Motion Planning for a Dual-arm Robot from Language Instructions and Cooking Images

Kota Takata [1], Takuya Kiyokawa [1], Ixchel G. Ramirez-Alpizar [1], Natsuki Yamanobe [1], Weiwei Wan [1], and Kensuke Harada [1]

[1]Affiliation not available

October 30, 2023

Efficient Task/Motion Planning for a Dual-arm Robot from Language Instructions and Cooking Images

Kota Takata[1], Takuya Kiyokawa[1], Ixchel G. Ramirez-Alpizar[2], Natsuki Yamanobe[2]
Weiwei Wan[1], and Kensuke Harada[1,2]

***Abstract*— When generating robot motions based on instructions such as cooking recipes, ambiguity of the instructions and lack of necessary information are problematic for the robot. To solve this problem, we propose an efficient motion planning approach for a dual-arm robot by constructing a graph representing a motion sequence based on a recipe consisting of verbal instructions and cooking images. A functional unit is generated based on the linguistic instructions in the recipe. Since most recipes lack the necessary information for executing the motion, we first consider extracting the information about the cooking motion like cutting from the food images of the recipe and supplementing it. In addition, to supplement the actions that humans perform unconsciously, we generate functional units for actions not explicitly mentioned in the recipe based on the current situation of the cooking process, and then connect them to the functional units generated from the recipe. Moreover, during the connection we consider the motion of the robot's arms in parallel for an efficient execution of the recipe, similar to those of a human. Through experiments, we demonstrate that for a given recipe, the proposed method can be used to generate a cooking sequence with the supplementary information needed, and executed by a dual-arm robot. The results show that the proposed method is effective and can simplify robot teaching in cooking tasks.**

I. Introduction

When a robot performs a task based on a cooking recipe, problems such as ambiguous instructions or missing information must be solved. In particular, there are cases where the written instructions contain ambiguous descriptions, and/or the information necessary to execute the task is not available (or missing). For example, a written instruction such as "cut the food into pieces of appropriate size" is not enough for the robot to execute it. In this case, it is necessary to complement the missing information, this can be done by estimating the processing method of the ingredients from the cooking images, which usually accompanied a recipe. In addition, cooking recipes often omit information that humans unconsciously supplement (common knowledge). Therefore, for a robot to perform the task, the information in the recipe is not sufficient, and the information that the human completes unconsciously has to be supplemented. For example, when a recipe describes "cutting carrots," the actions of "placing the carrots on the cutting board" and "holding the knife" are usually not described in the recipe. The order of execution of these motions are also common sense for a human, however, not for the robot. Therefore, it is necessary to determine the motions that are not explicitly stated in the recipe and their order.

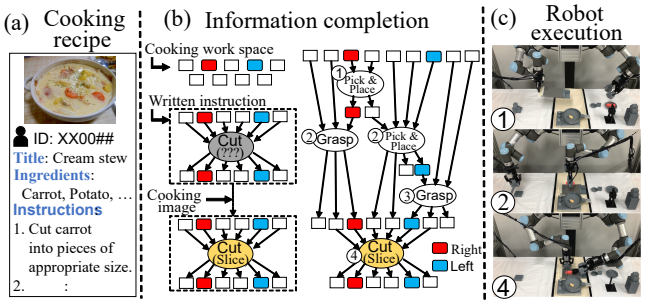

Fig. 1: (a) Cooking recipe consisting of a cooking image and written instructions. (b) First, the information missing in the written instructions is obtained from the cooking image and complemented. Next, information on motions that are not explicitly stated in the recipe are complemented, taking into account the parallel execution of the arms. (c) The robot executes the action based on the graph structure.

To solve these problems, we propose an efficient action planning method for dual-arm robots based on the construction of a graph from a cooking recipe consisting of written instructions and food images. In this method, we first generate a functional unit, which is the basic unit of the graph and consists of object nodes and motion nodes, based on the written instructions. Next, we supplement the missing information in the functional unit taking advantage of the implicit information contained in the recipe's pictures. In particular, for ambiguities in written instructions, where specific information is missing, we create a CNN-based estimation system to obtained this information from the cooking images of the recipe. In this paper, we focus on the case where the information about the process of cutting (style and/or size) is missing or vague. Additionally, for the case where necessary information for the execution of the recipe instructions is missing, we use a graph structure to represent the cooking task, focusing on the relationship between the motion and the object and the object's change of state to deduce the missing information. Finally, our method constructs the graph structure consisting of the whole sequence of a cooking process by connecting the functional units. Furthermore, the proposed graph allows us to do an efficient task planning by appropriately assigning each task to the left or right hands, and deciding whether or not the

[1]Graduate School of Engineering Science, Osaka University, Japan
[2]Industrial Cyber-Physical Systems Research Center, National Institute of Advanced Industrial Science and Technology (AIST), Japan

tasks can be executed in parallel. In summary, the main contributions of this paper are: an efficient task planning method for a dual-arm robot based on a graph; information extraction of a cooking recipe through the combined use of the recipe's instructions and images; completion of missing information by connecting the constructed graph.

This paper is organized as follows. Section II describes the related work. Sections III through VI describe the proposed method. In Section VII, we describe the results of our experiments with a real robot and evaluate the efficiency of the task. Finally, in section VIII, we conclude and discuss future work.

II. RELATED WORKS

To plan the motion of a robot performing a task, hierarchical structured task and motion planning have been proposed by Wolfe et al. [1]; the proposed framework in this paper can also be categorized into task and motion planning. Recently, some extensions of task and motion planning have been proposed [2]–[4]. Dual-arm manipulation has been discussed by several researchers such as [5], [6]. In recent years, motion planning for dual-arm manipulation methods have been proposed such as sequential [7], [8] and coordinated [9] motion planning methods. Stavridis et al. [10]. Paulius et al. [11], [12] proposed a representation framework called FOON (Functional Object-Oriented Network) to model the relationship between actions and objects and the state changes of objects in a task. There have been researches on robotic task motion planning from natural language processing such as [13]–[16].

For planning cooking motions, some researchers realized the motion of a robot used for cooking such as cutting[17], [18], peeling [17], and pouring [19]. Recently, research has been conducted on robotic motion planning from written information such as a recipe. Inagawa et al. [20] generate a cooking behavior from a cooking recipe that can be executed by a robot based on the obtained behavior code. Beetz et al. [13] applied natural language processing to a recipe uploaded in the web and performed action planning of robots. Lisca et al. [21] proposed a framework on conducting chemical experiments from written information. However, in these studies, the information which is not explicitly written is not considered (e.g. there is no missing information in the recipes). Kazhoyan et al. [22] prepared reasoning rules written in action description and used them for supplying the information which is not contained in the written instructions. In addition, large-scale recipe data have been trained with RNN to build inference models [23]. On the other hand, we consider the task efficiency of a dual-arm robot based on the graph structure. In addition, visual information has not been considered in these researches.

III. GRAPH GENERATION

In this section, we describe the elements required for graph generation and how it is generated.

A. Definitions

1) Verb frame: In our method, written instructions are structured in verb frame [23], [24], and a graph is constructed based on this. For example, as shown in Fig.2, for a cutting motion and stir frying motion, the following arguments are defined: Food, Type, Container, Tool, and Equipment. These arguments are defined by the objects and types of motion described in the recipe instructions; they constitute the basic unit of the graph for representing the motions of the recipe instructions as object nodes and motion nodes in the graph. The arguments for cooking containers and tools omitted in the written instructions are initialized with values based on cooking common sense.

Cut	
Food	Carrot
Type	Slice
Container	Cutting board
Tool 1	Knife
Tool 2	Tong

Stir fry	
Food	Carrot
Container	Cooking pan
Tool	Spatula
Strength of fire	High heat
Equipment	Stove

Fig. 2: Verb frame

2) Graph structure: In the proposed graph structure, an object node and a hand node, which represent the state before the motion starts, and an object node and a hand node, which represent the state after the motion, are connected to a motion node. The hand node represents the hand that performs the motion. The basic unit of a graph is called a functional unit. A graph representing a task plan is constructed by connecting multiple functional units. The graph of our method shares similarities with the conventional method proposed by Paulius [11], [12]. However, they differ in that the nodes have the information necessary for motion planning in addition to the information for task planning. Also, hand nodes necessary for planning including the use of the left and right arms are introduced. An example of a graph is shown in Fig.3.

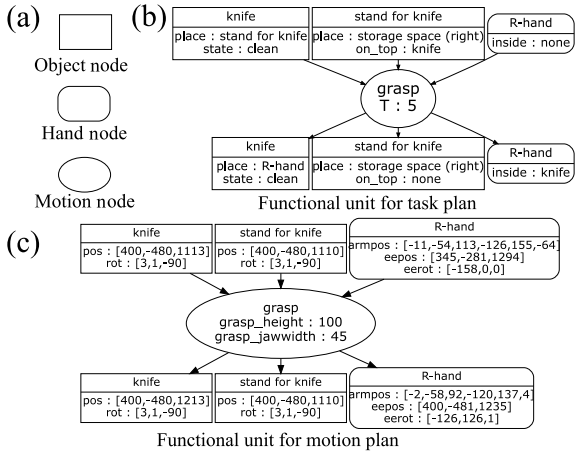

Fig. 3: A functional unit of the proposed graph structure.

3) Functional unit with variables and motion library: A functional unit with variables is a unit in which the type of object cannot be uniquely defined, thus it is defined as a variable. For example, a functional unit with variables for the "pouring operation" defines as variables the "food" that is the object to be poured, the "container A" in which the food is placed before the pouring operation is executed, and the "container B" in which the food is placed after the pouring operation is executed (Fig.4). In this way, it is possible to define the type of object for each motion. In our method, functions that define functional units with variables for each motion are stored in the Motion library. By introducing functional units with variables, it is possible to create functional units according to the situation by assigning values to the variables according to the situation.

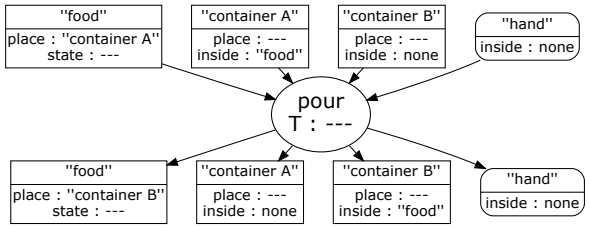

Fig. 4: A functional unit with variables in motion library.

B. Graph generation method

Based on the verb frame and the initial nodes at the beginning of the task, the functional unit of the recipe is generated (Fig.5). First, the initial nodes are generated, of which we focus on the nodes related to the motion and the hand nodes based on the values of the verb frame. These nodes are then given as arguments to the function that defines the functional unit with variables stored in the motion library to generate the functional unit for the motion indicated by the recipe. Attributes that are not defined by the functional unit with variables are inherited from the values of the attributes of the nodes given as arguments.

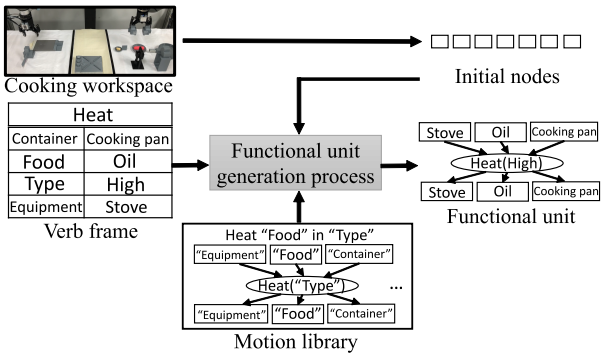

Fig. 5: Workflow for generating the functional unit of a recipe.

IV. Information completion from images

In our method, written instructions are structured by converting them into verb frames. In this process, in many cases, there are arguments whose values are empty due to insufficient written instructions. For example, vague instructions such as "cut the carrots into pieces" or "stir fry the pork in a cooking pan until it changes color" will result in incomplete verb frames such as Fig.6. In order to generate a graph of the recipe, we need to assign values to all arguments of the verb frame. Therefore, in our method, the missing information in the instructions is obtained from the image by using a processing state estimation system. In this paper, we focus on the cases where the cutting motions are missing in the written instructions.

Cut	
Food	Carrot
Type	???
Container	Cutting board
Tool 1	Knife
Tool 2	Tong

Stir fry	
Food	Carrot
Container	Cooking pan
Tool	Spatula
Strength of fire	???
Equipment	Stove

Fig. 6: Verb frame obtained based on insufficient written instructions.

The processing state estimation system is a system that outputs the processing state of an ingredient when given a cooking image and the name of the ingredient of interest as inputs. First, a food detection model for the food of interest is applied to the food image to detect the food of interest. The food detection model is constructed by retraining a deep learning model for object detection with food image data. Next, the processing state is estimated based on the images of the food of interest and the cooking image. Previous research on estimating the state of food [25], [26] have proposed solving the problem as an image classification problem for an input image of one type of food using a Convolutional Neural Network (CNN). In this research, the same approach is taken, but unlike previous research, the image classification problem is solved by providing the image of the food of interest and the cooking image as the input to the CNN, the processing state of the food is estimated based on the output of the CNN (Fig.7).

When multiple ingredients are detected, multiple estimation results must be integrated. To solve this problem,

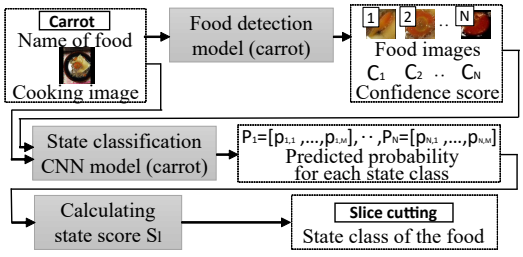

Fig. 7: Estimation system for cutting state.

we calculate a state score. While some food images show significant features of the processing state, others do not because they are hidden. Therefore, for the images with significant features of the processing state to greatly influence in the final estimation result, the estimation is performed by calculating the processing state score S_l using Equation (1).

$$S_l = \sum_{i=1}^{N} p_{i,l} C_i \tag{1}$$

where N is the number of food images obtained and C_k is the confidence score of the k-th ($k = 1,...,N$) image obtained by the detection model. $P_i = [p_{i,1},...,p_{i,M}]$ can be obtained from the output layer of the CNN by using the N food images as input. P_i represents the prediction probability of the M processing state classes. $p_{i,l}$ is the probability of the i-th cooking image of being in the l processing state $l \in \{state_1,...,state_M\}$. The higher the confidence score C_k is, the more the characteristics of the food are expressed, and thus the more the characteristics of the processing state of the food are also expressed. Therefore, by multiplying the confidence score C_k by $p_{k,s}(s = 1,...,M)$ and calculating the sum of N, the images that show significant characteristics of the processing state will greatly influence the final estimation result. The processing state l with the highest score S_l calculated by Equation 1 is determined to be the processing state of the food of interest in the cooking image. An example of state estimation by the processing state estimation system is shown in Fig.8.

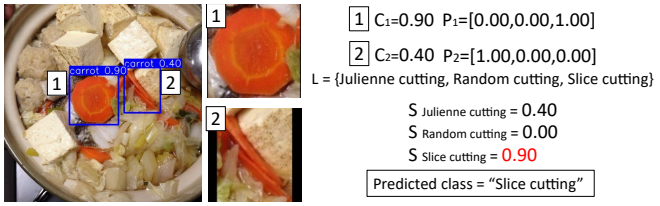

Fig. 8: Example of state estimation (cooking image was obtained from [27]).

V. Implicit information completion

In this section, we describe a method for supplementing information on motion not explicitly mentioned in a recipe (called sub motion) that are necessary to execute the (explicit) motion of the recipe (called main motion).

A. Determination of required motions

By comparing the attribute values of the initial node and the input node of the functional unit of the main motion, the sub motion that needs to be completed is determined. Here, the arm to be used for each motion is also determined based on the value of the attribute of the object to be moved. Task L and Task R are defined as motions that can be performed only by the left or right arm, and Task LR is defined as motions that can be performed by both arms.

In the following, we describe the procedure for determining the sub motions that need to be complemented. First, the input nodes of the functional unit of the main motion are maintained as an input node list. In addition, among the initial nodes that represent the initial state, the nodes that have the same name as the nodes in the input node list and the hand nodes are retained as the output node list. Next, the values of the attributes of the nodes with the same name in the output node list and the input node list are compared based on the rules to obtain the necessary completion sub motions. In the example shown in Fig.9, we obtain the sub motion lists Task L={"Tool B", "Container A"}, Task R={"Tool B"}, and Task LR={}, which are required to execute the motion of pouring potatoes into a bowl. Here, "Tool B" represents the motion of returning the tool to its original location, and "Container A" represents the motion of moving the container to the work space.

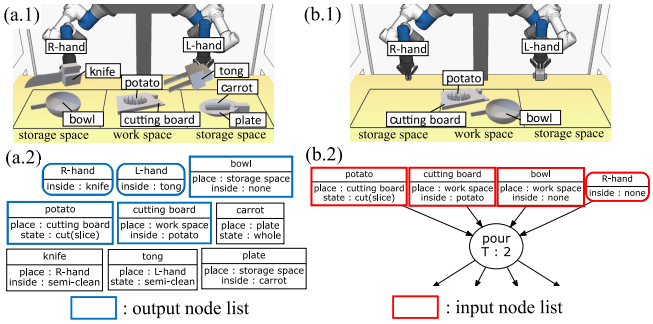

Fig. 9: Example of determining which sub motions need to be completed.

B. Representation of candidate steps in a tree graph

The list of sub motions does not have information about the task procedures of those sub motions. Therefore, a tree graph is constructed to determine the steps of the sub motions. Here, we consider the parallel processing of the arms to improve the efficiency of the task. The nodes of the tree have two types of information. The first is the combination of complementary motions: L_i, R_j represent dual-arm parallel processing, while *None* represents single-arm processing. Here, L_i, R_j represent the execution of the left arm and the right arm, respectively, and *None* means that no processing is performed. The second is whether the operation is executable or not. *True* indicates that the operation is executable, *False* indicates that the motion is not executable, and *None* indicates that the operation is not yet determined. A tree constructed for Task L={"Tool B", "Container A"} and Task R={"Tool B"}. Part of the graph structure is shown in Fig.10. The path from the root node to the leaf nodes represents the work procedure.

C. Determination of the viability of work procedures

The work procedure obtained here does not take into account the possibility of execution. Therefore, it is necessary to determine whether or not the generated work procedure can be executed. For the node that represents the situation at the time the task is executed, the value of the node's attribute

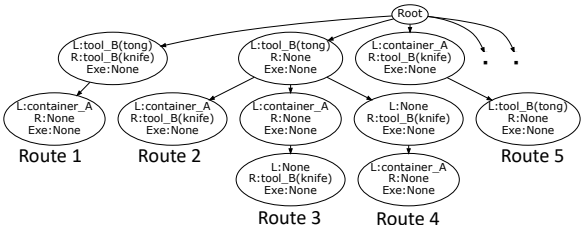

Fig. 10: Example of tree structure.

is judged based on the rule to see if it satisfies the condition, and the value of True or False is given to the executable attribute of the tree graph. In the situation shown in Fig.11, to execute the operation of pouring food into container B, the following two rules are required: Rule 1 is the place attribute of "Container B" equal to work space or stove in work space. Rule 2 is the inside attribute of "Hand" equal to none. In situation (a), this condition is satisfied, so it is feasible and True is given to the attribute values of the tree nodes. On the other hand, in situation (b), this condition is not satisfied, so it is not feasible and False is given to the attribute values of the tree nodes. For all the nodes of the tree graph, the feasibility of execution is determined, and the work procedure for which the attribute value of the leaf node is True is determined to be an executable work procedure.

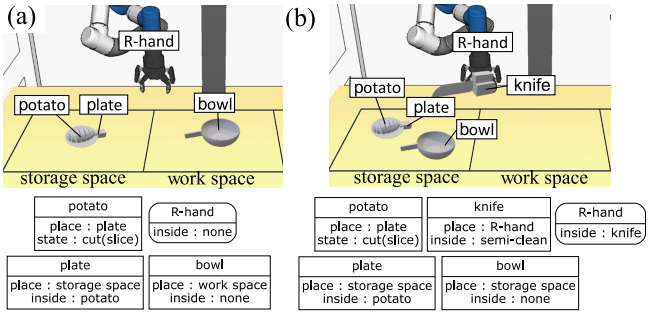

Fig. 11: Example of judging whether or not to execute.

D. Determination of efficient work procedures

Among the paths in the tree that represent feasible work sequences, the path with the smallest length is the most efficient work sequence (Fig.12). Based on this procedure, a work sequence for a dual-arm robot can be obtained by constructing a graph structure. The functional units of the sub motion are generated by giving the current node of interest as the argument of the function that defines the functional unit with variables in the motion library (Fig.13). The executable working procedure generates a functional unit for that motion according to this procedure and combines it with the current node, and the functional units of all sub motions are connected. Finally, it is combined with the functional unit of the recipe operation to obtain a series of work sequences necessary for the execution of the recipe instructions. In our method, the robot executes the task based on the graph of this work sequence.

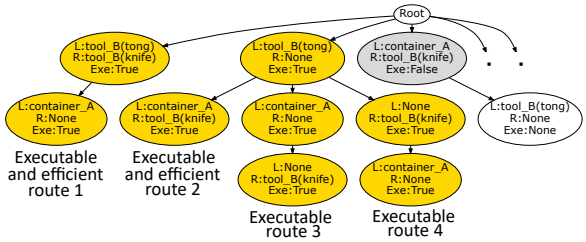

Fig. 12: Tree after the execution order is determined.

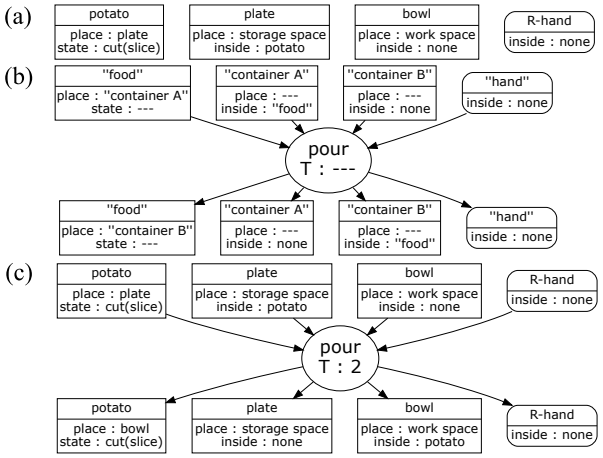

Fig. 13: Generating a functional unit for a sub motion.

VI. Motion Planning Based on Graph Information

In this section, we describe a method for planning the motion of a dual-arm robot based on a graph. By using this method, it is possible to generate some motions based on graphs, such as the pick and place motion of a container and the grasp motion of a tool. First, we describe the necessary predefinition, and then explain the motion planning. In advance, we define several grasping positions relative to the origin of the object (Fig.14(a.1)). The combination of linear trajectory and RRTconnect [28] is defined as the motion generation function for each motion (Fig.14(a.2)). Under these predefinitions, the key posture of the object is first determined based on the values of the input nodes of the motion object in the functional unit. For example, in a pick and place operation, the initial posture and the target posture are determined. The initial posture is obtained from the input node of the target object. In case the target posture can be determined from the position and posture of other related objects, it is obtained from the position and posture information of the input node of the related object. In the example shown in Fig.14, the posture of the final state of the frying pan (target object) is determined by the posture of the stove (related object) (Fig.14(b)). Next, the posture of the arm is obtained by solving the IK for the initial and target

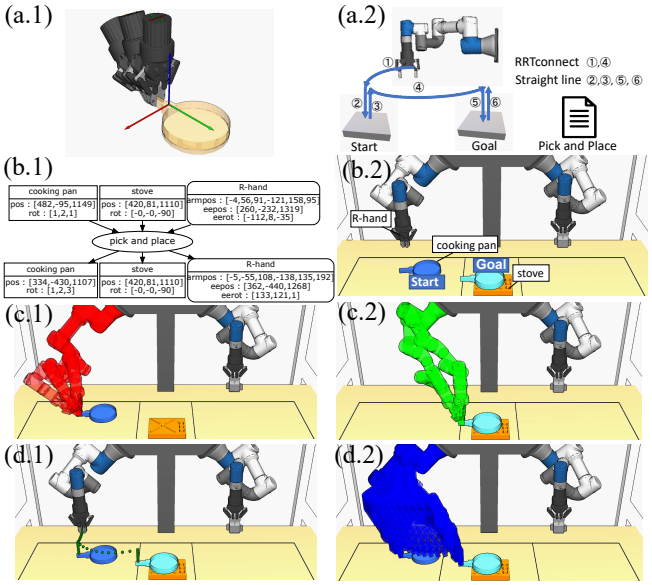

Fig. 14: Procedure for generating motions based on a graph.

states determined based on the predefined grasp posture and node values (Fig.14(c)). Finally, trajectory planning is performed for the obtained arm posture by combining linear trajectory and RRTconnect [28] based on the predefined motion generation function (Fig.14(d)).

VII. Experiment

This section explains the experiment conducted to show the effectiveness of the proposed approach.

A. Experimental environment

The experimental setup and the experimental environment are shown in Fig.15. In the recipe, at the written instructions 1, 2, and 3, it is necessary to complete motions that are not explicitly stated in the recipe. In addition, instruction 2 requires the completion of the processing method from the cooking image. In this experiment, two robotic arms are used, and a work space and two storage spaces (right and left) are prepared. The objects are arranged as shown in Fig.15, and the initial position and orientation of each object are known. If there is an object in the work space, both arms can be used. If there is an object in the storage space, only the arm near the storage space can be used. The robot arms are two UR3, the hand is a 2-Finger 85mm Gripper, the ingredients are food samples, and objects such as cooking tools and containers are designed for the robot.

B. Completing information from images

In this experiment, we built an estimation system using images from the Rakuten recipe in the Rakuten dataset [27] as training data. Initially, the written instructions are structured in verb frames, and the results obtained are shown in Fig.16. From this result, we can see that the processing method of cutting needs to be obtained from the image and complemented. We applied the estimation system to the

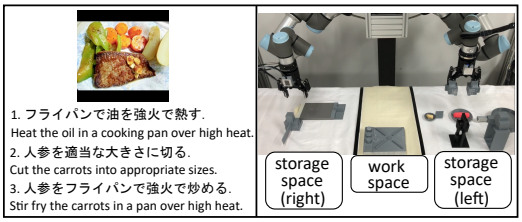

Fig. 15: Cooking recipe and environment for the experiment (Cooking image obtained from [27]).

cooking image of the recipe, the results of the estimation (food detection and state classification) are shown in Fig.17. In addition, the processing state score S_l for each state was calculated as $S_{julienne}$=0.00, S_{random} =0.25, and S_{slice}=2.90, respectively.The result is "slice cutting". This completes the missing information in the written instructions.

Heat	
Food	Oil
Heat type	High heat
Container	Cooking pan
Equipment	Stove

Cut	
Food	Carrot
Type	???
Container	Cutting board
Tool 1	Knife
Tool 2	Tong

Stir fry	
Food	Carrot
Container	Cooking pan
Tool	Spatula
Strength of fire	High heat
Equipment	Stove

Fig. 16: Verb frames obtained from the written instructions in the recipe.

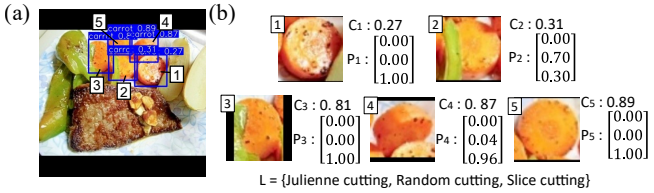

Fig. 17: Estimated results from cooking image.

C. Completing implicit information

Next, we complement the information that is not explicitly indicated in the recipe. First, we generate the functional units of the explicit indications of the recipe based on the verb frame shown in Fig.16. Next, we compare the attribute's values of the current node and the input node of the instruction of the recipe to determine the required operations. The current node is the node of each object at the start of the work shown in Fig.15 for instruction 1, and at the end of the previous operation for instructions 2 and 3. From this, it is determined that three sub motions are required for Heat operation, four for Cut operation, and four for Stir fry operation, as shown in Table I.

Next, the results for the tree graphs constructed to determine the steps for the obtained sub motions are shown in Table II. Among the feasible paths, the minimum path length was found to be 3 for Heat operation, 3 for Cut operation, and 2 for Stir fry operation. This result indicates

TABLE I: Sub motion list for each instruction.

	TaskR	TaskL	TaskRL
Heat		Container B[1] Food B[2]	Equipment A[3]
Cut	Tool A[4] Container A[5]	Tool A[4] Food A[6]	
Stir fry	Tool B[7]	Tool B[7] Tool A[4]	Food B[2]

[1] Pick and place : Moving the container to the top of the stove.
[2] Pour : Moving food that cannot be grasped to the other container.
[3] Turn on : Turning the stove on and ignite it.
[4] Grasp : Grasping the cooking tool.
[5] Pick and place : Moving the container to the work space.
[6] Pick and place : Moving food that can be grasped to the other container.
[7] Release : Returning the cooking tool to the stand for tool.

that the efficiency of the task is improved when using dual-arm parallel executions: once for Cut operation and twice for Stir fry operation. The obtained tree graph is partially shown in Fig.18. Finally, Fig.19 shows the graph representing the efficiently determined task plan.

TABLE II: Results of the tree graph structure.

	Number of routes		Length of routes		
	All	Executable	Min	Max	Ave
Heat	16	2	3	3	3.00
Cut	52	5	3	4	3.60
Stir fry	94	20	2	4	3.45

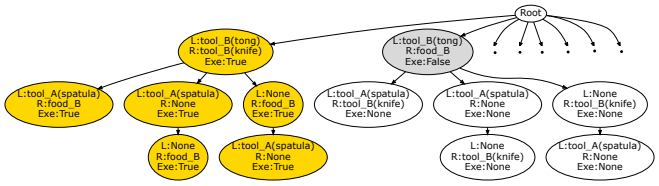

Fig. 18: Part of the tree structure for the sub motions of Stir fry operation.

D. Experiment with a robot

Finally, we conducted an experiment in which the dual-arm robot cooked in a real environment based on the obtained task and motion plan. As a result, the dual-arm robot succeeded in executing the cooking task instructed in the recipe by supplementing the motions that were not specified in the recipe before and after the instructions in the recipe. The robot experiment is shown in the Fig.20.

VIII. Conclusion

In this paper, we proposed a method to generate the motions of a dual-arm robot by supplementing the information necessary to carry out recipe instructions. In particular, when the instructions for the cutting motion are ambiguous, the information can be supplemented from the images in the recipe. In addition, it can complement the information about the preparatory motions of a dish and its working procedures that are not explicitly stated in the recipe. Furthermore, by taking into account the parallel processing of the two arms, an efficient work procedure can be planned. In the future, we would like to consider the case where two robots are used. Also, a future task would be to consider a system where a mobile robot is used and for collaborative work with humans. We would like to expand our method to include information for various cooking operations other than cutting, such as stir frying, baking, and mixing. Furthermore, our method can also be applied to other tasks that use instructions composed of images and text, such as the assembly of industrial products.

More figures and videos of this research can be seen at `www.roboticmanipulation.org/res/cook`

IX. Acknowledgement

This works is supported by NEDO (Project ID:P20006). We used "Rakuten Dataset" (`https://rit.rakuten.com/data_release/`) provided by Rakuten Group, Inc. via IDR Dataset Service of National Institute of Informatics.

References

[1] J. Wolfe, B. Marthi, and S. Russell, "Combined task and motion planning for mobile manipulation," in *Proc. of Int. Conf. on Automated Planning and Scheduling*, 2010.
[2] C. Garrett, T. Lozano-Pérez, and L. Kaelbling, "Ffrob: an efficient heuristic for task and motion planning," *Algorithmic Foundations of Robotics XI*, p. 179–195, 2015.
[3] B. Woosley and P. Dasgupta, "Integrated real-time task and motion planning for multiple robots under path and communication uncertainties," *Robotica 36(3)*, p. 353, 2018.
[4] W. Wan and K. Harada, "Developing and comparing single-arm and dual-arm regrasp," *IEEE Robot. Autom. Lett., vol. 1, no. 1*, pp. 243––250, 2016.
[5] B. Siciliano, "Advanced bimanual manipulation: results from the dexmart project," *Springer Science & Business Media*, 2012.
[6] J. Krüger, G. Schreck, and D. Surdilovic, "Dual arm robot for flexible and cooperative assembly," *CIRP Ann. 60(1)*, 2011.
[7] B. Cohen, M. Phillips, and M. Likhachev, "Planning single-arm manipulations with n-arm robots," in *Proc. of Annual Symposium on Combinatorial Search*, 2015.
[8] J. Kurosu, A. Yorozu, and M. Takahashi, "Simultaneous dual-arm motion planning for minimizing operation time," *Appl. Sci. 7(12)*, p. 2110, 2017.
[9] I. Ramirez-Alpizar, K. Harada, and E. Yoshida, "Human-based framework for the assembly of elastic objects by a dual-arm robot," *Robomech. J. 4(1)*, p. 20, 2017.
[10] S. Stavridis and Z. Doulgeri, "Bimanual assembly of two parts with relative motion generation and task related optimization," in *Proc. of IEEE/RSJ Int. Conf. on Intelligent Robots and Systems*, 2018, p. 7131–7136.
[11] D. Paulius *et al.*, "Functional object-oriented network for manipulation learning," *Proc. of IEEE/RSJ Int. Conf. on Intelligent Robots and Systems*, 2016.
[12] ——, "Task planning with a weighted functional object-oriented network," 2021.
[13] M. Beetz *et al.*, "Robotic roommates making pancakes," in *Proc. of IEEE-RAS Int. Conf. on Humanoid Robots*, 2011, pp. 529–536.
[14] R. Paul *et al.*, "Temporal grounding graphs for language understanding with accrued visual-linguistic context," in *Int. Joint Conf. on Artificial Intelligence*, July 2017.
[15] T. Howard *et al.*, "A natural language planner interface for mobile manipulators," in *Proc. of IEEE Int. Conf. on Robotics and Automation*, 2014.
[16] S. Tellex *et al.*, "Robots that use language," *Annual Review of Control, Robotics, and Autonomous Systems*, no. 3, pp. 1–35, 2020.

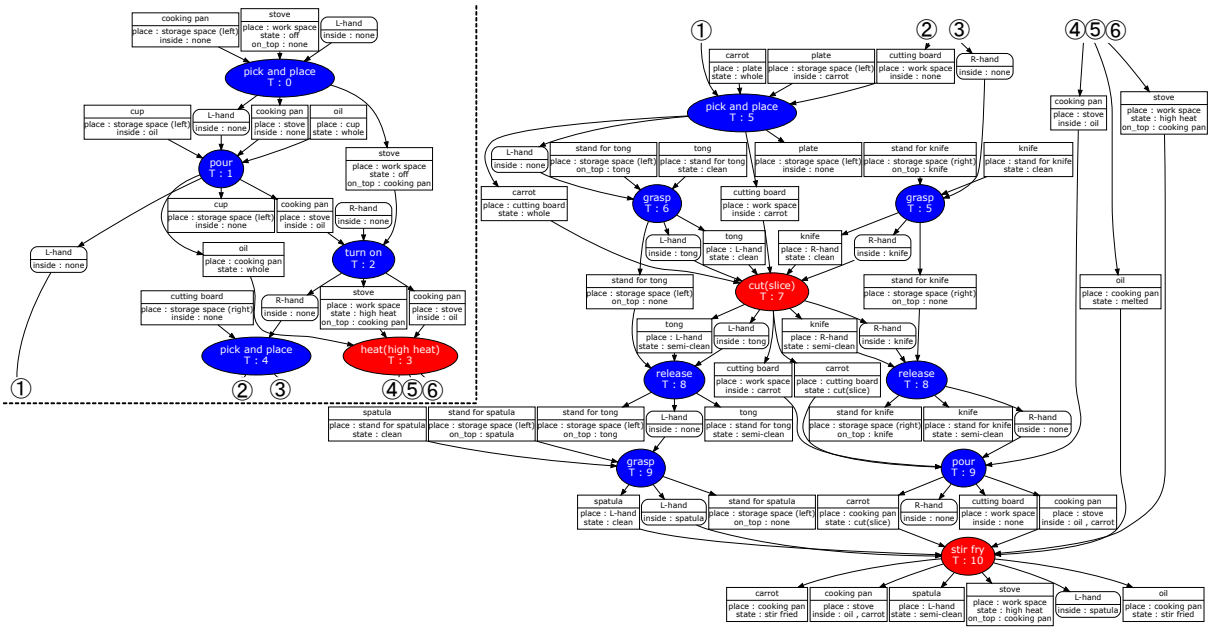

Fig. 19: Graph representing the work sequence. Red motion nodes (motions described in the recipe), blue motion nodes (motions not described in the recipe).

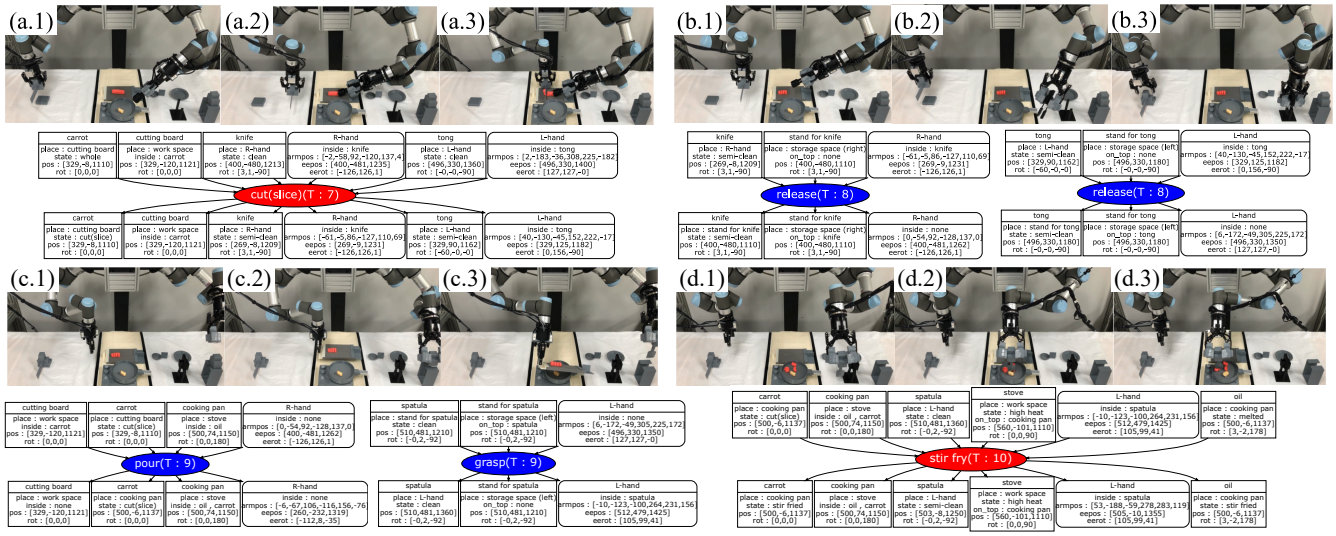

Fig. 20: Robot execution of a task based on a graph.

[17] K. Yamazaki *et al.*, "Recognition and manipulation integration for a daily assistive robot working on kitchen environments," in *Proc. of IEEE Int. Conf. on Robotics and Biomimetics*, 2010, pp. 196–201.

[18] X. Mu, Y. Xue, and Y.-B. Jia, "Robotic cutting: Mechanics and control of knife motion," in *Proc. of IEEE Int. Conf. on Robotics and Automation*, 2019, pp. 3066–3072.

[19] A. Yamaguchi and C. G. Atkeson, "Stereo vision of liquid and particle flow for robot pouring," in *Proc. of IEEE-RAS Int. Conf. on Humanoid Robots*, 2016, pp. 1173–1180.

[20] M. Inagawa, T. Takei, and E. Imanishi, "Japanese recipe interpretation for motion process generation of cooking robot," in *Proc. of IEEE/SICE Int. Symposium on System Integration*.

[21] G. Lisca, D. Nyga, F. Bálint-Benczédi, H. Langer, and M. Beetz, "Towards robots conducting chemical experiments," *Proc. of IEEE/RSJ Int. Conf. on Intelligent Robots and Systems*, pp. 5202–5208, 2015.

[22] G. Kazhoyan and M. Beetz, "Programming robotic agents with action descriptions," in *Proc. of IEEE/RSJ Int. Conf. on Intelligent Robots and Systems*, 2017.

[23] H. Chen *et al.*, "Enabling robots to understand incomplete natural language instructions using commonsense reasoning," in *Proc. of IEEE Int. Conf. on Robotics and Automation*, 2020, pp. 1963–1969.

[24] D. Das *et al.*, "Frame-semantic parsing," *Computational linguistics*, vol. 40, no. 1, pp. 9–56, 2014.

[25] R. Paul, "Classifying cooking object's state using a tuned vgg convolutional neural network," *arXiv preprint arXiv:1805.09391*, 2018.

[26] A. B. Jelodar, M. S. Salekin, and Y. Sun, "Identifying object states in cooking-related images," *arXiv preprint arXiv:1805.06956*, 2018.

[27] "Rakuten Group, Inc.: Rakuten Recipe data. Informatics Research Data Repository, National Institute of Informatics. (dataset). ," https://doi.org/10.32130/idr.2.4.

[28] J. J. Kuffner and S. M. LaValle, "Rrt-connect: An efficient approach to single-query path planning," in *Proc. of IEEE Int. Conf. on Robotics and Automation*, vol. 2, 2000, pp. 995–1001.