
        <grasping_object>Busbar</grasping_object>
    </subtask>
  - <subtask>
        <name> Busbar-Release </name>
        <sequence_number>3</sequence_number>
        <execution_time>23</execution_time>
        <grasp_type>NONE</grasp_type>
        <arm_id>RIGHT</arm_id>
        <grasping_object>NONE</grasping_object>
    </subtask>
  </task>
```

Fig. 8. ***Generation of the human assembly script by recognizing the assembly intent and action sequences and the objects involved from demonstrations.***

C. Human Assembly Script

Once the system recognizes the intent and action sequence of human assembly and the objects involved (from the demonstration), the results are formalized into the human assembly script, as illustrated in Fig. 8.

IV. ROBOTIC EMBODIMENT OF LEARNED ASSEMBLY TASKS: ROBOT ASSEMBLY SCRIPT

The second part of the proposed automated assembly involves the transformation of the human assembly script into a robot assembly script that is executable by robots in real assembly environments. This so-called robotic embodiment process aims to generate robot-executable action plans based on a set of robot actions predefined in the PDDL domain, so as to accomplish the assembly tasks defined in the human assembly script. Robot action planning also relies on 3-D modeling of the assembly workspace. When planning robot-executable actions, simulation-based verification and retargeting of initially planned robot actions play a key role. Once the planned robot actions are verified as not executable, robot action planning should retarget assembly states to generate compensatory robot actions to make the assembly task robot-executable.

A. 3-D Modeling of the Assembly Workspace

The robotic embodiment of learned assembly tasks requires modeling of the 3-D assembly workspace. Modeling enables the robots to simulate or execute the given assembly task based on the 3-D geometric information of the objects and tools involved in the assembly process. We propose the 3-D workspace modeling system illustrated in Fig. 9. The proposed system detects and recognizes objects of interest, estimates their 3-D poses, and overlays the detected objects with the applicable CAD models based on the 2-D images and 3-D point clouds of the workspace captured by a 3-D camera. As shown in Fig. 9, the system first detects the objects of interest and the featured parts of those objects based on a cascaded object detector formed by YOLO Ver. 3 [7], a serial connection of YOLO 1 and YOLO 2. In addition, we devise an object classification net, Part Net, which takes the object labels from the object detector and the featured parts of the detected objects as input to finalize object labeling. Part Net aims to correct labeling errors made by YOLO 1 to meet the high recognition rate required by the industry. We adopt an engineering approach to 3-D pose estimation of objects for the sake of accuracy. That is, we extract and localize the geometric features of objects in 3-D based on the captured 2-D image and 3-D point cloud. The extracted and localized geometric features are then matched with those of the corresponding CAD model for registration. Then, the iterative closest point (ICP) procedure is applied between the 3-D point cloud and the registered CAD model to refine the object's 3-D pose with high accuracy. The input and output parameters and the performance associated with individual modules are illustrated in Fig. 10.

Note that the objective of detecting featured object parts is twofold: 1) to provide high accuracy in object recognition by correcting errors in initial object labeling by the object detector and 2) to provide efficiency and

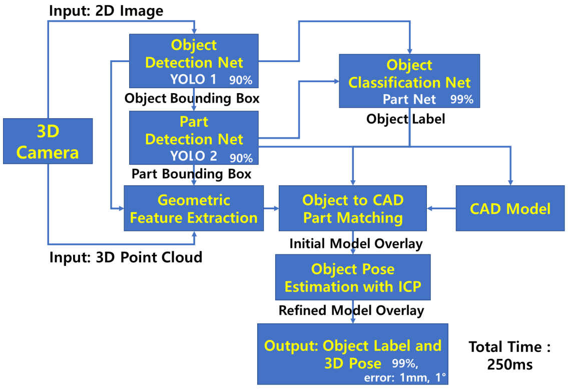


Fig. 9. ***Modeling of a 3-D assembly environment based on a hybrid DL network.***

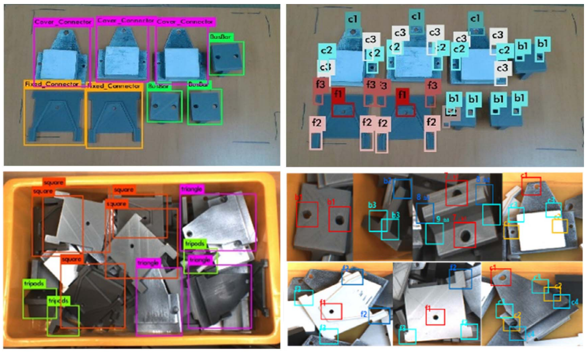


Fig. 10. *Objects (left) and part features (right) detected, respectively, by the object detection net and the part detection net, forming the cascaded object/part detector, for assembly objects on the workspace (top) and in a bin (bottom).*

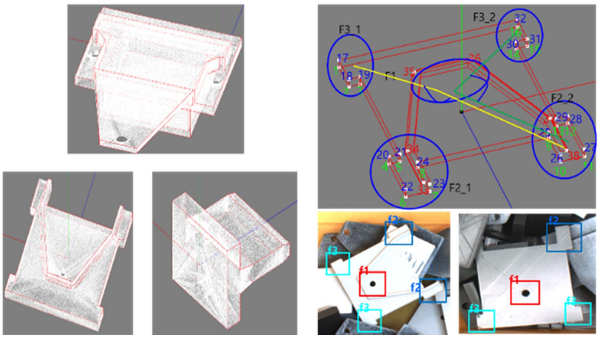


Fig. 12. *Matching the 3-D geometric features extracted from the detected part features with those of the predefined CAD models for 3-D pose estimation of individual objects.*

accuracy in the extraction of the geometric features of objects that are essential for 3-D pose estimation. For instance, geometric features, such as points, line segments, and circles, tend to be associated with the featured parts of objects, as shown on the right-hand side of Fig. 10. As such, the geometric feature extractor in Fig. 10 can extract the geometric features of individual objects simply by applying well-established engineering methods to extract point or line-segment features to the bounding boxes representing the detected parts [38], [39]. Then, these detected features are transformed into 3-D geometric features by incorporating the 3-D point cloud captured by the 3-D camera, as illustrated in Fig. 11.

The extracted 3-D geometric features will be used to estimate the 3-D pose of individual objects. Fig. 12 illustrates the process of matching between the extracted 3-D geometric features and the ground-truth features predefined in the object CAD models for the initial 3-D pose estimation. Note that, since industrial objects are often configured with the same geometric features for their subparts, for instance, f2_1 and f2_2 and f3_1 and f3_2 in Fig. 12, matching between the extracted 3-D geometric features and the ground-truth features requires to use geometric contexts among features, such as distances and angles among features, as additional matching clues. If the object-to-CAD feature matching, matching the detected part features with those of the predefined CAD models, is insufficiently precise for robotic assembly, ICP is applied to the two sets of the point cloud, one from the 3-D camera and the other from the CAD model with the pose estimated by feature matching, to more precisely refine the 3-D poses of individual objects.

We implement the proposed 3-D workspace modeling system for experimental verification. For this purpose, we collect real industrial objects: 20 categories and 100 objects are used as the training and testing data sets. In general, we obtain an object classification success rate of over 99% with less than 1 mm and 1° of pose error. To illustrate the effectiveness of the proposed 3-D workspace modeling process, Fig. 13 shows the final 3-D poses of individual objects estimated after the object-CAD model feature matching and the fine-tuning based on ICP.

B. Robot Action Planning With Retargeting

For robot action planning, we employ PDDL to generate a sequence of robot actions [29] that can accomplish the assembly tasks defined in the human assembly script. PDDL is widely used because it can generate primitive action sequences under a task level of abstraction.

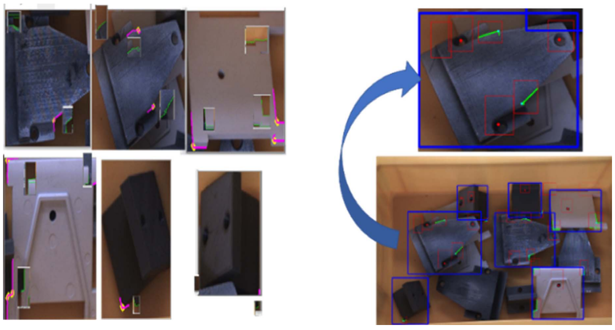


Fig. 11. *Point and line feature extraction based on the part features detected by the cascaded object/part detector: 2-D features (left) and corresponding 3-D features (right) in the 3-D point cloud.*

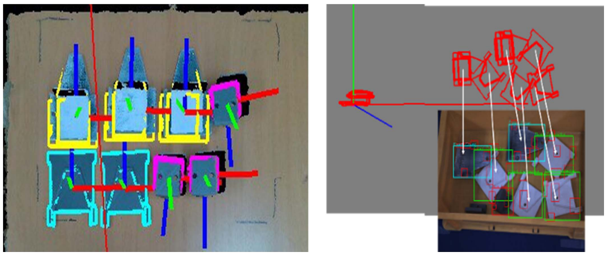


Fig. 13. *Estimated 3-D pose of objects after object-CAD model feature matching and fine-tuning based on ICP. Overlay of the CAD model over the 3-D point cloud captured by the 3-D camera (left and right).*

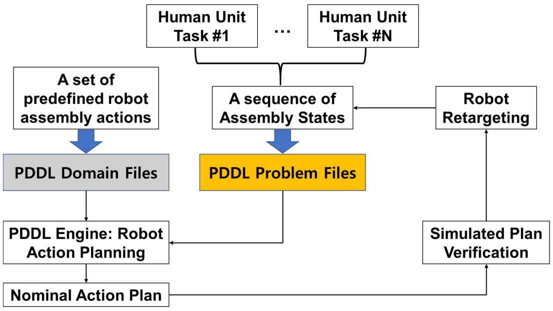

Human Unit Task #1
...
Human Unit Task #N
A set of predefined robot assembly actions
A sequence of Assembly States
Robot Retargeting
PDDL Domain Files
PDDL Problem Files
PDDL Engine: Robot Action Planning
Simulated Plan Verification
Nominal Action Plan

Fig. 14. ***Process of generating a nominal plan based on PDDL.***

The grammatical structure of PDDL consists of two types of files: the domain file and the problem file. The domain file defines actions in terms of their conditions and effects as state transitions. The problem file specifies the initial and terminal states associated with the objects to be handled. In our system, the sequence of assembly states for the robot to accomplish is automatically extracted from the human assembly script and is defined in the PDDL problem files, as illustrated in Fig. 14. The initial and terminal states of each problem file include the initial and terminal poses of the objects and the grippers of robot arms, together with the physical relationships between them. On the other hand, the PDDL domain files are selected from the library of domain files containing a set of predefined robot actions.

As shown in Fig. 14, the nominal action plans generated by the PDDL engine are subject to simulation-based verification to assess their executability by robots. For example, suppose that the nominal action plan for a robot is plugging a USB memory stick into a set-top box, where the nominal plan includes “approach USB memory stick,” “pickup USB memory stick,” and “insert USB memory stick into set-top box” by directly inheriting the sequence of states from the human assembly script. However, the simulated plan verification process reveals that the nominal plan is not robot-executable because the slot in the set-top box is already connected to another device. In this case, the robot has to remove the device from the slot before inserting the USB memory stick into the slot, which we refer to here as robot retargeting. In general, robot retargeting aims to find a solution that addresses the error caused by unexpected situational variation and by the difference in physical and perceptual capabilities between humans and robots, such as the robot kinematic constraints associated with joint limits and degrees of freedom. Robot retargeting is done by adding additional states for the robot to accomplish. For instance, “removing another device from the USB slot of the set-top box” is the complementary robot task that is assigned as the result of robot retargeting.

Robot retargeting also involves the control of waypoints in the assembly workspace. Waypoint control is necessary at times when the initial waypoints fail to produce the required degree of precision and avoid collisions during assembly, for example, consider assembling a bus bar into a power breaker where several positions and postures are defined as the initial waypoints. The initial waypoints through which the end-effector of the robot must pass become subject to retargeting, for instance, when the work cell configuration is altered, e.g., there is a change in the initial position of the bus bar. As another example, due to the limitations of the sensor systems, the 3-D poses of part features, such as holes, that are important for assembly operation may not be recognized as accurately as required. In this case, we need to incorporate the accurate part geometric model from CAD to precisely define the 3-D poses of part features and automatically modify the waypoints for retargeting. Algorithm 1 shows how to assess waypoint reachability and generate trajectories to reach the waypoints. If unreachable, the system analyzes CAD files of parts in the workspace to automatically modify the waypoints with the help of 3-D workspace modeling.

Algorithm 1 Waypoint Control Algorithm

```
1: Input: The current joint configuration, q, the
    end-effector target(waypoint) pose, Td,
    and joint limits, $(q_{min}, q_{max})$.
2: Check if the target pose is reachable
    iterNum = 1
    do
        $\mathrm{T} = f(q)$
        $\mathrm{X} = T^{-1}\bar{T}_{\mathrm{d}}$
        $\mathrm{V} = \log(X)$
        $\Delta\mathrm{q} = J^{+}V : J^{+}$Jacobian Psuedo-Inverse
        $q = q + \Delta\mathrm{q}$
        iterNum = iterNum + 1
while($\|\Delta q\| >$ min_distance and iterNum < threadhold)
3: Generate trajectory to reach the target pose under
    the dynamic joint constraint using quadratic
    programming
    $min_{\Delta q}\lambda = \|q\|^2$ : $\lambda$ regularization term
    $s.t.\ J\Delta q = \varepsilon V$  : $\varepsilon$ step size
    $A\Delta\mathrm{q} \le b$          : $A$ and $b$ are constant
                                determined by joint limits
4: Output: Target pose reachability: reachable if the
    iteration (step 2) converges, not reachable if not.
    Joint trajectory to target pose Generate trajectory
    to reach the target pose when reachable
```

Fig. 15 illustrates a simulated retargeting example when the initial position of a bus bar is changed in the work cell. We found that 98.16% of the 10.717 experiments with different initial positions produce a successful solution.

C. Robot Assembly Script

The application of robot action planning to the assembly tasks defined in a human assembly script, with the support of plan retargeting and 3-D workspace modeling, leads to the creation of a robot assembly script as a formal

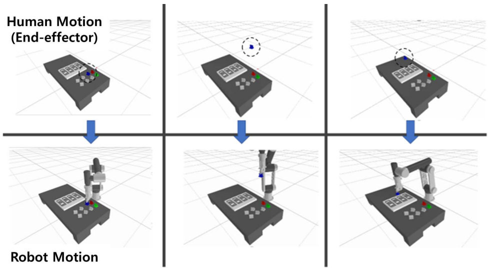


Fig. 15. ***Example of robot retargeting for assembly of a bus bar of a power breaker.***

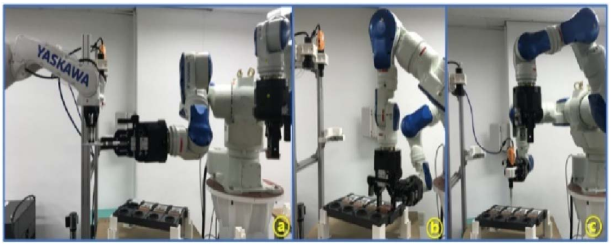


Fig. 17. ***Illustration of the order of power breaker assembly by a cooperative robot system.***

representation of a robot-executable task, as illustrated in Fig. 16.

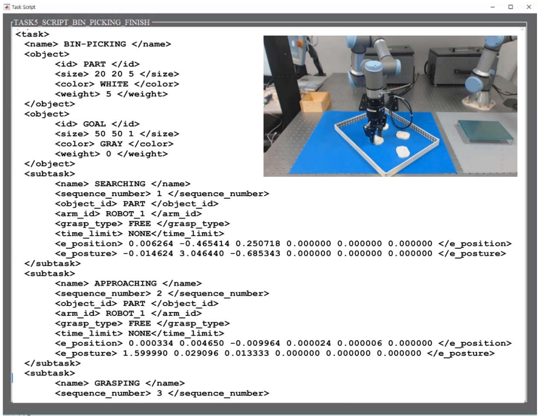
D. Case Study: Power Breaker Assembly

In this section, we present a case study using a power breaker assembly in a manufacturing setting to show how a sequence of unit robot assembly tasks specified in the robot assembly script is executed in adaptation to a real-world assembly environment. The adaptation to a real-world assembly environment is aided by vision-based real-time recognition of the 3-D poses of the objects and tools to be manipulated during assembly. As described in Section IV-A, the real-time 3-D pose estimation of objects and tools involves segmentation of objects and tools from a workspace image, feature extraction and matching with their CAD counterparts, and 3-D pose refinement by aligning their CAD representations with their 3-D point cloud representations obtained from a 3-D camera.

For the assembly of a power breaker or an air circuit breaker, we set up a collaborative robot system consisting of a single-arm robot with a suction gripper and a dual-arm robot with sliding grippers, as illustrated in Fig. 17. Power breaker assembly requires the assembly of two frames and five parts: the upper and lower frames and the fixed contact, fixed contact cover, moving contact, CT case, and busbar. The single-arm robot picks up unaligned fixed contact cover parts and transfers them to the dual-arm robot. The dual-arm robot performs the assembly of the received parts and self-picked parts along with the lower frame.

The sequence of robot unit tasks for assembly of the power breaker is automatically generated as the robot observes a human assembly process consisting of 11 unit human assembly actions. Each human assembly action is then converted to the corresponding robot unit task or unit robot operation that matches the human assembly action. The sequence of unit robot operations, thus generated, as illustrated in Fig. 18, has the same order as that of the unit human assembly actions. Note that some unit robot operations, such as delivering parts between two robots, are added to the sequence of unit robot operations generated directly from that of unit human assembly actions.

The 3-D pose of a part, for instance, a fixed breaker cover that is randomly stacked in a bin, is recognized based on a process consisting of part detection and feature extraction from the cascaded object detector, the 3-D point cloud representation of the part from the 3-D bin image captured by the 3-D camera, matching between the CAD features of the part and the extracted part features, and 3-D point cloud registration between the CAD model and the captured point cloud of the part, as described in detail



```
<task>
  <name> BIN-PICKING </name>
  <object>
        <id> PART </id>
        <size> 20 20 5 </size>
        <color> WHITE </color>
        <weight> 5 </weight>
  </object>
  <object>
        <id> GOAL </id>
        <size> 50 50 1 </size>
        <color> GRAY </color>
        <weight> 0 </weight>
  </object>
  <subtask>
        <name> SEARCHING </name>
        <sequence_number> 1 </sequence_number>
        <object_id> PART </object_id>
        <arm_id> ROBOT_1 </arm_id>
        <grasp_type> FREE </grasp_type>
        <time_limit> NONE</time_limit>
        <e_position> 0.006264 -0.465414 0.250718 0.000000 0.000000 0.000000 </e_position>
        <e_posture> -0.014624 3.046440 -0.685343 0.000000 0.000000 0.000000 </e_posture>
  </subtask>
  <subtask>
        <name> APPROACHING </name>
        <sequence_number> 2 </sequence_number>
        <object_id> PART </object_id>
        <arm_id> ROBOT_1 </arm_id>
        <grasp_type> FREE </grasp_type>
        <time_limit> NONE</time_limit>
        <e_position> 0.000334 0.004650 -0.009964 0.000024 0.000006 0.000000 </e_position>
        <e_posture> 1.599990 0.029096 0.013333 0.000000 0.000000 0.000000 </e_posture>
  </subtask>
  <subtask>
        <name> GRASPING </name>
        <sequence_number> 3 </sequence_number>
```

Fig. 16. ***Robot assembly script generated for the bin-picking task shown in the photograph.***

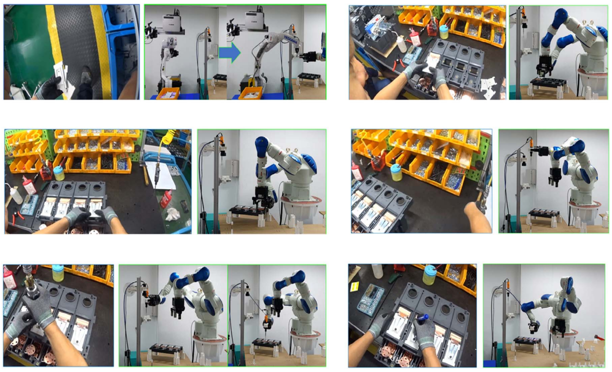


Fig. 18. ***Automatically generated robot unit tasks based on the human assembly script for power breaker assembly, including picking parts, assembling parts, grasping tools, bolting, and inspection.***

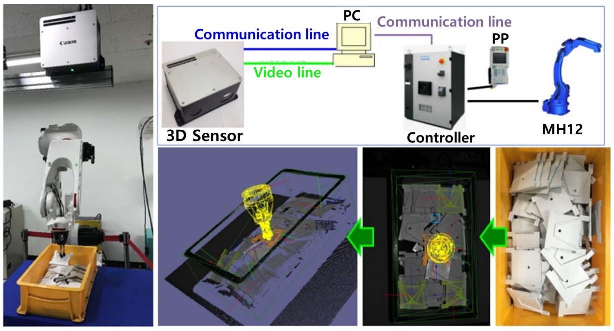


Fig. 19. ***Recognizing unaligned parts and selecting the target part to pick up.***

in Section V-A. The order in which the randomly placed parts are picked up follows the order of their height. Refer to Fig. 19 for an illustration of this process.

The path the robot takes when picking up and delivering a part, while avoiding collisions with objects in the environment, is generated based on the geometric shape of the part, the kinematics of the robot, and the 3-D model of the environment, especially for bolting/inspection operations, as shown in Fig. 20. For instance, to pickup the fixed contact cover with a vacuum gripper, the part reference vector, the suction reference point, the normal vector of the suction plane, and the orientation angle of the part must be designated. Note that, when the direction of the normal vector is incorrect, the suction plate may not be in close contact with the surface of the part, and thus, the pickup or delivery operation may fail.

V. TASK EXECUTION WITH LEARNED SKILLS

When assembly environments can be precisely modeled by sensors such that the target state involved in a robot unit task is well-defined, sensor-guided robot actions may be sufficient for completing the unit task, as described in Section IV. However, when robot unit tasks defined in the robot assembly script are under uncertainties and variations that are too difficult to model and control, the robot needs to resort to skills necessary to overcome such hurdles. To this end, we predefine a set of robot assembly skills that are pretrained for the robot to exercise for the unit tasks that require skills. Refer to Table 2 for an exemplary list of robot skills associated with robot unit tasks.

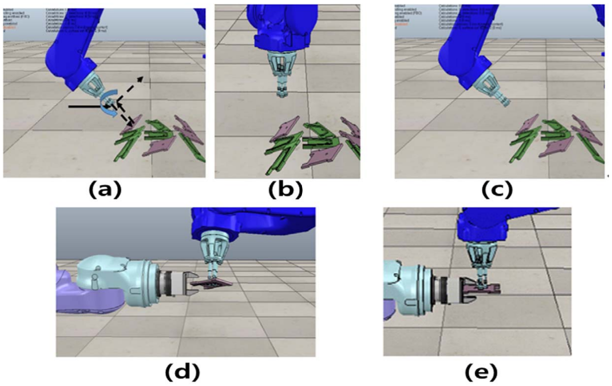


Fig. 20. ***Obtaining stable paths for pickup and delivery of unaligned parts in terms of the part reference vector. (a) Part reference vector, (b) approaching motion before correction, (c) approaching motion after correction, (d) delivering motion before correction, and (e) delivering motion after correction.***

Table 2 Robot Unit Task and Applied Robot Skills

Robot Unit Task	Target Condition	Applied Robot Skill
Grasping-set-top-box (GS)	large change of position	DCNN in Eq. (8)
Inserting-set-top-box (IS)	small change	DMP
Grasping-HDMI-cable-connector (GH)	large change of position	DCNN in Eq. (8)
Regrasping-HDMI-cable-connector (RH)	small change	DMP
Inserting-HDMI-cable-connector (IH)	large change of position and force/torque	DCNN in Eq. (9)
Grasping-power-connector (GP)	large change of position	DCNN in Eq. (8)
Regrasping-power-cable-connector (RP)	small change	DMP
Inserting-power-cable-connector (IP)	large change of position and force/torque	DCNN in Eq. (9)

We propose that a set of robot skills is pretrained by integrating RL with DL and IL. More specifically, we represent each learned robot skill by a DCNN to be trained by a supervisor capable of self-improvement with RL and IL. The skill-embedding DCNNs provide both skill classification and motion generation, so it is possible to use DCNNs to select an appropriate skill from multiple choices. The learned skills can be improved upon (motion paths can be optimized and execution time can be reduced) based on policy learning by weighting exploration with the returns (PoWER). Note that robot skill can be modeled either by DMP for a task with milder uncertainties and variations, e.g., an insertion task, or by DCNN for a task with higher uncertainties and variations, e.g., a grasping task.

A. Learning Skills With DCNN

Unlike DL approaches that require a large amount of training data [17]–[20] to reach high-performance levels, IL approaches rely on a small amount of data from human demonstrations [21]–[23] but come at the potential cost of loss of performance. Here, we integrate IL, DL, and RL in such a way as to learn and improve upon skills by compensating for their shortcomings and maximizing their strengths. To this end, we introduce a supervisor that generates a sufficient amount of training data for the skill-embedding DCNN [25]. The supervisor first learns from human demonstrations based on an IL process and then

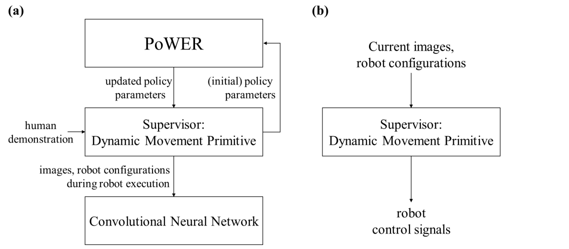

(a)
PoWER
updated policy parameters
(initial) policy parameters
human demonstration
Supervisor: Dynamic Movement Primitive
images, robot configurations during robot execution
Convolutional Neural Network
(b)
Current images, robot configurations
Supervisor: Dynamic Movement Primitive
robot control signals

Fig. 21. ***Entire learning and execution processes: (a) learning process, involving a mixture of IL, RL, and DL and (b) execution process that is done after the learning process.***

carries out self-improvement or self-optimization through an RL process. DCNNs are then trained on the skill using the training data generated by the supervisor.

Fig. 21(a) illustrates the learning process, which combines the IL, RL, and DL processes, as described above. Here, the supervisor uses the dynamic movement primitive (DMP) [36], [37] to learn from human demonstrations and augment skill-related data. DMP is defined as

$$\tau\dot{\mathbf{V}} = K\left(\mathbf{X}^g - \mathbf{X}\right) - D\mathbf{V} + \left(\mathbf{X}^g - \mathbf{X}^0\right)\boldsymbol{\xi} \tag{1}$$

and

$$\tau\dot{\mathbf{X}} = \mathbf{V} \tag{2}$$

where $\mathbf{X}$, $\mathbf{V}$, $\mathbf{X}^0$, and $\mathbf{X}^g$ represent the position, velocity, initial position, and target position vectors, respectively. Similarly, to a linear-damper system, DMP ensures convergence to the final goal or target [24] depending upon the external force term, $\boldsymbol{\xi}$. Note that τ, K, and D indicate the constants that are used to adjust the time scale, spring, and damping terms, respectively. The external force term, which is learned from the human demonstration data set, is defined as

$$\boldsymbol{\xi}(s) = \frac{\sum_{i=1}^{L}\omega_i\psi_i(s)s}{\sum_{i=1}^{L}\psi_i(s)} \tag{3}$$

where $\psi_i(s) = \exp(-h_i(s-c_i)^2)$ is a Gaussian basis function with c_i and h_i, respectively, representing the center and the variance. The parameters L and ω_i indicate, respectively, the number of Gaussian basis functions and their weighting values. The parameter L represents the number of Gaussian basis functions. The term $\boldsymbol{\xi}$ is directly dependent on the phase variable s, which monotonically decreases from 1 to 0, independent of time, and is obtained by the following canonical system:

$$\tau\dot{s} = -\alpha s \tag{4}$$

where α is a predefined constant. A DMP is learned from the average path of several demonstrations. First, the average path $\mathbf{X}(t)$ is recorded, and its derivatives, $\mathbf{V}(t)$ and $\dot{\mathbf{V}}(t)$, are computed for each time step $t = 0, \ldots, T$. Then, the canonical system, $s(t)$, is computed for an appropriately adjusted temporal scaling parameter, τ, which is predefined. Based on (1), $\boldsymbol{\xi}_{\text{target}}(s)$ is computed according to

$$\boldsymbol{\xi}_{\text{target}}(s) = \frac{-K\left(\mathbf{X}^g - \mathbf{X}\right) + D\mathbf{V} + \tau\dot{\mathbf{V}}}{\mathbf{X}^g - \mathbf{X}^0} \tag{5}$$

where $\mathbf{X}^0$ and $\mathbf{X}^g$ are set to $\mathbf{X}(\mathbf{0})$ and $\mathbf{X}(T)$, respectively. With $\boldsymbol{\xi}_{\text{target}}(s)$, $\boldsymbol{\xi}(s)$ can be estimated for motion generation by regressing ω_i in (3) based on the training data $\{\mathbf{X}_n^m\}$, with $m = 1.2, \ldots, M$ and $n = 1.2, \ldots, N$, in such a way as to minimize the error criterion $\boldsymbol{J} = \sum_s(\boldsymbol{\xi}_{\text{target}}(s) - \boldsymbol{\xi}(s))^2$.

Now, we define a skill by the triple, $\boldsymbol{\Theta}$, as follows:

$$\boldsymbol{\Theta} = \left\{\boldsymbol{\Omega} = \{\omega_i\}_{i=1}^{L}, \mathbf{X}^g, T\right\} \tag{6}$$

where $\boldsymbol{\Omega}$ indicates the parameters of the external force term of a DMP. Note that the target, $\mathbf{X}^g$, and the total length of the policy, T, are added to $\boldsymbol{\Theta}$ in order to optimize the skill through an RL process. DMP generates a motion trajectory to satisfy the target $\mathbf{X}^g$ during the length of policy T.

We use a DCNN to implement the skill due to its proven strength in generalization with supervised learning given a sufficient number of training data points [25], [26]. The skill representation by DCNNs deals with the case in which the accurate target pose is not available, so DMP alone cannot represent the skill. Instead, the target pose is available only with uncertainties, which can be handled by the ability of the DCNN to generalize. Furthermore, here, we allow robots to carry out self-supervised learning of DCNNs based on the DMPs explored by RL, starting with the initial DMP from human demonstrations. To this end, the PoWER algorithm is used to improve the policy parameters of DMPs and to perform the self-supervised learning process for DCNNs, as presented in detail in Section V-B As shown in Fig. 21(a), an RL-updated DMP generates robot motion trajectories, during which process a number of training data points, including images, F/T sensor readings, and the joint and end-effector configurations of the robot, are collected. We let the robot collect a sufficient amount of data by repeating this process in various situations. DCNNs are trained by minimizing the loss between the reference motion trajectory generated by the DMP and the output of the DCNN, with the current image and joint configuration data given as inputs. After the learning process is completed, the robots can perform the tasks by generating robot control signals in the appropriate situations based only on DCNNs, as shown in Fig. 21(b).

Fig. 22 illustrates the structure of the DCNN designed to represent a skill, which is similar to the one proposed in [27]. The DCNN consists of three convolutional layers,

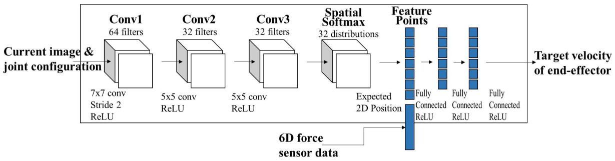


Fig. 22. ***Architecture of DCNN used to learn robot skills associated with unit robot assembly operations.***

a spatial softmax layer, and three fully connected layers. The DCNN takes images (848 × 480 pixels) as input and outputs the 6-D position/orientation of an end-effector. Also, 64, 32, and 32 filters are created in the three convolutional layers, respectively. Unlike a DCNN for object recognition, the pooling process is excluded in the DCNN for motion generation because the accuracy of the target position in the image is important, while ReLU is used as an activation function in every layer. Here, the spatial softmax computes the expected position to convert the pixelwise representations estimated in convolutional layers to spatial coordinate representations, which can be manipulated by the fully connected layers. That is, the spatial softmax helps to estimate 3-D positions or motor torques that the robot can perform [26]. In the fully connected layers, the feature vectors have 64, 32, and six dimensions.

B. Improving Skills Through RL

We apply RL to the aforementioned skills trained on demonstrations to improve performance. As shown in Algorithm 2, the skill parameters of DMPs are optimized into $\mathbf{\Theta}^*$ based on the RL process, iPoWER, as a means of improving the DCNN supervisor. The iPoWER algorithm shown in Algorithm 2 represents a slightly modified version of the original PoWER algorithm [28] with reduced execution time.

The iPoWER algorithm is based on a deterministic policy $\bar{\mathbf{a}} = \mathbf{\Omega}^T\mathbf{\Psi}(\mathbf{X}, t)$ with the weighting parameters $\mathbf{\Omega}$ and the basis functions $\mathbf{\Psi}$ of a DMP [24]. However, when optimizing a DMP, this policy is turned into a stochastic policy using additive exploration $\varepsilon(\mathbf{X}, t)$ for model-free RL. That is, the policy, $(\mathbf{a}_t|\mathbf{X}_t, t)$, is represented in the following form: $\mathbf{a} = \mathbf{\Omega}^T\mathbf{\Psi}(\mathbf{X}, t) + \varepsilon(\mathbf{\Psi}(\mathbf{X}, t))$ with $\lfloor \varepsilon_t \rfloor_{ij} \sim N(0, \sigma_{ij}^2)$, where σ_{ij} is a metaparameter of the exploration. Note that σ_{ij} is also subject to optimization in this algorithm. In the iPoWER algorithm, the length of a corresponding DMP can be reduced by the stop signal t_{stop} when $\mathbf{X}^g = \mathbf{X}_t$ and $t < T_k$. This means that robots arrive quickly at the target compared to human demonstrations during the RL process. The stop signal for t_{stop} is generated when the robot reaches the target, $|\mathbf{X}^g \pm \epsilon|$, within extremely small margins.

To calculate the expected return values for the improvement process, the reward function should be defined.

Algorithm 2 iPoWER Algorithm for Improving the Parameters of Skills Considering Execution Time Step and Path Optimization

```
Input: a set of initial parameters
  $\mathbf{\Theta} = \{\mathbf{\Theta}_1, \mathbf{\Theta}_2, \ldots, \mathbf{\Theta}_N\}$ of all skills.
Using initial parameters $\mathbf{\Theta}_i = \{\mathbf{\Omega}_i, \mathbf{X}_i^g, T_i\}$
  of a skill belonging to the
CNN with the maximum likelihood,
Set initial parameters $\mathbf{\Theta}_k = \mathbf{\Theta_0} = \{\mathbf{\Omega}_0, \mathbf{X}^g, T_0\}$
  of a motor skill
while true do
    Sampling: Using $\mathbf{\Omega}_k$, $\mathbf{X}^g$, and $T_k$, generate
              rollout $(\mathbf{X})$ from
              $\mathbf{a} = (\mathbf{\Omega}_k + \varepsilon_t)^{\mathrm{T}}\mathbf{\Psi}(\boldsymbol{X}, t)$
              based on Eq. (1) with
              exploration $\lfloor \varepsilon_t \rfloor_{ij} \sim N(0, \sigma_{ij}^2)$
              as a stochastic policy.
              if $\mathbf{X}_t = \mathbf{X}^g$ and $t < T_k$ then
                  Set $\tilde{T} = t$ and collect all
                  information
                  $(t, \mathbf{X}_t, \mathbf{a}_t, \mathbf{X}_{t+1}, \varepsilon_t, r_{t+1})$
                  for $t = \{1, 2, \ldots \tilde{T} + 1\}$.
              else
                  Discard all information
                  $(t, \mathbf{X}_t, \mathbf{a}_t, \mathbf{X}_{t+1}, \varepsilon_t, r_{t+1})$
                  for $t = \{1, 2, \ldots \tilde{T} + 1\}$.
              end if
    Estimating: Use unbiased estimate of the
              value function
              $\hat{Q}^\pi(\mathbf{X}, \mathbf{a}, t) = \sum_{\tilde{t}}^{\tilde{T}} r(\mathbf{X}_{\tilde{t}}, \mathbf{a}_{\tilde{t}}, \mathbf{X}_{\tilde{t}+1}, \tilde{t})$;
              difference between the target
              position from human
              demonstration and the current
              robot position.
    Reweighting: Reweight rollouts and discard
              low-reward rollouts.
              Update the parameters of DMP
              using $\mathbf{\Omega}_{k+1} = \mathbf{\Omega}_k +$
              $\langle \sum_{t=1}^{\tilde{T}} \varepsilon_t Q^\pi(\mathbf{X}, \mathbf{a}, t) \rangle$
              $/ \langle \sum_{t=1}^{\tilde{T}} Q^\pi(\mathbf{X}, \mathbf{a}, t) \rangle$, where $\langle \cdot \rangle$
              means an expectation by the
              importance sampler.
              Update the parameters $\mathbf{\lambda}_{k+1}$ of
              the HMM using reaction
              force/moment
              recorded during the motion
              generation
              $\mathbf{a} = (\mathbf{\Omega}_{k+1} + \varepsilon_t)^{\mathrm{T}}\mathbf{\Psi}(\mathbf{X}, t)$.
              Update $T_{k+1} = \tilde{T}$.
    if $\mathbf{\Theta}_{k+1} \approx \mathbf{\Theta}_k$ then
        break
    end if
end while
Output: the optimized parameters
  $\mathbf{\Theta}^* = \{\mathbf{\Omega}_{k+1}, \mathbf{X}^g, T_{k+1}\}$.
```

Its generation equation is defined as

$$r(t) = \exp\left(-\alpha\left(\mathbf{X}^{g} - \mathbf{X}(t)\right) - \beta\left(\frac{1}{\mathbf{Y}^{s} - \mathbf{Y}(t)}\right)\right) \qquad (7)$$

where $\mathbf{X}$ and $\mathbf{Y}$ indicate the robot state values, such as camera images, F/T sensor data, and tool orientations, which can be measured. The superscripts g and s denote the target and starting values of each variable depending on the given task. Here, the term $(\mathbf{X}^{g} - \mathbf{X}(t))$ is used to get a high return value when the robot configuration is close to the target values, and the term $(1/(\mathbf{Y}^{s} - \mathbf{Y}(t)))$ is used to get a high return value when it is far from the starting value. The parameters α and β are constants that are used to adjust the degree of reflection of each term. Equation (7) is designed to take the form of $\exp^{-(x)}$; therefore, a lower value of each term provides a higher return value.

C. Case Study: Set-Top-Box Assembly Using Skills

To show how the execution of pretrained skill-based primitive tasks helps cope with uncertainties and variations in a real assembly environment, we present a case study using a set-top-box assembly currently in practice by a local manufacturing company.

Here, the pretrained robot skills associated with unit assembly operations play a key role. We pretrain the robot to be able to carry out the following eight-unit assembly operations with skills: 1) "grasping-set-top-box (GS)" using the skill of grasping the set-top box after estimating its position and posture; 2) "inserting-set-top-box (IS)" using the skill of inserting the set-top box into a fixed jig; 3) "grasping-HDMI-cable-connector (GH)" using the skill of grasping the HDMI-cable-connector after estimating its position and posture; 4) "regrasping-HDMI-cable-connector (RH)" using the skill of regrasping the HDMI-cable-connector to measure the F/T values during the insertion motion; 5) "inserting-HDMI-cable-connector (IH)" using the skill of inserting the HDMI-cable-connector into the hole of set-top-box; 6) "grasping-power-cable-connect (GP)" using the skill of grasping the power-cable-connector after estimating its position and posture; 7) "regrasping-power-cable-connector (RP)" using the skill of regrasping the power-cable-connector to measure the F/T values during the insertion maneuver; and 8) "inserting-power-cable-connector (IP)" using the skill of inserting the power-cable-connector into the set-top box. Fig. 23 illustrates the eight robot skills introduced above, while Table 2 shows how each skill is implemented, either by DCNN or by DMP. A skill is implanted by DCNN when the degree of uncertainty associated with the goal pose or the target condition is high and by DMP when it is lower.

Note that skills 1), 3), 5), 6), and 8) are represented by DCNNs trained through the supervised learning process described in Section V. In contrast, skills 2), 4), and 7) are represented by DMPs. This is because 2), 4), and 7) are used in situations that involve little change in the position and posture of the objects they deal with and can be performed in various environments. For 1), 3), and 6), it is necessary for the robot to consider the relative position and orientation between the robot and the target object. In contrast, for 5) and 5), it is important for the robot to generate the appropriate motion trajectories that account for the relative force and torque between the female and the male objects. Therefore, we use CNNs for 1), 3), 5), 6), and 8) to connect the motion generation to the perception of object poses and interaction forces.

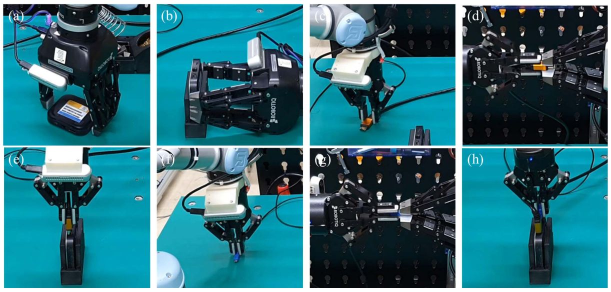


Fig. 23. *Illustrations of eight skills in the set-top-box assembly task: (a) GS, (b) IS, (c) GH, (d) RH, (e) IH, (f) GP, (g) RP, and (h) IP.*

We set up the experimental testbed for the set-top-box assembly with the support of the pretrained robot skills for the unit robot assembly operations, as illustrated in Fig. 24. The testbed is equipped with UR3 and UR5 robotic arms from Universal Robots, Denmark, two FT300 F/T sensors, two-and three-finger grippers from Robotiq, Canada, and two cameras from Intel, USA. Note that the positions and orientations of the set-top box, HDMI cable connector and power cable connector are designed to change randomly for the experiment. The robot can localize the objects (connectors and holes) using the camera on its wrist. The clearance between the jig and the set-top box is approximately 300 μm, and the clearance between the cable connectors and holes of the set-top box is approximately 10 μm.

To calculate the expected return values for the robot skills, the following reward function r^{g} is assigned

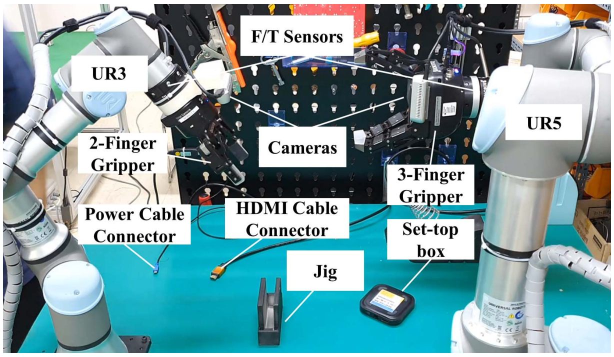


Fig. 24. *Experimental setup of the set-top-box assembly task.*

to 1), 3), and 6):

$$r^g = \exp\left(-\mathbf{I_d}(t)\right) \tag{8}$$

where $\mathbf{I_d}(t)$ indicates the dissimilarity between the target image ($\mathbf{I}_g$) and the current image ($\mathbf{I}(t)$), e.g., the difference between the images of the target and current states. The reward value increases as the current image becomes more similar to the target image. On the other hand, the reward function, r^i, for skills 5) and 8) is defined as

$$r^i = \exp\left(-\alpha \mathbf{F}_{x,y,z}(t) - \beta \mathbf{M}_{x,y,z}(t) - \gamma \mathbf{P}_z(t) - \delta \mathbf{I}(t)\right) \tag{9}$$

where **F**, **M**, and **P** indicate, respectively, the force, moment, and distance components of the robot. In particular, **P** represents the deviation from the reference axis of the tool coordinate system. **F**, **M**, and **P** are calculated using absolute error equations in the following form, as illustrated for **F** only:

$$\mathbf{F}_{x,y,z}(t) = \left|F_x^g - F_x(t)\right| + \left|F_y^g - F_y(t)\right| + \left|F_z^g - F_z(t)\right|$$

where the superscript, g, indicates the target value. The last term, $\mathbf{I}(t)$, represents the dissimilarity between the target and the current images, as in (8), while α, β, γ, and δ are constants that are used to adjust the contribution of individual **F**, **M**, **P**, and **I** values to the reward function, r^i.

First, DMP and the target image, $\mathbf{I}^g$, are extracted from a human demonstration as the supervisory exemplar and for computation of the return value, respectively, in RL. Since the target image is obtained from the camera attached to the robot's wrist, which defines the relative position and orientation between the robot and the target object, the absolute pose of the robot is irrelevant to the skill description. For the "inserting" skill, the human demonstration defines the target force, $\mathbf{F}^g_{x,y,z}$, torque, $\mathbf{M}^g_{x,y,z}$, and insertion depth P^g_z. The motion trajectories of the robot were extracted at 50 Hz using the kinesthetic teaching method; then, training data were acquired through self-reproduction.

To perform the "set-top-box-assembly" task, eight DMPs are modeled as the supervisors that train the DCNNs using the training data. Individual DMPs improve by themselves, as do DCNNs, through the processes of self-exploration, self-reproduction, and self-improvement based on Algorithm 2 and the reward functions shown in (8) and (9). Three "grasp" and three "insert" type skills undergo this process of self-improvement of supervisors and DCNNs through 5.000–10.000 repetitions and 200–300 repetitions, respectively. We observe that the return values increase, while the execution time is reduced, with an increasing number of iterations. Fig. 25 shows the performance of iPoWER in comparison with the original PoWER algorithms. The iPoWER algorithm is better at generating optimal paths and reducing the number of execution time steps, as shown in Fig. 25(a) and (b).

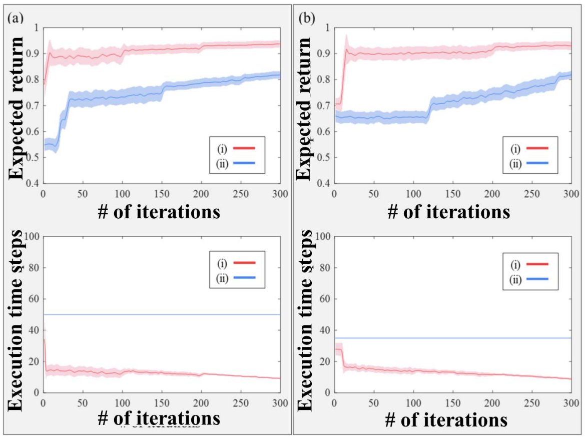


Fig. 25. ***Performance of the original PoWER and the iPoWER algorithm in the set-top-box-assembly task for (a) "grasping-set-top-box" and (b) "inserting-hdmi-cable-connector" skills. The upper and lower rows illustrate the return values and the number of execution time steps, respectively. The red lines in (i) indicate the results using the iPoWER algorithm, and the blue lines in (ii) indicate the results using the original PoWER algorithm.***

Having pretrained the robot on several skills, we next carry out experiments to evaluate the performance of the learned skills. The first experiment aims to evaluate the performance of the grasping skill when it is used to pick up three different objects, the "set-top box," "HDMI-cable-connector," and "power-cable-connector," which are randomly placed on a worktable. Specifically, we intend to test the DCNNs trained on the grasping skill to evaluate their ability to generate appropriate control signals to pick up objects using the images of the scene as input. The experiment results in 98 successes out of 100 trials. Note that the two failures occur when the cables are slightly tilted. These failures occur because we exclude the flipping or standing motion necessary to grasp tilted objects from the learning process. The experiment deals with insertion skills and uses the same three objects described in the first experiment. In the experiment, two UR robots are to insert the "set-top box" and the "cables" into the jig and the "set-top box," respectively, with respective clearances of 300 and 10 μm. The UR robots should complete the insertion process based on the input images and the reaction force/torque based on DCNNs. We evaluate whether the DCNNs can generate appropriate force/torque and pose control signals based on the images and the robot configurations given as the input. As a result, we achieved 97 successes out of 100 trials. Note that the three failures occurred when the connectors fell off of the cliff hole outside the "set-top box."

Finally, we evaluate the performance of the task planning and DCNN skill arrangement based on the PDDL engine. We carried out four experimental cases, as shown in Fig. 26. The cases, illustrated, respectively, by Fig. 26 (a), (b), (c) and (d), are as follows.

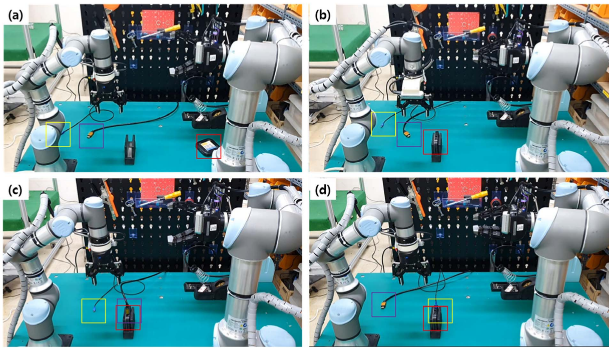


Fig. 26. ***Illustrations of the initial experimental setups used to evaluate the performance of skill and the robot's planning performance. In these illustrations, the red, purple, and yellow boxes indicate the "set-top box," "HDMI-cable-connector," and "power-cable-connector," respectively. (a) Random placement of the three objects. (b) Insertion of "set-top box" into "jig." (c) Insertion of "set-top box" and "HDMI-cable-connector," respectively, into "jig" and "set-top box." (d) Insertion of "set-top box" and "power-cable-connector," respectively, into "jig" and "set-top box."***

(a) The "set-top box," the "HDMI-cable-connector," and the "power-cable-connector' are to be randomly placed on the floor.
(b) The "set-top box" is to be inserted into the "jig," and the "HDMI-cable-connector" and "power-cable-connector" are to be randomly placed on the floor. Note that the "set-top box" is already in its target state.
(c) The "set-top box" and "HDMI-cable-connector" are to be inserted into the "jig" and the "set-top box," respectively. In this case, two target objects are in their target states, with the exception of the "power-cable-connector."
(d) The "set-top box" and "power-cable-connector" are to be inserted into the "jig" and the "set-top box," respectively. In this case, two target objects are in their target states, with the exception of the "HDMI-cable-connector.'

In the case of (a), the URs performed the "set-top-box-assembly" task according to their nominal plan. In the cases of (b)–(d), the URs performed their unit tasks that are not yet in their target states based on the proposed task planning and DCNN arrangement method.

The experiments described in Sections IV and V successfully validated the proposed automated assembly system for its applicability to power breaker assembly and set-top-box assembly in real-world manufacturing settings. Although successfully validated, the current prototype system is by no means without limitations and failures. In general, the success of the proposed system depends on the novelty of assembly environments or the deviation from what is learned and the capability of the system to cope with the novelty based on sensor-based workspace modeling, robotic task replanning, and pretrained robot skills. For instance, we occasionally observed the following failure modes during experiments: 1) robot fails in reaching the target as obstacles hinder the robot from reaching the target position, or the vision system fails in identifying the 3-D pose of the target object and 2) robot fails in inserting a screw into a hole as the robot either fails in grasping or incorrectly grasps the screw, or the sensing and control errors of the robot are excessive to insertion tolerances. Note that, by further extending the capability of replanning and robot skills, we plan to increase the power of recovery from failures due to excessive variations and uncertainties.

VI. CONCLUSION

In this article, we presented an automated robotic assembly system built upon an autoprogramming environment that can reduce the setup time and cost for reconfiguring and reprogramming robots when it is frequently necessary to reassign robot tasks, as in smart manufacturing plants. A three-part approach was implemented: learning by observation, a robotic embodiment with action planning, and simulated retargeting and execution with pretrained skills. The approach was shown to be effective and viable through implementation and experimentation. We demonstrated that the DL-based real-time recognition of human assembly action sequences and grasping types allows the robot to effectively learn the given assembly task from observation. Furthermore, we showed that PDDL-based robot action planning from the learned human assembly, together with simulation-based verification and retargeting into action planning, represents an effective means of robotic embodiment. In particular, for task execution, we verified that pretraining of robotic skills through DL and RL is crucial for the robot to adapt to the uncertainties and variations that are often seen in the assembly process. Such uncertainties and variations would be difficult to handle otherwise. We successfully validated the proposed system by developing a prototype system and applying it to two real-world manufacturing scenarios, power breaker assembly, and set-top-box assembly, using commercially available robots. In addition, we also showed how recent advancements in DL and RL can impact the next generation of automated assembly for the smart manufacturing of the future.

In the future, we plan to continue improving this automated assembly system, especially its ability to deal with unexpected failures that may happen during assembly. Also, we are interested in applying the proposed system to smart workbench-based man–machine collaboration systems in order to determine the method and the order of robot operations in collaboration with human tasks while analyzing human work behaviors and methods, so as to provide guidance to improve productivity and safety. Moreover, the capacity to understand the ways in which humans work may be applicable to the development of an autonomous system that allows proficiency-or skill-based optimal task assignment to workers when planning production for smart manufacturing.

Acknowledgment

The authors greatly appreciate their research colleagues who contributed to the experimentation and documentation for the work described in this article, including Dongjin Kim of the Korea Advanced Institute of Science and Technology (KAIST) and Yeonho Lee and Soojin Lee of Sungkyunkwan University (SKKU).

REFERENCES

[1] *The Hahngroup, Rethink Robotics*. Accessed: Mar. 2021. [Online]. Available: https://www.rethinkrobotics.com/

[2] Syddansk Universitet, European Commission. *Intelligent Observation and Execution of Actions and Manipulations*. Accessed: Mar. 2021. [Online]. Available: https://cordis.europa.eu/project/id/269959

[3] "A reconfigurable robot workcell for fast set-up of automated assembly processes in SMEs," Dept. Autom., Biocybern., Robot. Jožef Stefan Inst., ReconCell. Accessed: Mar. 2021. [Online]. Available: http://www.reconcell.eu/

[4] B. Nemec, K. Yasuda, N. Mullennix, N. Likar, and A. Ude, "Learning by demonstration and adaptation of finishing operations using virtual mechanism approach," in *Proc. IEEE Int. Conf. Robot. Automat. (ICRA)*, May 2018, pp. 7219–7225.

[5] *Latest Official Release: 0.8.2, OpenRave*. Accessed: Mar. 2021. [Online]. Available: http://openrave.org

[6] S. Niekum, S. Osentoski, G. Konidaris, S. Chitta, B. Marthi, and A. G. Barto, "Learning grounded finite-state representations from unstructured demonstrations," *Int. J. Robot. Res.*, vol. 34, no. 2, pp. 131–157, Feb. 2015.

[7] J. Redmon and A. Farhadi, "YOLOv3: An incremental improvement," 2018, *arXiv:1804.02767*. [Online]. Available: http://arxiv.org/abs/1804.02767

[8] S.-E. Wei, V. Ramakrishna, T. Kanade, and Y. Sheikh, "Convolutional pose machines," in *Proc. IEEE Conf. Comput. Vis. Pattern Recognit. (CVPR)*, Jun. 2016, pp. 4724–4732.

[9] K. He, X. Zhang, S. Ren, and J. Sun, "Deep residual learning for image recognition," in *Proc. IEEE Conf. Comput. Vis. Pattern Recognit. (CVPR)*, Jun. 2016, pp. 770–778.

[10] G. Huang, Z. Liu, L. Van Der Maaten, and K. Q. Weinberger, "Densely connected convolutional networks," in *Proc. IEEE Conf. Comput. Vis. Pattern Recognit. (CVPR)*, Jul. 2017, pp. 2261–2269.

[11] Wikipedia, Wikimedia Foundation. *Planning Domain Definition Language*. Accessed: Feb. 2021. [Online]. Available: https://en.wikipedia.org/wiki/Planning_Domain_Definition_Language

[12] T. Feix, I. M. Bullock, and A. M. Dollar, "Analysis of human grasping behavior: Correlating tasks, objects and grasps," *IEEE Trans. Haptics*, vol. 7, no. 4, pp. 430–441, Oct. 2014.

[13] Y. Yang, C. Fermuller, Y. Li, and Y. Aloimonos, "Grasp type revisited: A modern perspective on a classical feature for vision," in *Proc. IEEE Conf. Comput. Vis. Pattern Recognit. (CVPR)*, Jun. 2015, pp. 400–408.

[14] T. Feix, J. Romero, H.-B. Schmiedmayer, A. M. Dollar, and D. Kragic, "The GRASP taxonomy of human grasp types," *IEEE Trans. Human-Machine Syst.*, vol. 46, no. 1, pp. 66–77, Feb. 2016.

[15] T.-Y. Lin, P. Goyal, R. Girshick, K. He, and P. Dollár, "Focal loss for dense object detection," in *Proc. IEEE Conf. Comput. Vis. Pattern Recognit.*, Oct. 2017, pp. 2980–2988.

[16] T.-Y. Lin, P. Dollár, R. Girshick, K. He, B. Hariharan, and S. Belongie, "Feature pyramid networks for object detection," in *Proc. IEEE Conf. Comput. Vis. Pattern Recognit. (CVPR)*, Jul. 2017, pp. 2117–2125.

[17] R. Girshick, "Fast R-CNN," in *Proc. IEEE Conf. Comput. Vis. Pattern Recognit.*, Dec. 2015, pp. 1440–1448.

[18] S. Gu, E. Holly, T. Lillicrap, and S. Levine, "Deep reinforcement learning for robotic manipulation with asynchronous off-policy updates," in *Proc. IEEE Int. Conf. Robot. Automat. (ICRA)*, May 2017, pp. 3389–3396.

[19] Y. Yang, Y. Li, C. Fermuller, and Y. Aloimonos, "Robot manipulation action plans by 'watching' unconstrained videos from the world wide Web," in *Proc. 29th AAAI Conf. Artif. Intell.*, 2015, pp. 1–7.

[20] S. Kumra and C. Kanan, "Robotic grasp detection using deep convolutional neural networks," in *Proc. IEEE/RSJ Int. Conf. Intell. Robots Syst. (IROS)*, Sep. 2017, pp. 769–776.

[21] C. Finn, T. Yu, T. Zhang, P. Abbeel, and S. Levine, "One-shot visual imitation learning via meta-learning," 2017, *arXiv:1709.04905*. [Online]. Available: http://arxiv.org/abs/1709.04905

[22] D. Silver, J. Bagnell, and A. Stentz, "High performance outdoor navigation from overhead data using imitation learning," in *Proc. 4th Robot., Sci. Syst.*, Zürich, Switzerland, vol. 1, 2008.

[23] S. Ross and D. Bagnell, "Efficient reductions for imitation learning," in *Proc. 13th Int. Conf. Artif. Intell. Statist.*, 2010, pp. 661–668.

[24] P. Pastor, H. Hoffmann, T. Asfour, and S. Schaal, "Learning and generalization of motor skills by learning from demonstration," in *Proc. IEEE Int. Conf. Robot. Automat. (ICRA)*, May 2009, pp. 763–768.

[25] K. Makantasis, K. Karantzalos, A. Doulamis, and N. Doulamis, "Deep supervised learning for hyperspectral data classification through convolutional neural networks," in *Proc. IEEE Int. Geosci. Remote Sens. Symp. (IGARSS)*, Jul. 2015, pp. 4959–4962.

[26] C. Mitash, K. E. Bekris, and A. Boularias, "A self-supervised learning system for object detection using physics simulation and multi-view pose estimation," in *Proc. IEEE/RSJ Int. Conf. Intell. Robots Syst. (IROS)*, Sep. 2017, pp. 545–551.

[27] S. Levine, C. Finn, T. Darrell, and P. Abbeel, "End-to-end training of deep visuomotor policies," *J. Mach. Learn. Res.*, vol. 17, no. 39, pp. 1–40, 2016.

[28] J. Kober and J. Peters, "Policy search for motor primitives in robotics," in *Proc. Adv. Neural Inf. Process. Syst.*, 2009, pp. 849–856.

[29] S. H. Lee, I. H. Suh, S. Calinon, and R. Johansson, "Autonomous framework for segmenting robot trajectories of manipulation task," *Auton. Robots*, vol. 38, no. 2, pp. 107–141, 2015.

[30] A. Dömel, S. Kriegel, M. Kaßecker, M. Brucker, T. Bodenmüller, and M. Suppa, "Toward fully autonomous mobile manipulation for industrial environments," *Int. J. Adv. Robot. Syst.*, vol. 14, no. 4, pp. 1–19, 2017.

[31] E. Guizzo, "Boston dynamics enters warehouse robots market, acquires kinema systems," in *Proc. IEEE Spectr.*, Apr. 2019. [Online]. Available: https://spectrum.ieee.org/automaton/robotics/industrial-robots/boston-dynamics-warehouse-robots-acquires-kinema-systems

[32] S. Song and J. Xiao, "Deep sliding shapes for amodal 3D object detection in RGB-D images," in *Proc. IEEE Conf. Comput. Vis. Pattern Recognit. (CVPR)*, Jun. 2016, pp. 808–816.

[33] F. Perazzi, A. Khoreva, R. Benenson, B. Schiele, and A. Sorkine-Hornung, "Learning video object segmentation from static images," in *Proc. IEEE Conf. Comput. Vis. Pattern Recognit. (CVPR)*, Jul. 2017, pp. 3491–3500.

[34] W. Grieskamp, Y. Gurevich, W. Schulte, and M. Veanes, "Generating finite state machines from abstract state machines," in *Proc. Int. Symp. Softw. Test. Anal. (ISSTA)*, 2002, pp. 112–122.

[35] W. Kerr, A. Tran, and P. Cohen, "Activity recognition with finite state machines," in *Proc. 22nd Int. Joint Conf. Artif. Intell.*, 2011, pp. 1–6.

[36] S. Schaal, "Dynamic movement primitives—A framework for motor control in humans and humanoid robotics," in *Adaptive Motion of Animals and Machines*. Tokyo, Japan: Springer, 2006, pp. 261–280.

[37] A. C. Dometios, Y. Zhou, X. S. Papageorgiou, C. S. Tzafestas, and T. Asfour, "Vision-based online adaptation of motion primitives to dynamic surfaces: Application to an interactive robotic wiping task," *IEEE Robot. Autom. Lett.*, vol. 3, no. 3, pp. 1410–1417, Jul. 2018.

[38] J. Kim and S. Lee, "Extracting major lines by recruiting zero-threshold canny edge links along Sobel highlights," *IEEE Signal Process. Lett.*, vol. 22, no. 10, pp. 1689–1692, Oct. 2015.

[39] C. Harris and M. Stephens, "A combined corner and edge detector," in *Proc. Alvey Vis. Conf.*, 1988, pp. 147–152.

ABOUT THE AUTHORS

Sanghoon Ji received the B.S. and M.S. degrees in control and instrumentation engineering and the Ph.D. degree in electrical engineering and computer sciences from Seoul National University, Seoul, South Korea, in 1995, 1997, and 2007, respectively.

From 1997 to 2002, he was a Research Engineer with IAE, Yongin, South Korea. From August 2007 to September 2008, he was a Deputy General Manager with Doosan Infracore Ltd., Yongin. He is currently the Director of the Applied Robot R&D Department, Korea Institute of Industrial Technology, Ansan, South Korea. His research interests include multiagent robot systems and robot S/W platforms.

Sukhan Lee (Life Fellow, IEEE) received the B.S. and M.S. degrees in electrical engineering from Seoul National University, Seoul, South Korea, in 1972 and 1974, respectively, and the Ph.D. degree in electrical engineering from Purdue University, West Lafayette, IN, USA, in 1982.

From 1998 to 2003, he was an Executive Vice President and a Chief Research Officer with the Samsung Advanced Institute of Technology, Giheung, South Korea. Since 2003, he has been a Professor of information and communication engineering and of artificial intelligence and robotics with Sungkyunkwan University, Suwon, South Korea.

Dr. Lee is a Life Fellow of the Korea National Academy of Science and Technology.

Sujeong Yoo received the B.S. and M.S. degrees in control and instrumentation engineering and the Ph.D. degree in electrical engineering and computer sciences from Seoul National University, Seoul, South Korea, in 1993, 1995, and 2013, respectively.

She worked for four years at LG, Seoul, South Korea, as a Research Engineer on projects with machine learning and artificial neural network. She is currently a Principal Researcher with the Applied Robot R&D Department, Korea Institute of Industrial Technology, Ansan, South Korea. Her research interests include computer vision and machine learning.

Ilhong Suh (Fellow, IEEE) received the Ph.D. degree in control engineering from the Department of Electrical Engineering, Korea Advanced Institute of Science and Technology (KAIST), Daejeon, South Korea.

He is currently a Co-Founder and the CEO of CogAplex Company, a startup of AI-robot, Seoul, South Korea. He is also a Professor Emeritus with the Department of Intelligent Robotics Engineering and the Department of Electronics and Computer Engineering, Hanyang University, Seoul. His research interests lie in the areas of machine learning and control for robots including action-coupled perception and learning, skill acquisition, autonomous navigation, and human–robot interaction.

Dr. Suh is a Senior Fellow of Korea National Academy of Engineering.

Inso Kwon (Member, IEEE) received the B.S. and M.S. degrees in mechanical design and production engineering from Seoul National University, Seoul, South Korea, in 1981 and 1983, respectively, and the Ph.D. degree in robotics from the Robotics Institute, Carnegie Mellon University, Pittsburgh, PA, USA, in 1990.

He has been the KEPCO Chair Professor with the Department of Electrical Engineering, Korea Advanced Institute of Science and Technology, Daejeon, South Korea, since 1992.

Dr. Kwon is a member of the Korean Robotics Society.

Frank C. Park (Fellow, IEEE) received the B.S. degree in electrical engineering from the Massachusetts Institute of Technology (MIT), Cambridge, MA, USA, in 1985, and the Ph.D. degree in applied mathematics from Harvard University, Cambridge, in 1991.

He joined the Faculty of the University of California at Irvine, Irvine, CA, USA, in 1991. Since 1995, he has been a Professor of mechanical engineering with Seoul National University, Seoul, South Korea. His research interests include robot mechanics, planning and control, vision and image processing, mathematical data science, and related areas of applied mathematics.

Dr. Park has been an IEEE Robotics and Automation Society Distinguished Lecturer. He is also the incoming President of the IEEE Robotics and Automation Society for the term 2022–2023.

Sanghyoung Lee received the Ph.D. degree in artificial intelligence for robot manipulation from the Department of Electronics and Computer Engineering, Hanyang University, Seoul, South Korea, in 2013.

He is a Researcher with the Innovative Smart Manufacturing R&D Department, Korea Institute of Industrial Technology (KITECH), Cheonan, South Korea. His research interests include artificial intelligence, machine learning and deep learning for robot manipulation, collaborative robotics, and industrial robots.

Hongseok Kim received the B.S., M.S., and Ph.D. degrees in electrical engineering from Seoul National University, Seoul, South Korea, in 1980, 1983, and 1990, respectively.

He was with the Korea Institute of Industrial Technology (KITECH), Ansan, South Korea, for 28 years, before his retirement in June 2019. He served as the Director of the Center for Robotics Industry Promotion and the Center for Intelligent Robotics, KITECH, from 2004 to 2009, supported by the Korea Ministry of Trade, Industry and Energy (MoTIE). From 2014 to 2019, he served as the General Manager of the MoTIE supported project on AI and robotics, from which he helped to establish the research direction related to this article. He is currently with the School of Electrical Engineering, Kookmin University, Seoul.