IEEE Access
Multidisciplinary | Rapid Review | Open Access Journal

Received 31 July 2023, accepted 15 September 2023, date of publication 27 September 2023, date of current version 4 October 2023.

Digital Object Identifier 10.1109/ACCESS.2023.3319822

RESEARCH ARTICLE

Error Correction in Robotic Assembly Planning From Graphical Instruction Manuals

ZHENTING WANG[1], (Graduate Student Member, IEEE), TAKUYA KIYOKAWA[1], (Member, IEEE), ISSEI SERA[2], NATSUKI YAMANOBE[3], (Member, IEEE), WEIWEI WAN[1], (Senior Member, IEEE), AND KENSUKE HARADA[1,3], (Fellow, IEEE)

[1]Graduate School of Engineering Science, Osaka University, Toyonaka 560-8531, Japan
[2]OMRON Corporation, Kyoto 600-8530, Japan
[3]ICPS Research Center, National Institute of Industrial Science and Technology (AIST), Tokyo 135-0064, Japan

Corresponding author: Zhenting Wang (ou@hlab.sys.es.osaka-u.ac.jp)

This work was supported by the New Energy and Industrial Technology Development Organization (NEDO) under Project JPNP20006.

ABSTRACT To enhance the accuracy of robotic assembly planning by understanding the graphical instruction manual, this paper proposes a novel two-step error correction method. While constructing the Assembly Task Sequence Graph (ATSG) from the instruction manual, we performed an error correction focusing on the component, symbol, speech bubble, and model number included in the manual. The component and symbol information were used to check the correctness of the manipulated components, the needed motion, and tool used in a single step of the assembly task. The speech bubble information was used to remove the repeatedly drawn components on the instruction images. The model number was recognized to distinguish components that have different models but were classified into the same class in the object detection procedure. After constructing the ATSG, we additionally performed the error correction by checking the total number of components used for the assembly task. The effectiveness of the proposed method was verified by comparing the ATSG generated from four different error correction methods of five different chairs. The results show that the proposed method decreased the influence of wrong detection results to generate an ATSG with higher accuracy and generalized the previous ATSG for different types of chairs. Finally, we built an extended ATSG from the graphical instruction manual of a kid chair, and the chair assembly task was performed by a dual-arm robot.

INDEX TERMS Error correction, instruction manual, robotic assembly, task planning.

I. INTRODUCTION

Assembly tasks are one of the most universal manufacturing tasks that are required in several industrial applications. For the furniture industry, the graphical instruction manual is the common way to give instructions to humans about how to construct a product. The graphical instruction manual for furniture is usually composed of several pages that depict the assembly procedure from individual furniture components into a complete and functional product. For most furniture instruction manuals, only significant processes are shown, and they may contain several manipulation motions on one page. For our human, we can use our knowledge and experience to understand these images, such as determining the proper motion and the appropriate task order. However, it is still a challenge for robots to automatically understand the details of the assembly task and accomplish it based on the ambiguous knowledge obtained from the implicit instruction images.

To enable robots to automatically accomplish the furniture assembly task, the first but most important problem is how to understand the instruction manual and extract accurate information that is useful for generating an assembly plan in a form that can be implemented directly by the robot.

The associate editor coordinating the review of this manuscript and approving it for publication was Zheng H. Zhu.

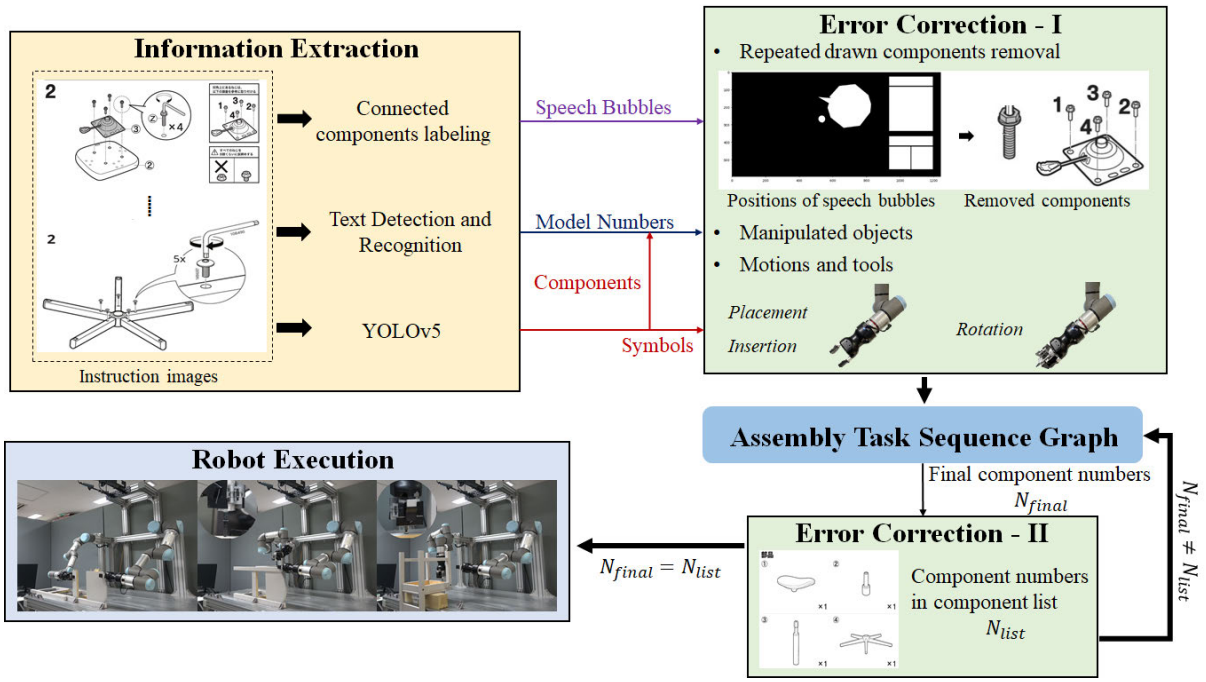


FIGURE 1. Framework of the proposed method.

In our previous study [1], we proposed a graph structure for assembly task planning called the Assembly Task Sequence Graph (ATSG), which expresses the assembly task in a form that can be realized by a robot. This method provides a simple and feasible scheme to generate a robot-implenented assembly plan directly from the given graphical instruction manual. However, it only extracted components drawn on instruction images. Other information, such as the appropriate motion and tool to implement one task, was all decided by the detected components. Even though it was a simpler method to generate the ATSG, the assembly sequence graph heavily relied on the accuracy of the component detection result. In other words, the wrong detection result could have a high possibility of generating a wrong ATSG that will mislead the robot. What's more, features of the furniture graphical instruction manual make the part detection more difficult. This is because of the overlapping between different components and repeatedly drawn components (used to highlight some information or show the details of manipulated motions). Therefore, only using detected components to generate the ATSG has a high risk of obtaining an inaccurate assembly sequence.

Errors in the ATSG generation come from wrong component detection results, which include the excess and lack of component numbers compared with the real component numbers used in the assembly task for one product. The excess of component numbers derives from two aspects. The first condition of the excess is a general definition of the wrong detection: the component is detected but classified into a wrong class, or there does not exist a component in any class but the result has detected and classified it into one class. Another condition is the repeatedly drawn components on instruction images. These repeatedly drawn components are usually used to highlight some matters that need attention, like the manipulated motion between two components. This kind of repeatedly drawn component may be detected and classified into the right class, but will not be used in the assembly task. The lack of component numbers includes the situation that the component in the image is not detected and classified into any class. Besides the furniture components, there are several other redundant visual elements drawn on the assembly instruction images that will be useful for generating robot assembly plans, such as text, symbols, and speech bubbles. This paper proposes overcoming the shortages of the previous research by using a two-step error correction method to generate a more accurate ATSG from an instruction manual. Fig.1 shows the framework of this method.

The first part of this system is the *Information Extraction*. We collected several graphical instruction manuals for different chairs. In addition to the furniture components used to generate ATSG, the symbol, speech bubble, and model number information are extracted by using different methods. The extracted elements are used to construct an ATSG with error correction. The second part of the system is the *Error Correction* for existing ATSG, which is realized by a two-step method. In the first step, speech bubbles were used to remove the repeatedly drawn components on instruction images. Next, model numbers were used to distinguish components that have different models but were classified into the

same class during the component detection procedure. Then, symbol information was used to determine the motion and the tool to perform a task. ATSG was generated first, according to the above information. In the second step of error correction, we detected components on the component list page, which is usually the first page of the graphical instruction manual. The number of components detected on the component list page was used to compare it with the component number in the generated ATSG.

Contributions of this paper are:

1. An information detection and recognition system was proposed to extract the arrow, speech bubble, and model number from the instruction images.

2. A two-step error correction method was proposed to generate a more accurate assembly task sequence graph from graphical instruction manuals.

The organization of this paper is as follows. Several related studies for robot assembly task planning and graphical instruction manual recognition are reviewed in section II. Section III summarises the method proposed in our previous study for constructing the ATSG. Section IV introduces different methods we use to extract the needed information from furniture graphical instruction manuals. Section V introduces the details of the error correction strategies. Section VI discusses the error correction results and shows the experiment for a real robot to execute an assembly task according to the extended ATSG.

II. RELATED WORKS

Assembly planning is a classical research topic in robotics. The goal of assembly is to determine an order of operations that brings individual components together to create a new product [2]. The selection of the assembly sequence has a great effect on assembly efficiency and complexity. A good assembly plan can increase the efficiency and quality and decrease the cost and time of the manufacturing process. Takamatsu et al. [3] proposed a method for recognizing assembly tasks as a sequence of movement primitives. Gu et al. [4] developed a framework for robot skill learning through human demonstration, which could automatically recognize objects, actions, and assembly states using a RGB-D camera. Thomas et al. [5] proposed a method that combines CAD-based motion planning with reinforcement learning. Knepper et al. [6] proposed an automated assembly system for the furniture. The geometry of individual components was given in a table. Kiyokawa et al. [7] proposed a method to generate easy-to-handle assembly sequences for robots by considering two tradeoff objectives about the insertion conditions and the degree of constraints among assembled parts.

Task planning from implicit instruction manuals plays an important role in automatic assembly tasks of furniture for robots. A deeper and precise understanding of the instruction manual can generate a proper and clear assembly sequence. Nevertheless, the feature of the graphical instruction manual brings some difficulties in using it. One problem is that the instruction manual just gives a discrete description of a continuous assembly task. Without having prior knowledge of the assembly procedure and the product to be assembled, it is hard to understand the instruction manual. Another problem is the overlapping between different components, which makes it difficult to recognize objects drawn on the instruction image. To overcome the overlapping problem, Shao et al. [8] presented a technique that took the input as a multi-step assembly instruction in a vector graphic format and grouped vector graphic primitives into semantic elements like components, mechanical connectors, arrows, highlights, and numbers. Park et al. [9] proposed a method to understand assembly instructions in manuals by using both conventional non-learning approaches and deep neural networks. They first extracted components having strict geometric structures by using conventional non-learning algorithms, and then used deep neural networks to recognize the extracted components. Lee et al. [10] proposed a method to recognize the speech bubble area using Cascade Mask R-CNN and developed a context-aware data augmentation scheme for speech bubble segmentation.

In this paper, based on characteristics of different elements on the instruction image, different detection and recognition methods are used to extract components, symbols, speech bubbles, and model numbers.

III. ATSG GENERATION FROM GRAPHICAL INSTRUCTION MANUALS

To ensure the assembly plan can be executed by a robot with a single manipulator, the plan must be sequential to make sure there is only one subassembly at a time. The assembly task sequence graph is a directed graph with nodes indicating components, motions, and tools, and edges indicating the direction [1]. The smallest unit structure of the ATSG is the assembly unit, which consists of multiple input object nodes, one motion node, and one output object node. The output object node is divided into two kinds of components based on the size of the components. One is the main child component, which is the biggest component among the multiple input object nodes. The remaining components are named as sub-child components. The name of the output object node is determined by the name of the main child component. To ensure the generated assembly graph embodies the task drawn on each instruction image in a particle size that the robot can realize, one constraint is given to the input object node: only two input object nodes can exist in one assembly unit. If an assembly unit has three or more input object nodes, the assembly unit is divided into numerous connected assembly units in accordance with the appropriate task order to guarantee that there are only two input object nodes in one assembly unit.

The existing ATSG was generated from the detected types of components drawn on instruction images. First, components drawn on instruction images were extracted. The extracted components were used to decide the motion,

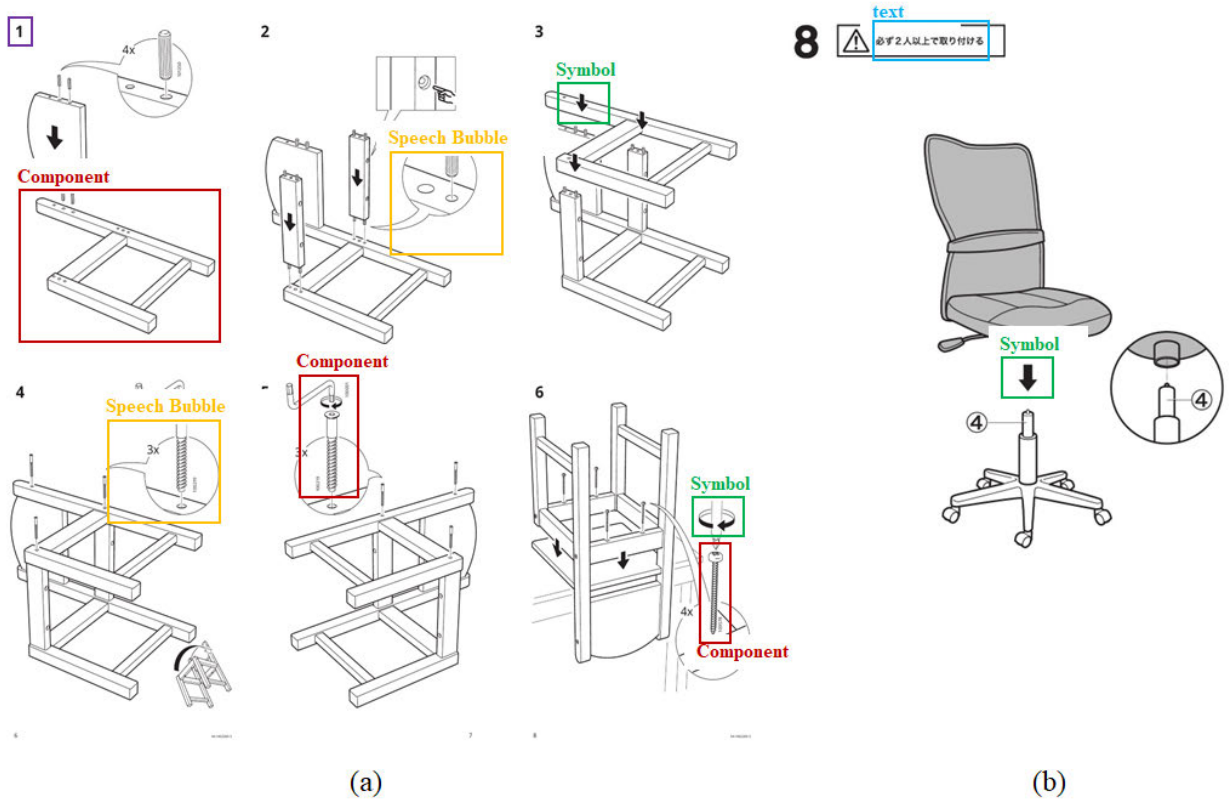


FIGURE 2. Visual elements excluded in instruction manuals. (a) is a series instruction images from *Ikea* of a kid chiar. (b) is one page of the instruction manual from *Nitori* of an office chair. Elements in red, green, yellow, purple, and blue boxes are manipulated objects, symbols, speech bubbles, numbers, and texts, respectively.

and the tool will be used in each assembly step. The existing ATSG gives a graph structure that can decompose the whole assembly task into several sub-tasks that can be implemented by the robot directly based on the task order obtained from the instruction manual. However, because all of the information needed for the assembly task, like the manipulated components and motions for manipulating these components, was decided only by the type of components, the wrong detection result of components has a great effect on the generated ATSG.

Compared to the aforementioned studies, in this paper, we extend the existing ATSG by adding the symbol, speech bubble, and model number information extracted from the graphical instruction manual to build an ATSG with high precision. The symbol information is used to determine the motion. The speech bubble information is used to remove the repeatedly drawn components on the instruction images. The model number information is used to distinguish components with different models.

IV. GRAPHICAL INSTRUCTION MANUAL INFORMATION EXTRACTION

Graphical instruction manuals are widely used in furniture products. Besides furniture components, there are lots of other visual elements in assembly instruction images that will be useful for generating assembly sequences, as shown in Fig.2 in boxes with different colors.

1. Furniture components depict shapes of parts that the user needs to manipulate in each step.

2. Symbols, like arrows, usually indicate how components are connected, which is the key to deciding the motion needed to be manipulated in each step.

3. Numbers are always used to indicate the order of assembly steps, the model numbers of each part, and the numbers of different parts used in each step.

4. Speech Bubbles are always used to show the detailed shape of some small components, model numbers, or highlight the motion between two components at a specific step and location.

5. Texts usually give detailed information or points that need to be paid attention to about the motion.

Many researchers worked on finding domain-specific design principles in order to create effective visualization in graphical instructions. Agrawala et al. [11] identified three main principles for the assembly instructions from human-subject experiments. The first one was using a step-by-step sequence of diagrams showing one primary action in each diagram. Thus, only a few task procedures are drawn, and it is necessary to understand the needed assembly information from the discontinous scenes. The second one

was using guidelines and arrows to depict actions required to fit components together. The third one was to ensure components added in each step were visible. These principles give some guidelines for generating assembly task sequences directly from instruction images.

In this paper, we extract the furniture component, symbol, speech bubble, and model number information, and use them to correct the existing ATSG to generate a new ATSG with a higher accuracy. We use the furniture instruction manual from *Nitori*[1] and *Ikea*.[2] While the instructions of different companies differ slightly in the visualization style, they all use similar diagrammatic elements, and what we are concerned about, the furniture component, arrow, and speech bubble, are expressed in a similar form.

A. FURNITURE COMPONENT AND ARROW DETECTION

Arrows are one of the most commonly used graphical elements, which are used to point out the order of sequences, connect elements, and indicate motions [12]. Two kinds of arrows are usually used in the furniture instruction manual to indicate the action. One is the 2D arrow that is used to point the direction, and the other is the 3D arrow with a curvy path that is used to express the change of the orientation, especially the rotation of screws. Therefore, we focus on these two arrows in this paper. The arrow information is used to decide the motion and the tool to connect two components. We define three motions in the furniture assembly task, as shown in TABLE 1. The first one is *Insertion*, which is assigned when detecting 2D arrow in the instruction image. The second one, *Rotation*, is assigned when detecting 3D arrow. The last one is *Placement*. Placement is assigned when there are three or more components that need to be assembled on one page of the instruction manual, and there are no arrows near the detected components. There may be other motions that are also very important for the whole assembly task, such as turning over the subassembly to implement the next assembly motion. This kind of motion is not for two different individual components, so it cannot be expressed in the ATSG structure. However, this motion is not the essential motion we are concerned about, and fortunately, this motion is able to be handled in the motion planning process as changing the pose of the manipulated object.

TABLE 1. Three motions in the furniture assembly task.

Arrow			Null
Motion	*Insertion*	*Rotation*	*Placement*

YOLOv5 [13] is used to detect furniture components and arrows. To construct a dataset of furniture assembly instruction manuals, we collect instruction images of different chairs from *Nitori* and *Ikea*. Each instruction manual is segmented according to the step number. Segmented images for one manual are saved in sequence to guarantee the order of assembly steps. Three data augmentation methods are performed. The first one is the horizonal flip. Because the segmented images are of different sizes, we then implement cropping. Cropping images is a practical processing step for images with mixed height and width dimensions by cropping a central patch of each image [14]. The last one is the rotation. Finally, we obtain 2137 images. Among these images, 1875 images are used to train the model, 87 images for testing, and 175 images for validating. The mean average precision of the trained model is 0.873.

B. SPEECH BUBBLE EXTRACTION

Speech bubble extraction is a common procedure in comics. There are two methods for speech bubble extraction. One method is based on the neural network. Dubray et al. [15] developed a method to automatically detect and segment speech bubbles, including the carrier and tails in comic books based on a deep CNN. Another method is to perform edge detection on the near surroundings of the text zones. It is necessary to use the Optical Recognition System (OCR) to recognize characters first when using this method. Rigaud et al. [16] proposed a method in which the speech bubbles were located first, and the text zones were only used to give a confidence.

In furniture instruction manuals, the speech bubble is usually used to highlight some important elements, such as the indispensable motion to attach two components and the model number of the manipulated component. In this paper, we combine topological and spatial position relationships to segment speech bubbles at the pixel level. First, speech bubbles are distinguished by the connected components labeling, which is an indispensable operation in computer vision to detect connected regions in a binary image. Connected components labeling groups pixels in an image into components based on pixel connectivity. All pixels in a connected component share similar pixel intensity values and are connected with each other. After the connected components labeling, we extract each speech bubble on the segmented instruction image by its label. Then, the area of the bounding box of the connected component is calculated. The connected component, whose area occupies 5%∼30% of the image area, is retained as the speech bubble that needs to be checked later. After this, contours of speech bubbles in connected components are found. Fig.3 shows the detected speech bubbles of one image in a chair instruction manual. Fig.3(a) shows the image after preprocessing, such as adaptive thresholding and erosion. Fig.3(b) shows the result of rough estimation of bounding boxes using connected components. Fig.3(c) shows the contours of speech bubbles in the original image. The speech bubble information is used in the first step of the error correction.

[1]https://www.nitori-net.jp/ec/
[2]https://www.ikea.com/jp/ja/

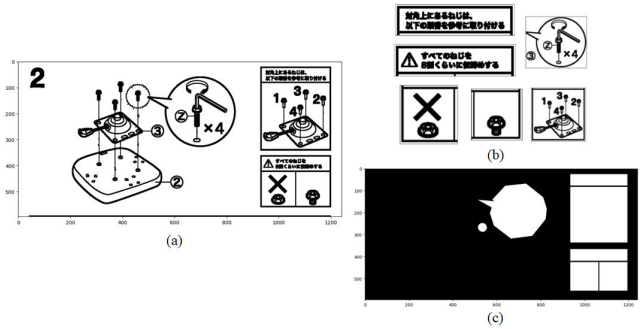


FIGURE 3. Speech bubbles extraction results.

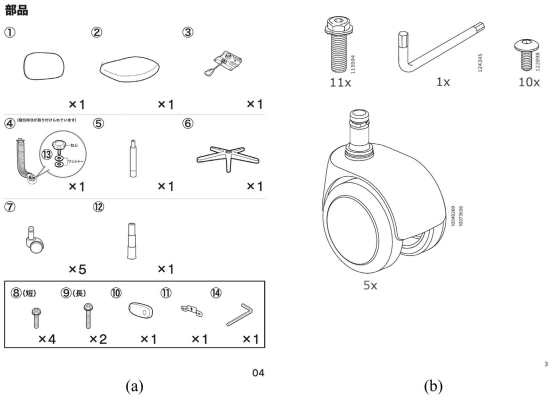


FIGURE 4. Part list page in the furniture instruction manuals from Nitori (a) and Ikea (b).

C. MODEL NUMBER RECOGNITION

A furniture instruction manual usually has one page to show all the components used in this assembly task. Fig.4 show two examples of part list page in the furniture instruction manuals from *Nitori* and *Ikea*. There are two different models of screws in these two assembly tasks. However, from the component detection process, only "screw" class is defined. Therefore, if the ATSG is constructed just using the detected result from YOLOv5, it cannot decide which kind of screw is used. Therefore, it is necessary to detect the model number of the same class components.

We used the MMOCR [17][3] to recognize the model number on the instruction images, which is an open-source toolbox for text detection, text recognition, and the corresponding downstream tasks including key information extraction. MMOCR supports a wide variety of state-of-the-art models for text detection and recognition for different expressions of the text.

We confirm components in one class but have different model numbers by comparing the model number recognized from the component list page with the model number recognized from each frame image. Therefore, both text detection and text recognition in MMOCR were used in this paper. The FCENet [18] is used to detect the text on instruction images.

[3]https://github.com/open-mmlab/mmocr

TABLE 2. Text detection and recognition results.

Frame No.	Text Recognition Results
Frame 1	['4x', '1', '101350', '000', '3']
Frame 2	['4x', '2', 'AA-1462260-3', '4']
Frame 3	['3', '5']
Frame 4	['3x', 'AA-1462260-3', '100219', '6', '100001', '4']
Frame 5	['3x', '5', '100219', '100001', '7']
Frame 6	['4x', 'AA-1462260-3', '6', '8', '109578']

The proposed Fourier Contour Embedding (FCE) method enables us to approximate text contours with arbitrary closed shapes. For text recognition, we use the SAR (Show, Attend and Read) [19] method, which is an easy-to-implement strong baseline for recognizing irregular text in images. The text detection and recognition results for Fig.2(a) are shown in TABLE 2.

There is a wrong detection and recognition result in Frame 1 as '000'. This wrong result comes from the hole on the leg in Frame 1. The shape of the hole is misrecognized as the '000'. However, this misrecognition will not affect the generation of the ATSG by comparing the recognition result of each frame with the recognition result of the component list page.

V. ERROR CORRECTION FOR ASSEMBLY SEQUENCE

In our previous research [1], the ATSG was generated just according to the detected components on the instruction images, which means the motion node and the tool node are also decided by the detected components. Although this method makes it easier to build the ATSG, it also causes some problems, especially on the accuracy of the generated ATSG. The inaccuracy mainly comes from two aspects, the number of components and the motion for each assembly step. Therefore, in this section, we first analyze the wrong generation, which comes from the wrong number of components and the wrong motion. Then, based on these two aspects, we proposed our two-step error correction method.

A. NUMBER OF COMPONENTS

The influence of number of components on the generation of ATSG includes two situations. The first source of errors in the generated ATSG is the repeatedly drawn components on the instruction images, as components *Screws* shown in Fig.5. In the furniture instruction manual, the repeatedly drawn components are usually located inside the speech bubble. Speech bubbles in the instruction image are used to show some small components and the model number of the component, while these components are already drawn outside of speech bubbles. Therefore, components drawn in the speech bubble will not be used in the real assembly task. If the repeatedly drawn components remain, a wrong ATSG will be generated with an inaccurate number of manipulated components.

Second is the existence of overlapping between different components on the instruction images, which will cause

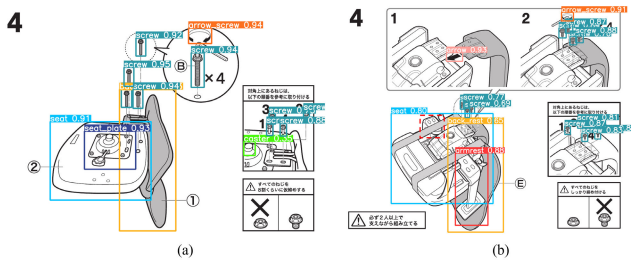


FIGURE 5. Wrong detection for number of components. (a) and (b) are detection results for two frames of two different chairs' assembly instructions. Both (a) and (b) show the first condition, in which components in speech bubbles have also been detected and counted as the manipulated components in this assembly step. (b) shows a case of wrong detection, missing the *Armrest* in the dashed box, caused by overlapping between the *Seat* and this *Armrest*.

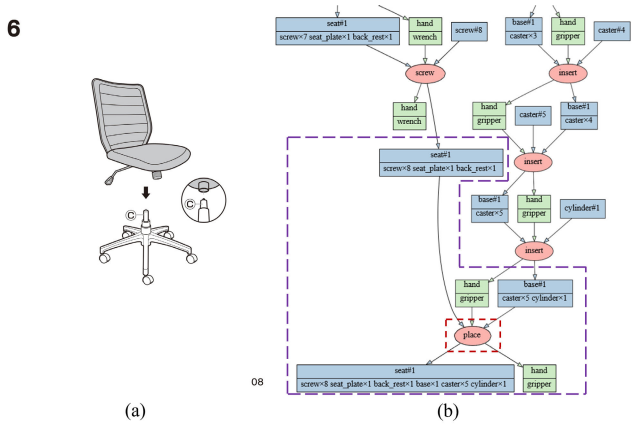


FIGURE 6. One case of wrong estimation of motion node in ATSG.

wrong results in the component detection as shown in Fig.5(b).

B. MOTION

The second is about the motion node in the ATSG. For the existing ATSG, the motion node is decided by the main child component based on the concept of action relationship [20], which generates the possible assembly motions from the manipulated object. The output object node in the assembly unit contains all components that were manipulated in and before this step. Among all the components, the main child component in the output object node is decided by the size of the component. The motion for one assembly step in an assembly unit is determined by the action relationship result of the main child component. This motion decision method will generate a wrong motion node when the attachment of two assembled parts occurs at two components that are not the main child components in the output object nodes in two assembly units. Fig.6 shows one case of wrong estimation of the motion node in ATSG of one step. Fig.6(a) is one page from the instruction manual for a *Nitori* office chair. Fig.6(b) shows part of the generated ATSG using the previous method. This page shows one step of the assembly, which means inserting the upper assembled part into the lower assembled part. However, the motion node in the ATSG is "place". This is because the main child component of the upper assembled part is the "seat", and the main child component of the lower assembled part is the "base". Therefore, the motion for the seat and base estimated by the action relationship is *Placement*.

C. TWO-STEP ERROR CORRECTION METHOD

A two-step error correction method is considered to overcome problems mentioned above.

There may be an excess or a lack of the detected component number compared with the real component number used in the assembly task. The excess of the component number comes from two cases. One case is caused by the repeatedly drawn components located in speech bubbles. From the result of speech bubble extraction, it is able to get the location of speech bubbles in one instruction image. The locations of detected components are expressed using the bounding box obtained from the YOLOv5 detection result. If any vertex of the bounding box of one component lies inside the extracted speech bubbles, this component is removed. Another case of excess comes from the wrong detection in YOLOv5. The wrong detection in YOLOv5 has two meanings. One meaning is that a component is classified into a wrong class, which can be farther divided into two situations by whether this component is used in this assembly or not. If the detected component in one step is not used in the whole assembly task, this component is deleted directly. The error correction is performed following the generation of the ATSG for the component that is used in the assembly task but is classified incorrectly.

Another meaning of wrong detection is that components are drawn on the image but cannot be detected and classified into any class, which causes the lack of some components during generating ATSG. For components that were recognized in the wrong classes and the lack of components, we check whether the number of this component in the final output object node of ATSG (the final product) is equal to that in the component list page in the instruction manual after generating the assembly sequence. If the number of a component in the final product is greater than that in the component list page, then assembly units containing the unnecessary number of the component are removed from the assembly sequence. Assembly units containing the missing components are added to the assembly sequence.

For the second problem, instead of just using the detected components to decide motion nodes, we use detected arrows to estimate motions to avoid generating wrong motions caused by the lack of information about components' relationships in the output object node, which is composed of several components. By using the arrow information, the decision for the motion node is totally independent of the output object node.

TABLE 3. Error correction results for motion nodes.

Instructions	Method A	Method B	Method C	Method D
Nitori_1	1	0.61	1	0.5
Nitori_2	1	0.52	1	0.33
Nitori_3	0.94	0.82	0.94	0.52
Ikea_1	1	1	1	0.86
Ikea_2	1	0.94	1	0.89

TABLE 4. Component number in each frame from Yolov5 detection results.

Instructions	Frame						Total[1]
	1	2	3	4	5	6	
Nitori_1	12(5)	13(6)	9(3)	7(0)	9(0)	-	13
Nitori_2	9(3)	7(0)	13(4)	15(8)	11(0)	20(6)	20
Nitori_3	14(8)	5(1)	3(0)	9(3)	8(0)	9(0)	12
Ikea_1	5(0)	9(0)	7(0)	7(1)	7(1)	9(0)	2
Ikea_2	7(1)	7(1)	7(0)	7(1)	15(8)	-	11

[1] The total number of components in speech bubbles.
* The number in the bracket is the number of components in speech bubbles.

VI. ANALYSIS AND EXPERIMENTS

A. ERROR CORRECTION RESULTS

To show the effectiveness of the proposed two-step error correction method, we compared the error correction result using the proposed method with other three methods.

A) Just using the arrow and speech bubble information to correct the error during the assembly sequence generation process;

B) Just checking for component number consistency;

C) Using the arrow and speech bubble information during the assembly sequence generation process and checking for component number consistency after generating the ATSG (proposed method);

D) Nothing.

Five chairs are used to check the error correction results. Three of them are from *Nitori* and two of them are from *Ikea*. All instruction manuals are available for download from the websites of *Nitori* and *Ikea*.

First, motion nodes in the generated ATSG are considered. We first generate the assembly graph from graphical instruction manuals by human. Then, we compare motion nodes in the ATSG generated by four different methods. We define a parameter A to describe the error correction result of motion nodes for different chairs using different methods.

$$A = \frac{correct\ number\ of\ motion\ nodes}{total\ number\ of\ motion\ nodes} \tag{1}$$

TABLE 3 shows the result of motion nodes for five chairs using four methods. For each instruction, parameter A is calculated using four different error correction methods. When the parameter A is equal to 1, motion nodes generated from one of four different methods are totally the same as the motion nodes generated by humans watching the instruction, which means motions generated by this error correction method can be used in the real assembly task.

From TABLE 3, the proposed method (Method C) can generate the right motion for each assembly step for different instructions from different companies. For motion node generation, Method A has the same result as the proposed method (Method C), both methods use the arrow information to generate the motion node in each assembly step. Therefore, the error correction result is also the same. The accuracy of motion nodes for the assembly sequence generated from the instruction *Nitori_3* is 0.94, not 1. This is because in one frame of this manual, without showing the motion, the arrow is used to show the direction of the component when it is manipulated.

Second, we obtain the number of components in each frame of five instruction manuals using Yolov5 detection results, as shown in TABLE 4.

Then, we compare the total component number (component number obtained from the final output object node) of the ATSG generated by four different error correction methods with the total component number shown on the component list page in the graphical instruction manual. TABLE 5 shows the error correction result of the total component number. From TABLE 5, we can see that by using the speech bubble information during generating the assembly sequence and checking the total component number after generating the assembly sequence, it is able to decrease the effect of the incorrect object detection and the repeatedly drawn object on the graphical instruction manual. In detail, the component number in the final output object node of the ATSG generated by Method C (the proposed method) of all five instructions is the same as the component number shown on the component list page. Comparing the component number of the ATSG generated by Method D (directly generating the ATSG from detected components on instruction images) and the component number generated by Method A (only using the speech bubble information to remove repeated components) with the component number from the list, we can see that by removing the repeatedly drawn components in speech bubbles, it is possible to construct an ATSG with the component number closer to the right component number. This is because the wrong number of components in the component detection result usually comes from the repeatedly drawn components in speech bubbles.

For the error correction method that only checks the consistency of the component number between the generated ATSG and the component list page (Method B), only using the component number information makes it hard to know which step needs to add or remove the excess or lacking components, and how many components need to be added or removed in this step. Therefore, it usually generates a totally wrong ATSG with a wrong component number and a wrong task sequence. The effect of checking the component number consistency is to remove the component that is incorrectly classified into a class that is not used in this assembly task. For

TABLE 5. Error correction results for component number.

Instructions	Method A	Method B	Method C	Method D	Component number from list
Nitori_1	19	15	18	32	18
Nitori_2	25	20	24	45	24
Nitori_3	17	14	17	29	17
Ikea_1	27	18	27	29	27
Ikea_2	18	18	18	29	18

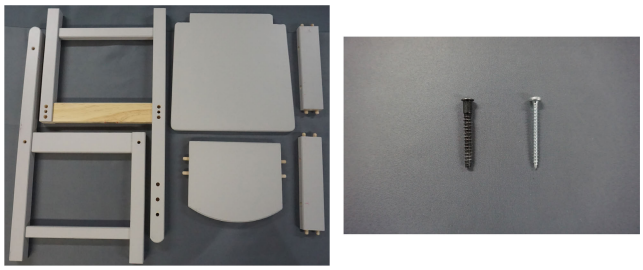


FIGURE 7. Components of a kid chair used to execute robot assembly task according to generated ATSG with error correction.

example, in the instruction manuals *Nitori*_1 and *Nitori*_2, there are two wrong detections in which the detection results include a component that is not used in the assembly in Frame 2 and Frame 3, respectively.

B. ROBOT EXECUTION

The generated ATSG of Ikea_1, a kid chair, is executed by a dual-arm UR3e robot. Fig.7 shows components used in this chair assembly task. Two kinds of screws are used to fix different components. The instruction manual of this chair is shown in Fig.2(a). From the instruction manual shown in Fig.2(a), in Frame 1, Frame 2, and Frame 3, 2D arrows were detected, therefore, the motion for these assembly steps is *Insertion*. In Frame 4 and Frame 5, 3D arrows and six screws with the same model number, as '100219' were detected, motion shown in these two frame is *Rotation*. In Frame 6, both 2D and 3D arrows were detected, therefore, motion in this frame is, first, the *Insertion*, and second, the *Rotation*. Only two of three motions introduced in IV will be used in this chair's assembly task.

The suited gripper and tool for these two motions are shown in Fig.8. For the *Rotation*, the screw-tightening tool designed by Hu et al. [21] is used. This tool is able to convert the gripping motion of two-finger parallel grippers into a rotation motion to realize the fastening screw task. This screw-tightening tool is splendid for fastening a screw in a narrow space, which is a condition that usually occurs in furniture assembly tasks. The planning strategy is based on the motion planning framework proposed by Wan et al. [22]. When given the initial pose and the goal pose of the objects, it is able to generate a sequence of grasps to pick up the object, move the object to the goal pose, and orientate the object. By using this system, it is able to plan the turning

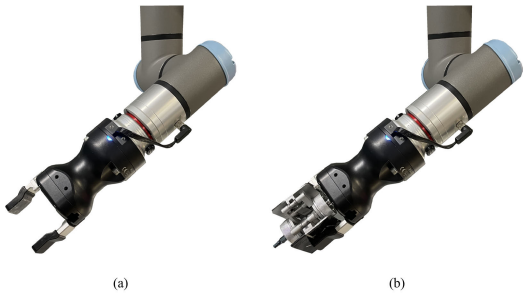


FIGURE 8. Tools for different motions estimated by the detected arrows. (a) is used to perform the *Insertion* and *Placement*. (b) is used to perform the *Rotation*.

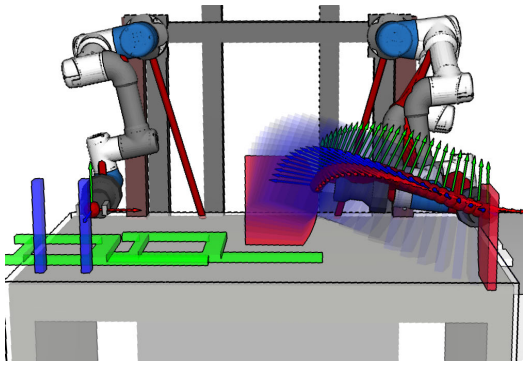


FIGURE 9. The result of the RRT connection for generating the assembly path of picking a part and then moving it to the goal pose.

over motion automatically if necessary. The RRT-connection [23] is used to generate the assembly path. Fig.9 shows the RRT-connection result for the sub-task inserting the back rest into the leg in the simulation environment.

Fig.10(a) shows the generated ATSG with the proposed error correction method, and Fig.10(b) presents the chair assembly experiment by the robot. All components shown in Fig.7 are put on the pre-defined position as shown in Fig.10(b)(i). Fig.10(b)(ix) shows the completed assembled chair. Fig.10(b)(ii)-(viii) are several robot assembly phases according to the generated ATSG. Fig.10(b)(ii)-(v) show the *Insertion* between different components. Fig.10(b)(vi)-(viii) show the screwing (Rotation) motion according to the ATSG in the blue dotted box in Fig.10(a). From the experiment, the generated ATSG is able to be executed by the robot.

VII. CONCLUSION AND FUTURE WORK

In this paper, we propose a novel two-step error correction method to enhance the accuracy of assembly task planning for robots by automatically understanding the graphical instruction manual. The inaccuracy of the existing assembly sequence generation method principally comes from the inaccurate object detection result. Therefore, to weaken the serious dependence on the object detection result, other visual elements in the graphical instruction manual, like the symbol, speech bubble, and model number, are detected and used to generate the assembly plan for robots with

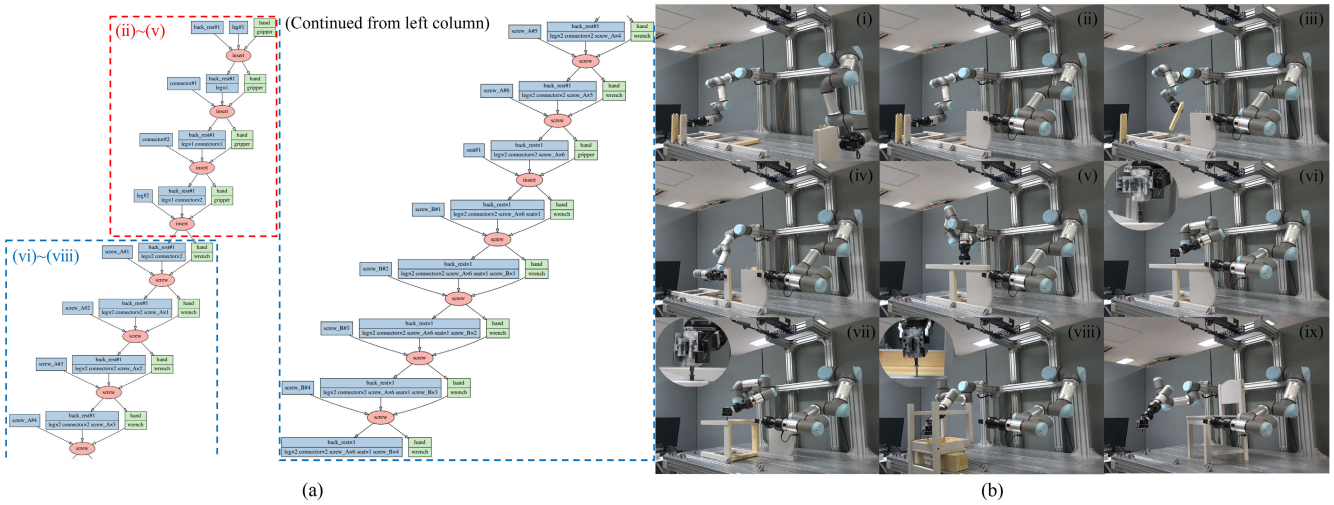


FIGURE 10. Generated ATSG and robot execution. (a) is the corresponding ATSG of the instruction manual shown in Fig.2(a). Fig.10(b) shows the assembly experiment by the robot. Fig.10(b)(i) is the initial environment setting of the assembly task. Fig.10(b)(ix) shows the completed product. Fig.10(b)(ii)-(viii) are several robot assembly phases.

a higher precision. Furniture components and arrows are detected by YOLOv5. Speech bubbles are extracted based on the connected components labeling. The model numbers are recognized using the open-source toolbox MMOCR. The extracted elements are used in the two-step error correction. The first step is during the generation of the assembly sequence. The appropriate motion is decided by the detected arrow information. The speech bubble is used to remove the repeatedly drawn components on instruction images. The detected component is checked to see if it is included in the component list page. The second step is after the generation. By checking the number of each component in the final output of the assembly sequence with the number of each component in the component list page, the correct component number is guaranteed. From the compared results of the generated ATSG in four different error correction methods, we can see that the proposed two-step error correction method is effective. Finally, a robot experiment assembling a kid's chair is executed based on the generated assembly sequence.

In the future, we will look at two major issues that remain in this topic. One is the detection of components. Even though we have obtained reasonable accuracy using YOLOv5 in this study for different kinds of chairs, for other furniture, the detection result may not be good because of the overlapping of different components. Another is focusing on the robot execution phase. Because the furniture is always big and heavy, it is difficult to accomplish the whole assembly task just by the robot itself. It is expected to find some methods to increase the assembly efficiency by human-robot collaboration.

REFERENCES

[1] I. Sera, N. Yamanobe, I. G. Ramirez-Alpizar, Z. Wang, W. Wan, and K. Harada, "Assembly planning by recognizing a graphical instruction manual," in *Proc. IEEE/RSJ Int. Conf. Intell. Robots Syst. (IROS)*, Sep. 2021, pp. 3138–3145.

[2] Z. Zhu and H. Hu, "Robot learning from demonstration in robotic assembly: A survey," *Robotics*, vol. 7, no. 2, p. 17, Apr. 2018.

[3] J. Takamatsu, K. Ogawara, H. Kimura, and K. Ikeuchi, "Recognizing assembly tasks through human demonstration," *Int. J. Robot. Res.*, vol. 26, no. 7, pp. 641–659, Jul. 2007.

[4] Y. Gu, W. Sheng, C. Crick, and Y. Ou, "Automated assembly skill acquisition and implementation through human demonstration," *Robot. Auto. Syst.*, vol. 99, pp. 1–16, Jan. 2018.

[5] G. Thomas, M. Chien, A. Tamar, J. A. Ojea, and P. Abbeel, "Learning robotic assembly from CAD," in *Proc. IEEE Int. Conf. Robot. Autom. (ICRA)*, May 2018, pp. 3524–3531.

[6] R. A. Knepper, T. Layton, J. Romanishin, and D. Rus, "IkeaBot: An autonomous multi-robot coordinated furniture assembly system," in *Proc. IEEE Int. Conf. Robot. Autom.*, May 2013, pp. 855–862.

[7] T. Kiyokawa, J. Takamatsu, and T. Ogasawara, "Assembly sequences based on multiple criteria against products with deformable parts," in *Proc. IEEE Int. Conf. Robot. Autom. (ICRA)*, May 2021, pp. 975–981.

[8] T. Shao, D. Li, Y. Rong, C. Zheng, and K. Zhou, "Dynamic furniture modeling through assembly instructions," *ACM Trans. Graph.*, vol. 35, no. 6, pp. 1–15, Nov. 2016.

[9] J. Park, I. Kang, J. Kwon, E. Lee, Y. Kim, S. You, S. H. Ji, and N.-I. Cho, "Recognition of assembly instructions based on geometric feature and text recognition," in *Proc. 17th Int. Conf. Ubiquitous Robots (UR)*, Jun. 2020, pp. 139–143.

[10] J. Lee, S. Lee, S. Back, S. Shin, and K. Lee, "Object detection for understanding assembly instruction using context-aware data augmentation and cascade mask R-CNN," 2021, *arXiv:2101.02509*.

[11] M. Agrawala, W. Li, and F. Berthouzoz, "Design principles for visual communication," *Commun. ACM*, vol. 54, no. 4, pp. 60–69, Apr. 2011.

[12] R. E. Horn, *Visual Language*. Washington, DC, USA: MacroVu, 1998.

[13] G. Jocher et al., "ultralytics/yolov5: v5.0—YOLOv5-P6 1280 models, AWS, Supervise.ly and YouTube integrations," Zenodo, 2021.

[14] C. Shorten and T. M. Khoshgoftaar, "A survey on image data augmentation for deep learning," *J. Big Data*, vol. 6, no. 1, pp. 1–48, Dec. 2019.

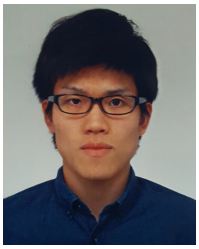
[15] D. Dubray and J. Laubrock, "Deep CNN-based speech balloon detection and segmentation for comic books," in *Proc. Int. Conf. Document Anal. Recognit. (ICDAR)*, Sep. 2019, pp. 1237–1243.

[16] C. Rigaud, J.-C. Burie, and J.-M. Ogier, "Text-independent speech balloon segmentation for comics and manga," in *Proc. 11th Int. Workshop Graphic Recognit. Current Trends Challenges (GREC)*. Nancy, France: Springer, 2017.

[17] Z. Kuang, H. Sun, Z. Li, X. Yue, T. H. Lin, J. Chen, H. Wei, Y. Zhu, T. Gao, W. Zhang, K. Chen, W. Zhang, and D. Lin, "MMOCR: A comprehensive toolbox for text detection, recognition and understanding," in *Proc. 29th ACM Int. Conf. Multimedia*. New York, NY, USA: Association for Computing Machinery, 2021, pp. 3791–3794.

[18] Y. Zhu, J. Chen, L. Liang, Z. Kuang, L. Jin, and W. Zhang, "Fourier contour embedding for arbitrary-shaped text detection," in *Proc. IEEE/CVF Conf. Comput. Vis. Pattern Recognit. (CVPR)*, Jun. 2021, pp. 3122–3130.
[19] H. Li, P. Wang, C. Shen, and G. Zhang, "Show, attend and read: A simple and strong baseline for irregular text recognition," in *Proc. AAAI Conf. Artif. Intell.*, vol. 33, 2019, pp. 8610–8617.
[20] K. Fukuda, I. G. Ramirez-Alpizar, N. Yamanobe, D. Petit, K. Nagata, and K. Harada, "Recognition of assembly tasks based on the actions associated to the manipulated objects," in *Proc. IEEE/SICE Int. Symp. Syst. Integr. (SII)*, Jan. 2019, pp. 193–198.
[21] Z. Hu, W. Wan, K. Koyama, and K. Harada, "A mechanical screwing tool for parallel grippers—Design, optimization, and manipulation policies," *IEEE Trans. Robot.*, vol. 38, no. 2, pp. 1139–1159, Apr. 2022.
[22] W. Wan, K. Harada, and F. Kanehiro, "Preparatory manipulation planning using automatically determined single and dual arm," *IEEE Trans. Ind. Informat.*, vol. 16, no. 1, pp. 442–453, Jan. 2020.
[23] J. J. Kuffner and S. M. LaValle, "RRT-connect: An efficient approach to single-query path planning," in *Proc. ICRA. Millennium Conf., IEEE Int. Conf. Robot. Automat. Symposia*, 2000, pp. 995–1001.

ZHENTING WANG (Graduate Student Member, IEEE) received the B.E. degree from the College of Information Science and Engineering, Northeastern University, Shenyang, China, and the M.E. degree from Osaka University, Osaka, Japan, where she is currently pursuing the Ph.D. degree with the Graduate School of Engineering Science. Her research interests include robot motion planning, robot manipulation, and human-robot collaboration.



TAKUYA KIYOKAWA (Member, IEEE) received the B.E. degree from the Kumamoto College, National Institute of Technology, Japan, and the M.E. and Ph.D. degrees in engineering from the Nara Institute of Science and Technology, Japan, in 2018 and 2021, respectively. Since 2021, he has been with Osaka University, Japan, as a specially-appointed Assistant Professor and the Nara Institute of Science and Technology as a specially-appointed Assistant Professor. His current research interests include robot manipulation and robot vision for reconfigurable robotic systems toward agile manufacturing. He is a member of RSJ, JSME, SICE, and JSAI.

ISSEI SERA received the master's degree from Osaka University, Osaka, Japan, in 2021. He is currently a Technology Strategy Planner with Technology and Intellectual property H. Q. of OMRON Corporation. His research interest includes semantic understanding in robotics.

NATSUKI YAMANOBE (Member, IEEE) received the M.E. and Ph.D. degrees from The University of Tokyo, in 2004 and 2007, respectively. She is currently a Senior Researcher with the Industrial Cyber-Physical Systems Research Center, National Institute of Advanced Industrial Science and Technology (AIST), and a Guest Associate Professor with the Tokyo University of Agriculture and Technology. Her research interests include robotic manipulation, human-robot interaction, and skill analysis/transfer for dexterous manipulation.

WEIWEI WAN (Senior Member, IEEE) received the Ph.D. degree in robotics from the Department of Mechano-Informatics, The University of Tokyo, Tokyo, Japan, in 2013. From 2013 to 2015, he was a Postdoctoral Research Fellow of the Japan Society for the Promotion of Science, Tokyo, and a Visiting Researcher with Carnegie Mellon University, Pittsburgh, PA, USA. He was a tenure-track Research Scientist with the National Advanced Institute of Science and Technology (AIST), Tokyo. He is currently an Associate Professor with the School of Engineering Science, Osaka University, Osaka, Japan. His research interests include smart manufacturing and robotic manipulation. He is a member of RSJ.

KENSUKE HARADA (Fellow, IEEE) received the Ph.D. degree from Kyoto University, Kyoto, Japan, in 1997. From 1997 to 2002, he was a Research Associate with Hiroshima University, Hiroshima, Japan. From 2005 to 2006, he was a Visiting Scholar with the Computer Science Department, Stanford University, Stanford, CA, USA. Before joining Osaka University, he was a Researcher with the National Institute of AIST, Tsukuba, Japan. He is currently a Professor with the Graduate School of Engineering Science, Osaka University. His research interests include the mechanics and control of humanoid robots and robotic hands.