Article

Graph Based Framework on Bimanual Manipulation Planning from Cooking Recipe

Kota Takata[1], Takuya Kiyokawa[1] Natsuki Yamanobe[2] Ixchel G. Ramirez-Alpizar[2] Weiwei Wan[1] and Kensuke Harada[12]

[1] Graduate School of Engineering Science, Osaka University, Japan; {kiyokawa@hlab.,wan@,harada@}sys.es.osaka-u.ac.jp
[2] ICPS Research Center, National Inst. Advanced Industrial Science and Technology, Japan; {n-yamanobe}@aist.go.jp
* Correspondence: harada@sys.es.osaka-u.ac.jp

Abstract: It is difficult to effectively operate a dual-arm robot using only the information written in a cooking recipe. To cope with this problem, this paper proposes a graph based approach on bimanual cooking motion planning from a cooking recipe. In our approach, we first decompose the cooking recipe into graph elements. Then, we try to connect the graph elements taking into account the attributes of the input/output nodes. If two graph elements cannot be connected each other, we search for a graph element that can be inserted between them from the database of graph elements. Since the constructed graph includes the whole sequence of the robot's motions to perform the cooking task, we can generate a task sequence of a dual-arm manipulator simultaneously performing two different tasks by using two arms. Through experimental study, we show that it is possible to generate robot motions from a cooking recipe and perform the cooking motions while simultaneously moving the left and right arms.

Keywords: Robot, Motion planning, Cooking, Task, Manipulation Graph

1. Introduction

One of the ultimate goals of robotics research is to use robots to replace a human performing everyday household chores. This research focuses on cooking as one of such household chores performed by humans. If a robot is to cook a food, it must understand cooking recipes written in a way that humans can understand, and must process ingredients appropriately based on these recipes. In order to deal with this problem, we propose a graph-based approach to convert a cooking recipe into an executable format and plan the robot's motion based on the recipe.

The information described in the recipe is not sufficient for a robot to perform a cooking task. To control a robot based on a cooking recipe, it becomes necessary for a robot to complement the actions which are not explicitly described in the recipe, like a human performs unconsciously. For example, if "cut carrots" is described in a recipe we need to execute the action of "placing the carrots on the cutting board" beforehand. In addition, we need the action of "grasping the knife" before the cutting operation. However, these instructions are usually not described in the cooking recipe.

In this study, we propose a motion planner for the robotic cooking tasks by using the information obtained from a cooking recipe where the planner can automatically determine objects and motions that are not explicitly described in the cooking recipe (Fig. 1). Our method represents a cooking task using a graph structure that can express the relationship among actions, tools, hands and objects with their state changes. Before a robot motion is planned, the elements of the network structure is stored in a database named the "action library" where each graph element includes a set of arguments expressing the status of a task. From the arguments of the input node, we can detect the lacking information

Preprints (www.preprints.org) | NOT PEER-REVIEWED | Posted: 17 October 2022 doi:10.20944/preprints202210.0237.v1

of the cooking recipe. The lacking information is complemented by substituting the graph element obtained by searching the database. In addition, by introducing the hand node, our proposed planner automatically determines the bimanual coordination by assigning the extracted motion from a cooking recipe to one of the hands.

In this paper, after explaining the related works in Section 2, we explain each element of the proposed network structure in Section 3. We explain how to compose the network structure from a cooking recipe in Section 4. We show the experiment's results in Section 5.

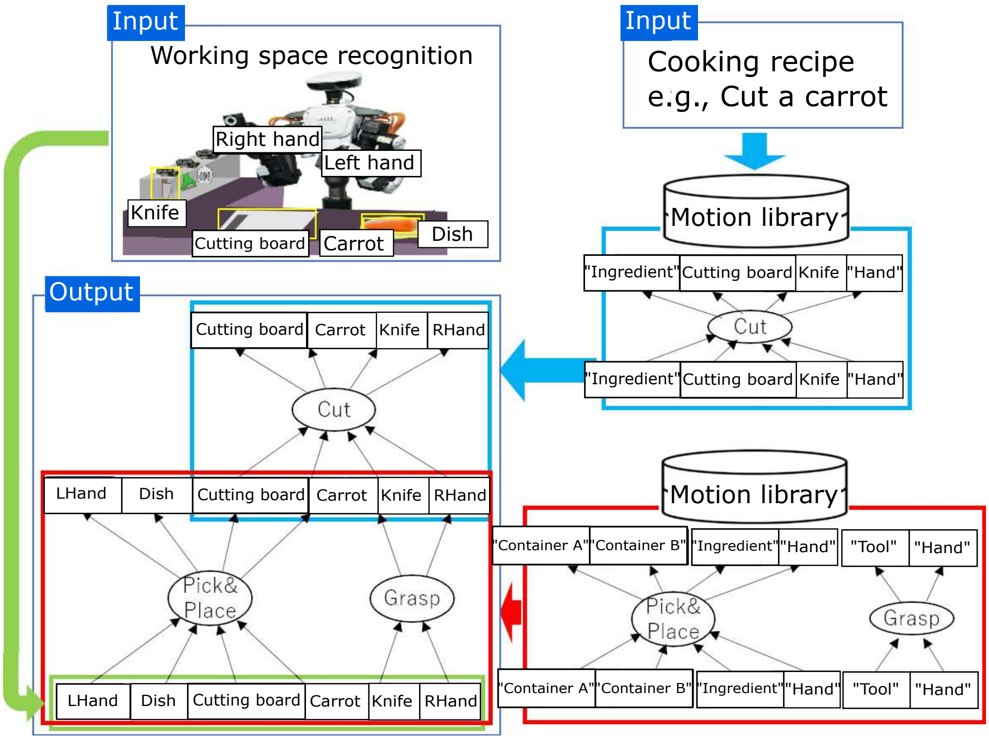

Figure 1. Overview of the proposed framework

2. Related Works

To plan the motion of a robot performing a task, Wolfe et al. [1] proposed the hierarchical structured planning problems including the task and the motion layers where our proposed framework can also be categorized into the task and motion planning problems. For the task planning, a series of logic operations has been proposed such as [2] and [3]. For the motion planning, the motion of a robot is planned by using several methods, such as the topological optimization [4,5] and graph search [6]. Recently, some extensions of the task and motion planning have been proposed, such as partially observable environment [7], symbolic method [8], and real-time planning [9], and regrasp planning[10][11].

In addition, dual-arm manipulation has been receiving attention by several researchers such as [12,13]. In recent years, motion planning for dual-arm manipulation methods have been proposed, such as sequential planning [14,15], coordinated planning [16], assembly planning [17] and assembly planning with regrasp [18].

In addition, some researchers focused on the robotic cooking task. Yamazaki et al.[19] performed robotic cooking operations like cut and peel by recognizing the foods and the cutting board. Mu et al.[20] analyzed the mechanics of cutting operation for cooking ingredients. Yamaguchi et al. [21] realized a robot to learn a pouring operation.

Recently, some researches have been conducted on robotic motion planning from linguistic information such as recipe. Inagawa et al. ([22]) propose a method to analyze a cooking recipe and generate a cooking behavior that can be executed by a robot based on the obtained behavior code. Beetz et al. [23] applied the natural language process-

Preprints (www.preprints.org) | NOT PEER-REVIEWED | Posted: 17 October 2022 doi:10.20944/preprints202210.0237.v1

ing to recipes uploaded in the web and performed the action planning of robots. Lisca et al. [24] proposed a framework on conducting chemical experiments from linguistic information. However, In these studies, the information which is not explicitly described in the linguistic information cannot be used to generate the robot motion. Kazhoyan et al. [25], however, prepared the reasoning rules written in action description and used it for supplying the information which is not contained in the language. In addition, large-scale recipe data have been trained with RNN to build inference models[26]. Paulius et al. [27][28][29] 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. On the other hand, we have proposed the framework on task and motion planning of dual-arm manipulator from a cooking recipe composed of food image and cooking instructions. In our previous research [30], we have explained the entire structure of robot motion planning and detailed the recognition of ingredients from the food image and coordination of two arms. This paper focuses on the motion planning framework based on a graph structure, which solves the mentioned problem, to plan the robot's cooking task from a cooking recipe.

3. Proposed Method

In this paper, we represent a cooking task using a graph structure including the actions associated with objects and the state changes of objects. This graph structure enables us to identify actions and objects that are necessary for executing a cooking task but are not described in a cooking recipe. Our proposed planner generates a sequence of cooking task motions while complementing it with actions not explicitly described in a cooking recipe. The graph structure used in this study extends the FOON (Functional Object-Oriented Network) [27][28] to handle tools like cooking utensils and to consider the robot states. Section 3.1describes the network structure used in this study, and section 3.2describes the procedure for creating a graph based on a cooking recipe.

3.1. Graph Structure

The graph structure used in this study is a directed graph consisting of edges and three types of nodes, i.e., the object, hand and motion nodes. The nodes have attributes and can represent their state change in the manipulation task. This network structure is created by connecting multiple functional units where the functional unit denotes the smallest unit of the graph structure representing the information contained in a single action. A functional unit also consists of edges and three types of nodes. In the following, we first describe the details of the nodes and functional units included in the graph structure.

3.1.1. Object Node

An object node represents the object in action. The types of objects in cooking include Ingredients, Seasonings, Tools, and Containers. An object node includes attributes of Type, Name, Place and State (see Table 1). If two instances of the same object type have 2 different attributes, they can be treated as different object nodes. This makes it possible to determine the action needed for executing the task. For example, as shown in Figure 2, it is possible to distinguish between "a knife that exists in a storage location" and "a knife that is being grasped by the robot". The knife needed for the cutting operation can be specified as the "grasped knife" on the right side of Figure 2. If the knife is in the storage area, we can determine that the grasping action is required before the cutting action.

Knife	
Storage space	Clean

Knife	
Hand	Clean

Figure 2. An example where same objects are expressed in different nodes

 doi:10.20944/preprints202210.0237.v1

Table 1. Examples of objects' name and attributes

Type	Name	Attribute1(Place)	Attribute2(State)
Food	Potato, Beef	Storage space, Cutting board	Whole, Cut, Chopped
Seasoning	Sugar, Salt, Water	Storage space, Cup, Bowl	Whole, Mixed
Tool	Knife, Spatula, Ladle	Storage space, Hand	Clean, Dirty
Container	Cutting board, Bowl	Storage space, Work space	ingredient inside(Potato)

3.1.2. Hand Node

A hand node is assigned for each robotic hand used in the cooking task. It includes the usage status of the robot hand as an attribute. Specifically, it has the names of the objects grasped by the hand. By having these attributes, it is possible to determine the task sequence according to the number of robot arms used. For example, we can plan the motion of a robot stir frying a food with simultaneously holding spatula and cooking pan with the right and left hands.

3.1.3. Motion Node

A motion node represents an action. In a cooking task, there are two types of motions: main motion and sub-motion. The main motion includes cooking motions like cut, pour and boil. These actions are usually described in a cooking recipe. The sub-motion is used to prepare for a cooking motion, e.g., pick & place, grasp and release. This node does not have any attributes.

Table 2. Type of motion node

Motion	Name
Main motion	Cut(half)
	Poar
	Boil
Sub-motion	Pick & Place
	Grasp
	Release

3.1.4. Functional unit

A functional unit consists of a motion node, hand nodes and object nodes where object and hand nodes work as inputs and outputs for a motion node. The attributes of the input object and hand nodes are updated according to the changes caused by the execution of the cooking task. The functional unit represents the objects needed for executing the motion and its changes during the motion execution. An example of the operation "cut potato in half" is shown in Figure 3. To execute this operation, the cutting board, the potato, and the knife need to be in the input object and hand nodes. We also show that, according to the motion execution, the state of the potato and the knife changes as shown in the yellow frame in Figure 3.

3.1.5. Discussion

Different from FOON, our proposed network structure can take into account the tools like cooking utensils and possible order of actions by considering the attributes of a hand node. For example, consider the case where a knife or a chopping board is in a storage location when the robot performs a cutting operation. In such a case, our method can include actions such as "grasping the knife" and "preparing the cutting board" in the task planning (Figure 4).

In addition, the possible execution order is determined based on the status of the robot hand. In the example shown in Fig. 4, there are multiple execution sequences,

Preprints (www.preprints.org) | NOT PEER-REVIEWED | Posted: 17 October 2022 doi:10.20944/preprints202210.0237.v1

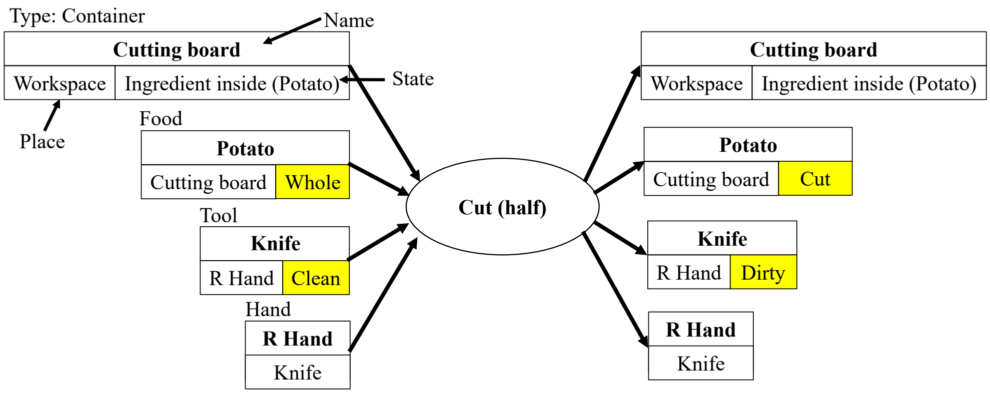

Figure 3. An example of functional unit of "Cut potato in half"

but by taking into account the status of the robot hands, we can determine the possible execution order, i.e., grasp of a knife should be done after the pick and place of potato, this is because the hand should not grasp anything before executing the pick and place.

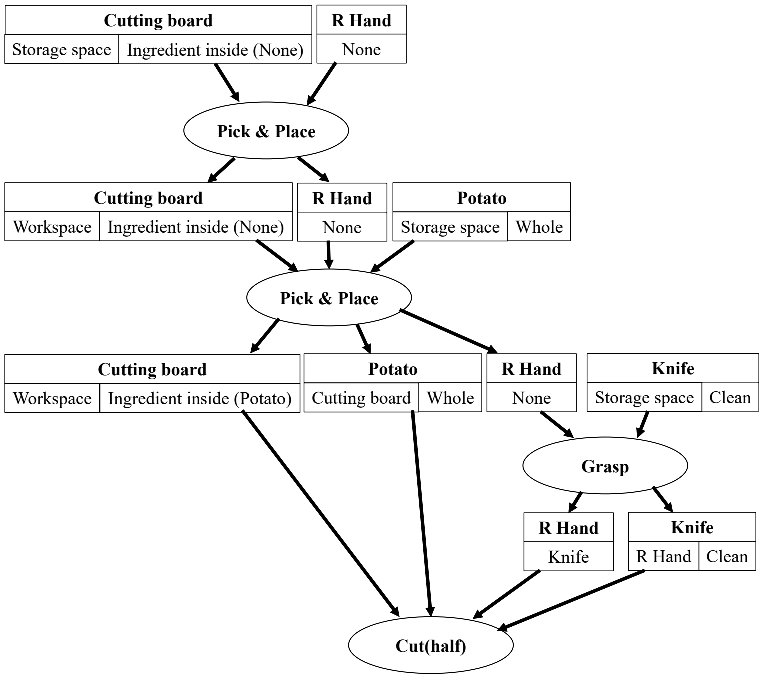

Figure 4. An example of task planning including the tool use.

3.2. *Graph Construction*

By using the graph structure described in section 3.1, we can identify the missing actions and objects in a cooking recipe. We store the functional units on basic operations used for cooking in the database. If missing actions or objects are identified, we search in the database and make adequate modifications to a functional unit stored in the database. We use this modified functional unit to complement the missing actions and objects. In this section, we first describe the motion database and then explain the procedure for creating a network with complementing the missing actions and objects.

3.2.1. Motion Database

In this subsection, we explain the motion database (DB) used in this study. In each DB, functional units in which the parameters have not been set yet are stored. We call such

Preprints (www.preprints.org) | NOT PEER-REVIEWED | Posted: 17 October 2022 doi:10.20944/preprints202210.0237.v1

functional unit as the functional unit with arguments. By setting adequate parameters to the arguments, an appropriate functional unit can be created.

In this study, we use two types of motion DB for main motion and sub-motion, which will be referred to as the main motion DB and the sub-motion DB, respectively. The main motion DB stores the functional units with arguments corresponding to the main motion, i.e., cooking motion explicitly described in a cooking recipe. The type of object required, also included in a functional unit, is determined for each main motion. For example, the objects required for the "pouring motion" are the "ingredient" to be poured, the "container" in which the ingredient is placed before pouring, and the "container" to which the ingredient is poured. As shown in this example, even if an object cannot be uniquely defined, the type of object can be defined for each operation, and objects that cannot be uniquely defined can be expressed as functional units with arguments (Figure 5). Such functional units with arguments are constructed for each main motion and stored in the main motion DB.

In the sub-motion DB, functional units for each sub-motion, i.e., the motion which is not described in the cooking recipe, are stored. Sub-motions are related to the handling of objects, and include "Pick & Place," "Grasp," "Release "Pick & Place", "Grasp", "Release", etc. The types of objects required for these motions can also be defined, and they can be expressed as functional units with arguments (Figure 6). Such functional units with arguments are constructed for each sub-motion and stored in the sub-motion DB.

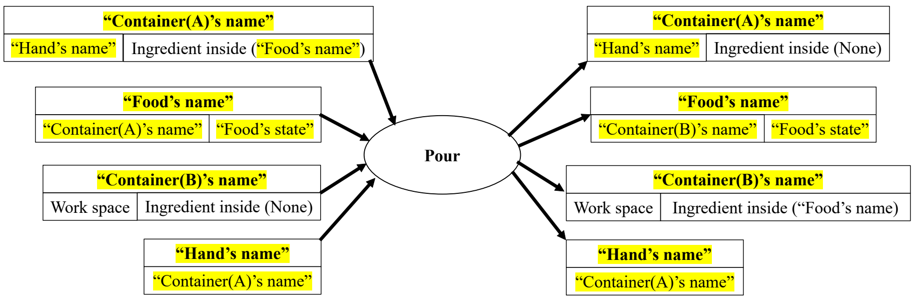

Figure 5. An example of the functional unit with arguments stored in the main motion DB

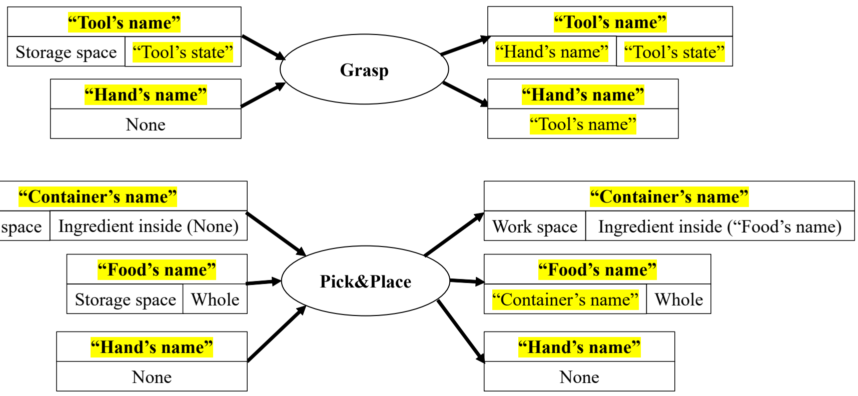

Figure 6. An example of the functional unit with arguments stored in the sub-motion DB

3.2.2. Graph Connection

In this subsection, we explain the procedure for creating a whole graph structure to achieve the cooking task from the recipe. Figures 7 and 8 show the outline of the creation procedure by using the motion DB.

The input is a sentence described in a cooking recipe. First, a sentence included in the cooking recipe is parsed to obtain the relation between the object and the motion (Figure

Preprints (www.preprints.org) | NOT PEER-REVIEWED | Posted: 17 October 2022 doi:10.20944/preprints202210.0237.v1

7①), where the i-th motion obtained here is denoted as the main motion (i) ($i = 1, 2, ..., N$). Then, we search for the functional unit with arguments stored in the main motion DB corresponding to the main motion (i), and substitute the extracted object and motion from the recipe to the arguments, and create the functional unit (Figure 7②). The next step is to recognize the object initially placed in the robot's working environment. We obtain the name, state, and place of the object, and create the object node. Then, the functional unit is created in which the object node obtained here becomes the output node (the input and motion nodes are assumed to be empty nodes) (Figure 7③). Finally, the functional units obtained here are connected based on the merging algorithm to complete the network (Figure 7④). The details of this merging algorithm are described in the following pages.

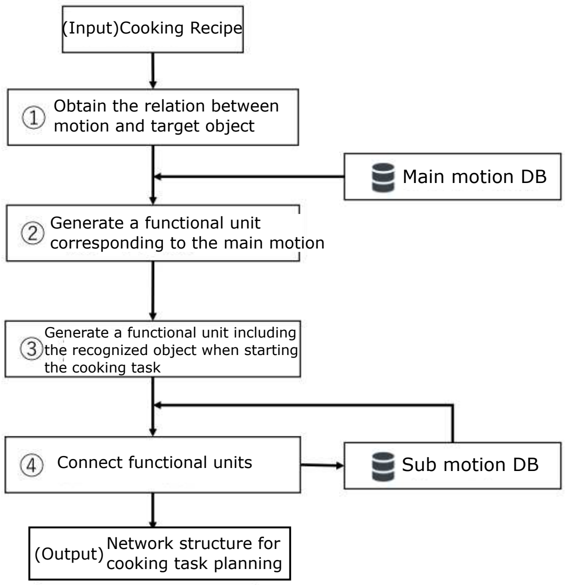

Figure 7. Flowchart of Graph construction from motion DB

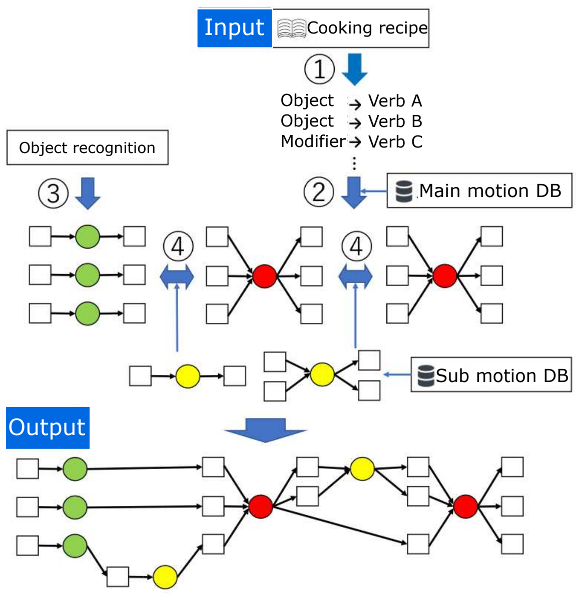

Figure 8. Overview of graph construction by using the functional units

Preprints (www.preprints.org) | NOT PEER-REVIEWED | Posted: 17 October 2022 doi:10.20944/preprints202210.0237.v1

The flowchart of the network connection is shown in Figure 9. We define the input/output node lists of the network structure of combined multiple functional units. In the connection algorithm, we first create a list of input nodes (Figure 9(A)) where the elements of this list are the input of the functional unit for the main motion i (abbreviated as the functional unit (i)) ($i = 1, \cdots, N$). Then, we create a list of output nodes (Figure 9(B)) where the elements of this list are the output nodes of a functional unit to be connected to the functional unit (i). For example, when a robot starts a cooking task, the elements of the output node list are the output of a functional unit where its input includes the objects' initial configuration.

Next, the elements of the input/output node lists are compared with the input/output of functional units to see if they have the same names and attributes (Figure 9(C)). For example, if the elements of the output node list match the input of the functional unit (j) as shown in Figure 10, we connect the functional unit (j) to the network structure by deleting these elements from the output node list and appending the output of the functional unit (j) to the output node list (Figure 9(C-1)).

Furthermore, if the elements of the output node list does not match the input of any functional units for the main motion (abbreviated as the main-functional units) as shown in Figure 11, we create a new functional unit for the sub motion (abbreviated as the sub-functional unit) and try to connect it to the network structure (Figure 9(C-2)). We substitute the elements of the output node list to the arguments of a sub-functional unit stored in the sub-motion DB. Then, the sub-functional unit is connected to the network structure by deleting the elements of the output node list and appending the output node of the sub-functional unit to the output node list (Figure 9(B)). The functional units are iteratively connected by comparing the elements of the input/output node lists with the input/output of functional units (Figure 9(C)). At each connection, we check if the input node list is empty (Figure 9(D)). If the input node list is empty, i.e., all the input nodes of the main motion are combined with functional units (Figure 9(D)), the actions necessary to execute the main motion are added and the main motion is ready to be executed. Finally, the order to execute each motion is determined, taking into account the number of available arms (Figure 9(E)). Here, an executable sequence will not exist if the arm is grasping an object and cannot perform any other action. If there is no executable sequence, we create and combine sub-functional units that eliminate such cases to obtain an executable task planning (Figure 9(E-1)).

After the above process is performed for the main motion (i) ($i = 1, 2, \cdots, N$), we obtain a whole network structure that represents the cooking task.

Preprints (www.preprints.org) | NOT PEER-REVIEWED | Posted: 17 October 2022 doi:10.20944/preprints202210.0237.v1

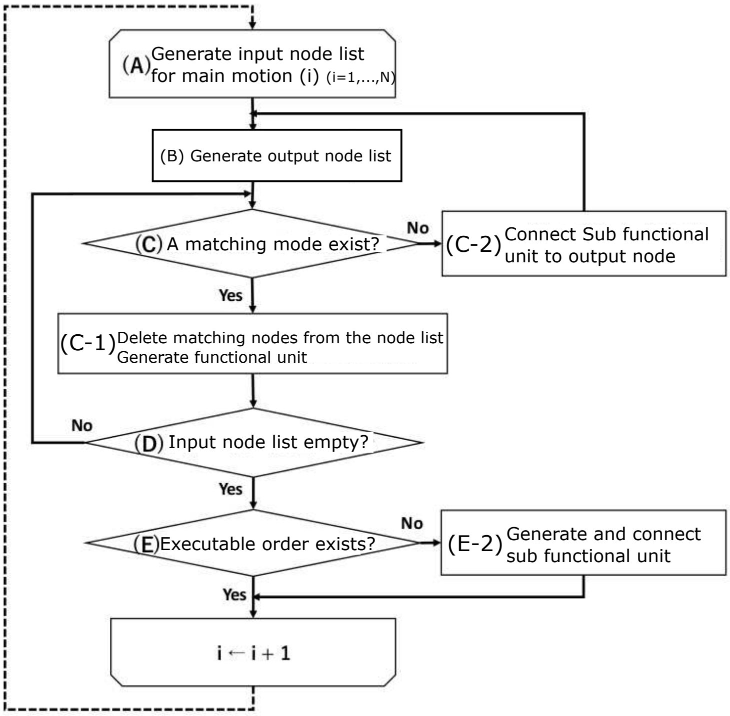

Figure 9. Flowchart of combine algorithm

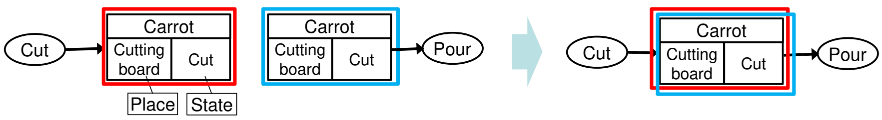

Figure 10. Connection of a functional unit when it has the same name and attributes in the input/output node list

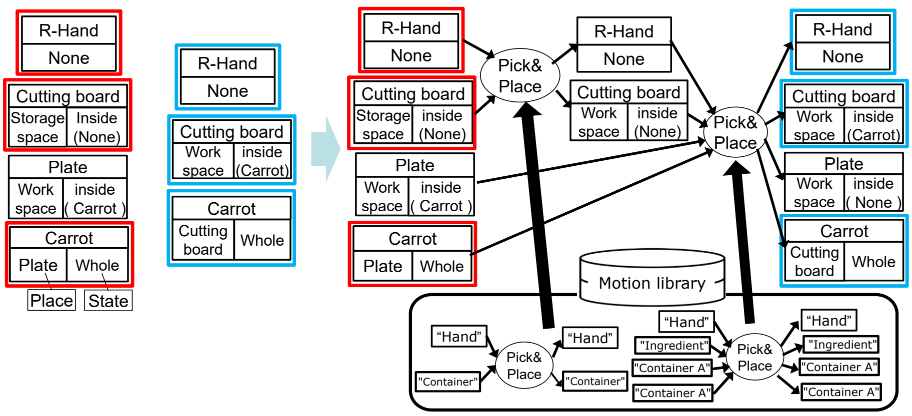

Figure 11. Connection of a functional unit when it does not have the same name and attributes in the input/output node list

3.3. Bimanual Task Planning

We introduce a dual-arm robot to efficiently perform the cooking task. Introduction of the hand nodes to a functional unit enables us to consider the number of hands used for each cooking action and to plan the cooking task considering multiple hands performing

Preprints (www.preprints.org) | NOT PEER-REVIEWED | Posted: 17 October 2022 doi:10.20944/preprints202210.0237.v1

multiple tasks in parallel. In order to reduce the execution time, we plan the cooking task taking into account if two actions can be simultaneously performed with two different hands. An example is shown in Figure 12 where we explain the method of determining the hand to be used, and explain the method of simultaneously executing multiple tasks.

First, the hand to be used in each action is determined by considering the distance from the hand to the object to be grasped. Figure 13 shows an example where the objects placed in the storage space (Right) and (Left) are grasped by the R-Hand and L-Hand, respectively. If there are multiple candidates, the hand to be used is determined by evaluating the length of the trajectory to execute the action. Figure 13 shows an example where the objects placed in the work space is grasped by the hand with shorter length of trajectory.

From the graph structure, we can extract two motion nodes which can be simultaneously performed. If two different hands can be assigned for these two actions, we plan these two actions to be executed simultaneously.

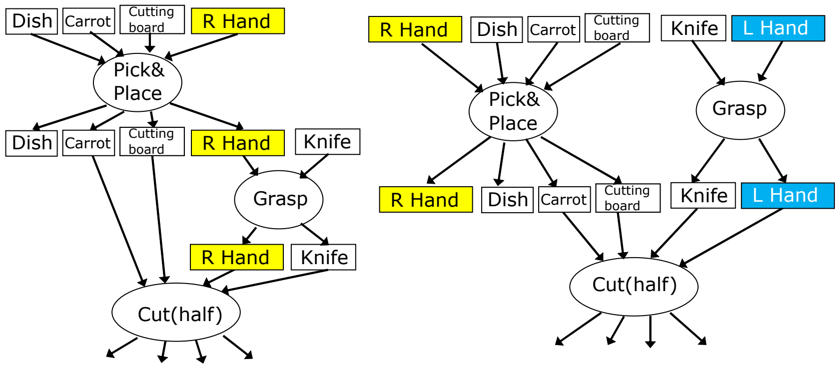

Figure 12. Motion planning taking the simultaneous execution with multiple hands (Left:single arm, right:dual-arm)

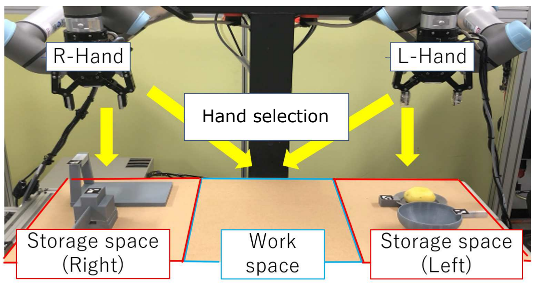

Figure 13. Selection of hand according to the area determined on the table

3.4. Motion Planning

In this section, we describe the motion planning method to realize the planned task sequence. Before executing the cooking task, we define multiple grasping poses to stably grasp a tool as shown in Figure 14. When executing the cooking task, we use a vision sensor to detect the pose of the tool. Once we obtain it, we select the highest priority grasping pose where the IK is solvable. The priority for grasping poses with the heuristic fashion. After the grasping pose is determined, the trajectory of the robot planned by using RRT-connect.

4. Experiment

To verify the effectiveness of our approach, we have conducted the experiments by using a real dual-arm manipulator.

Preprints (www.preprints.org) | NOT PEER-REVIEWED | Posted: 17 October 2022 doi:10.20944/preprints202210.0237.v1

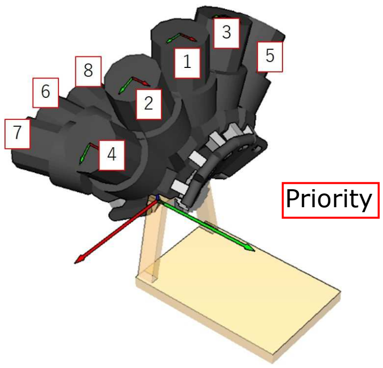

Figure 14. Definition of grasping pose considering its priority

4.1. Experimental setup

As a cooking recipe, we consider two simple Japanese sentences where they are equivalent to "Cut potatoes into half and pour them into a bowl" and "Heat oil in a frying pan over high heat and fry pork" in English. We used CaboCha [31] to obtain the dependency structure of Japanese sentences written in a recipe. In this experiment, we prepared a work space and a storage space in a kitchen environment. We assume that the objects are stored in the left or right storage spaces before the cooking task starts (Figure 15). We use two UR-3 robot arms in which a 2-Fingered 85mm Robotiq Gripper is attached at the tip. We have constructed cutting board, spatula, knife and stove by ourselves by using a 3D printer.

We attached an ArUco marker to each tool. The pose of the tool is detected by capturing the image of the ArUco marker by using a 2D RGB camera (Figure 16). We have prepared 8 grasping poses of the cutting board and 1 grasping poses of the knife.

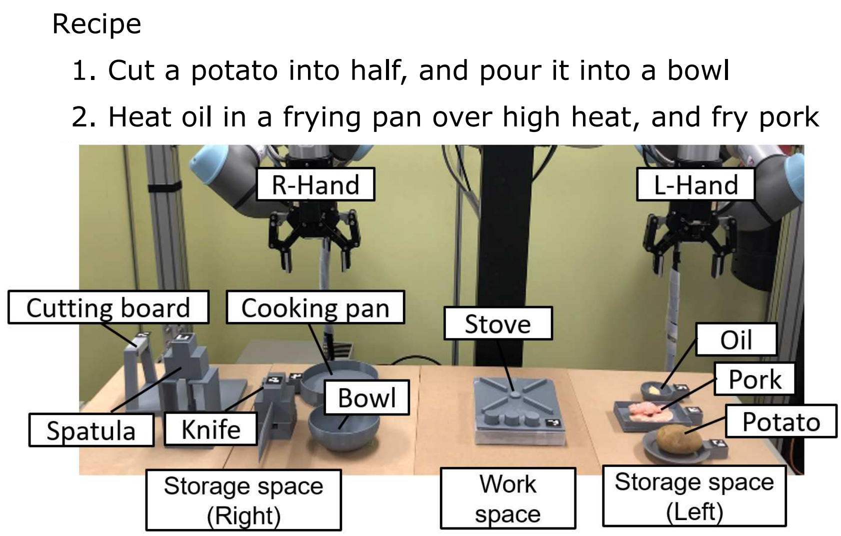

Figure 15. Experimental conditions

4.2. Results

We first generated a graph structure corresponding to each sentence. The graph structure corresponding to "Cut potatoes into half and pour them into a bowl" is shown in Figure 17 where motion nodes of functional unit stored in the main-motion DB is cut(half) and pour. To realize the cooking task, we need five functional units stored in the sub-motion DB where their motion nodes are three pick and place, a grasp and a release. Here,

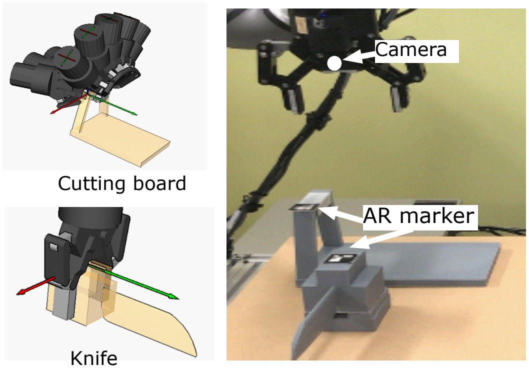

Figure 16. Camera equipped with the hand to see the AR markers attached at the tools

the pick and place of potato and grasp of the knife are performed in parallel. On the other hand, Figure 18 shows the motion of robot performing this cooking task.

The graph structure corresponding to "Heat oil in a frying pan over high heat and fry pork" is shown in Figure 19 where motion nodes of functional unit stored in the main-motion DB is pour, turn on, heat(High heat), stir fry. To realize the cooking task, we need two functional units stored in the sub-motion DB where their motion nodes are pick and place, grasp. Figure 20 shows the motion of robot performing this cooking task. As a result, the robot was able to cook autonomously, and we confirmed the effectiveness of our method for motion planning from cooking recipes consisting of simple sentences. In both cases, the calculation time of the motion planning is between 1 and 2 minutes.

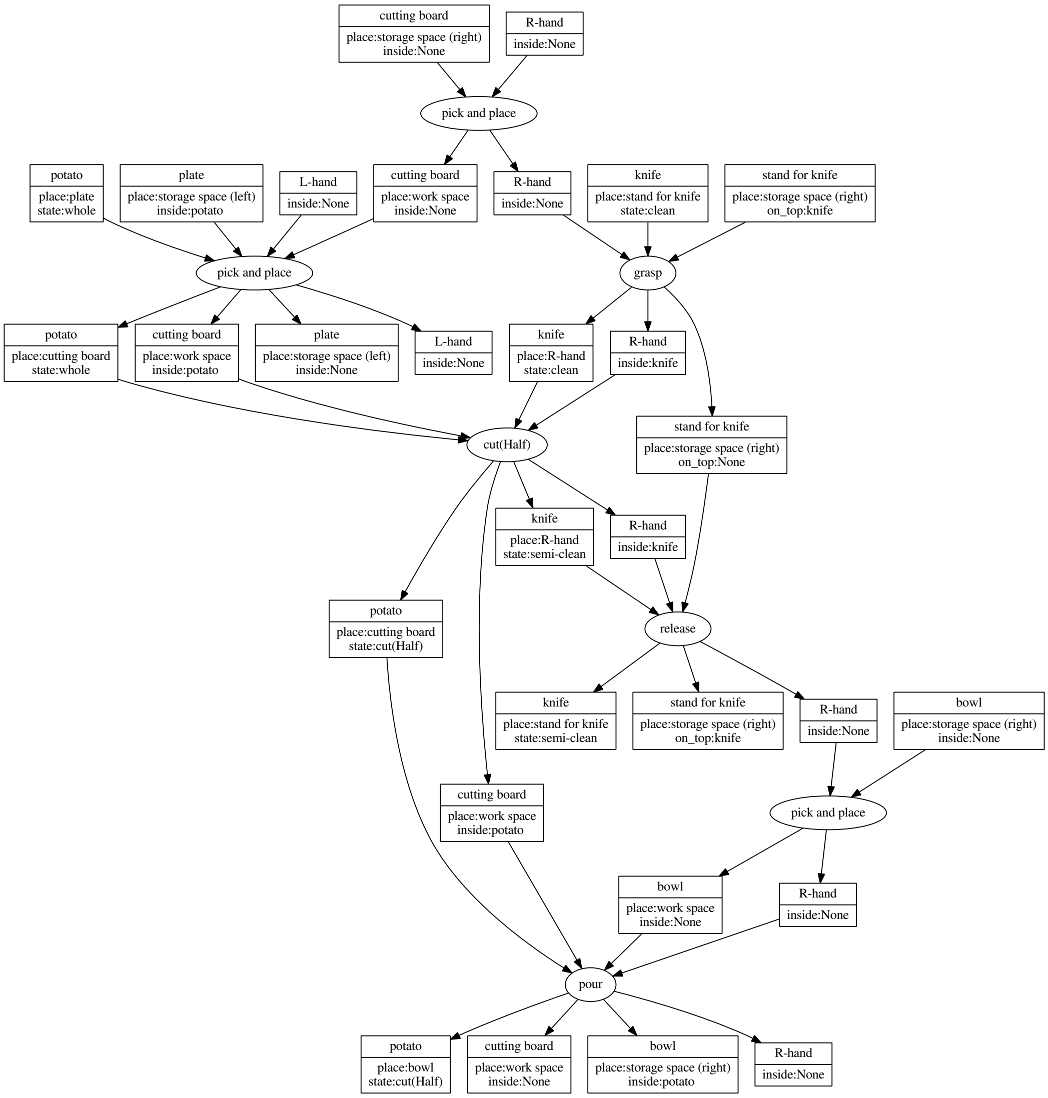

Figure 17. Obtained graph structure for the recipe "Cut potatoes into half and pour them into a bowl"

Preprints (www.preprints.org) | NOT PEER-REVIEWED | Posted: 17 October 2022 doi:10.20944/preprints202210.0237.v1

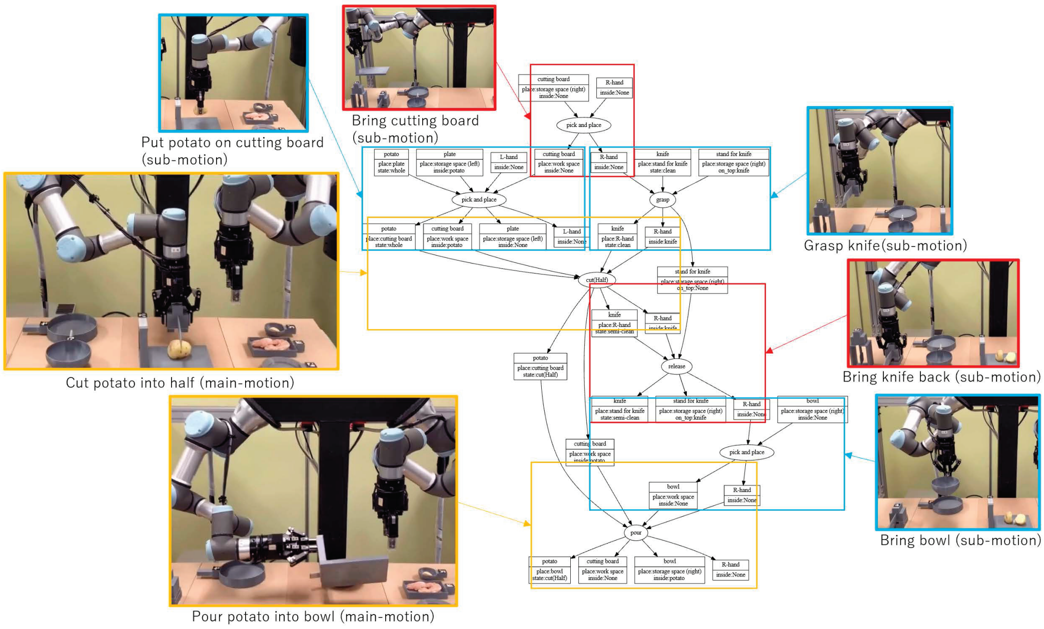

Figure 18. Result of experiment for the recipe "Cut potatoes into half and pour them into a bowl"

Preprints (www.preprints.org) | NOT PEER-REVIEWED | Posted: 17 October 2022 doi:10.20944/preprints202210.0237.v1

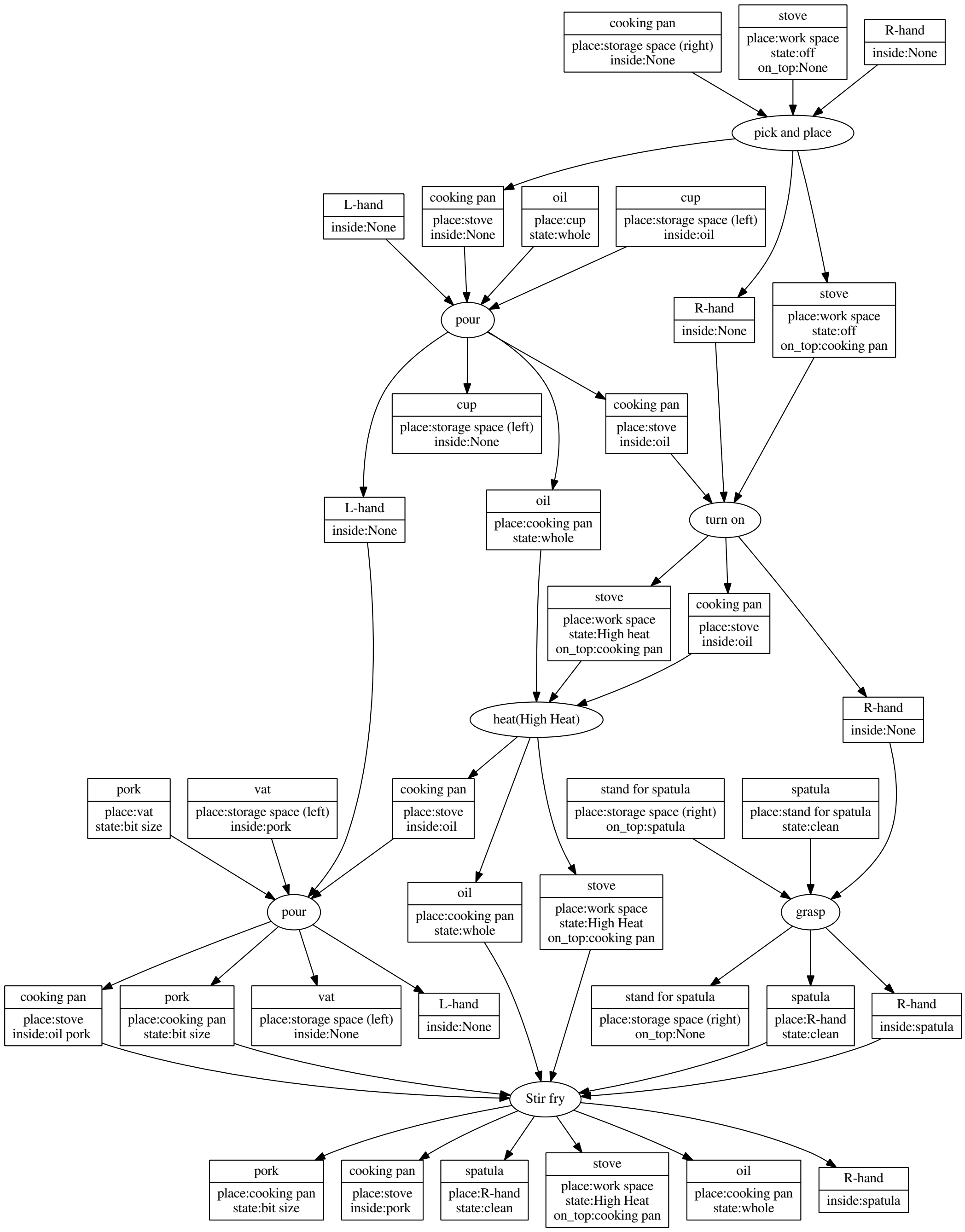

Figure 19. Obtained graph structure for the recipe "Heat oil in a frying pan over high heat and fry pork"

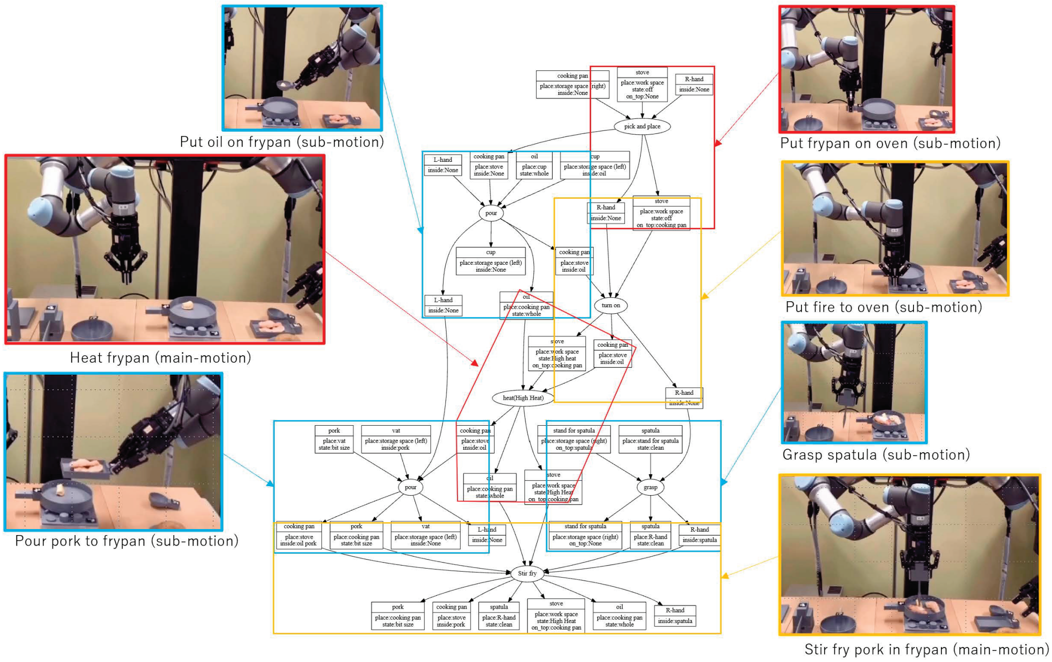

Figure 20. Result of experiment for the recipe "Heat oil in a frying pan over high heat and fry pork"

5. Conclusions

In this paper, we proposed a method for planning a cooking task by a dual-armed robot by representing the cooking task in a graph structure. Our proposed method can deal with actions and objects that are not explicitly described in the cooking recipe, and automatically add them to the task contents. We applied the proposed method to sentences of a simple cooking recipe, and confirmed that the dual-armed robot can perform the cooking task based on the result of task planning.

In the future, we will verify the effectiveness of the proposed method for cooking recipes that consist of more complex sentences.

Author Contributions: Conceptualization, K.H. and I.R.; methodology, K.T., T.K. and K.H.; software, K.T. and T.K.; validation, K.T.; formal analysis, K.T., N.Y., K.T., and I.R.; investigation, K.H. and T.K.; resources, K.H.; data curation, K.T.; writing—original draft preparation, K.T.; writing—review and editing, K.H., N.Y. I.R. and T.K.; visualization, K.T. and K.H.; supervision, K.H., T.K., N.Y. and I.R.; project administration, K.H., T.K., N.Y. and I.R.; funding acquisition, K.H. and N.Y. All authors have read and agreed to the published version of the manuscript.", please turn to the CRediT taxonomy for the term explanation. Authorship must be limited to those who have contributed substantially to the work reported.

Data Availability Statement: For more figures, videos and explanation on this research project, visit https://www.roboticmanipulation.org/res/cook.

Conflicts of Interest: The authors declare no conflict of interest.

References

1. Wolfe, J.; Marthi, B.; Russell, S. Combined task and motion planning for mobile manipulation. In Proceedings of the Proc. of 20th Int. Conf. on Automated Planning and Scheduling, 2010.
2. Wally, B.e.a. Flexible production systems: automated generation of operations plans based on ISA-95 and PDDL. *IEEE Robot. Autom. Lett*, *4(4)* **2019**, p. 4062.
3. Zhang, S.; Jiang, Y.; Sharon, G.; Stone, P. Multirobot symbolic planning under temporal uncertainty. In Proceedings of the Proc. of the 16th Conf. on Autonomous Agents and Multi-Agent Systems, 2017, p. 501–510.
4. Ratliff, N.; Zucker, M.; Bagnell, J.; Srinivasa, S. CHOMP: gradient optimization techniques for efficient motion planning. In Proceedings of the Proc. of IEEE Int. Conf. on Robotics and Automation, 2009.

5. Schulman, J.e.a. Motion planning with sequential convex optimization and convex collision checking. *Int. J. Robot. Res. 33(9)* **2014**, p. 1251.
6. Simeon, T.; Laumond, J.; Cortes, J.; Shbani, A. Manipulation planning with probabilistic roadmaps. *Int. J. Robot. Res. 23* **2004**, p. 729.
7. Kaelbling, L.; Lozano-Perez, T. Integrated task and motion planning in belief space. *Int. J. Robot. Res.* **2013**.
8. Garrett, C.; Lozano-Pérez, T.; Kaelbling, L.
9. Woosley, B.; Dasgupta, P. Integrated real-time task and motion planning for multiple robots under path and communication uncertainties. *Robotica 36(3)* **2018**, p. 353.
10. Wan, W.; Harada, K. Developing and comparing single-arm and dual-arm regrasp. *IEEE Robotics and Automation Letter, vol. 1, no. 1* **2016**, pp. 243—250.
11. W.Wan.; Harada, K. Regrasp planning using 10,000 grasps. In Proceedings of the Proc. of IEEE/RSJ Int. Conf. on Intelligent Robots and Systems, 2017, pp. 1929—1936.
12. Siciliano, B. Advanced bimanual manipulation: results from the dexmart project. *Advanced Bimanual Manipulation: Results from the Dexmart Project, vol. 80. Springer Science Business Media* **2012**.
13. Kruger, J.; Schreck, G.; Surdilovic, D. Dual arm robot for flexible and cooperative assembly. *CIRP Ann. 60(1)* **2011**.
14. Cohen, B.; Phillips, M.; Likhachev, M. Planning single-arm manipulations with n-arm robots. In Proceedings of the Proc. of 8th Annual Symposium on Combinatorial Search, 2015.
15. Kurosu, J.; Yorozu, A.; Takahashi, M. Simultaneous dual-arm motion planning for minimizing operation time. *Appl. Sci. 7(12)* **2017**, p. 2110.
16. Ramirez-Alpizar, I.; Harada, K.; Yoshida, E. Human-based framework for the assembly of elastic objects by a dual-arm robot. *Robomech. J. 4(1)* **2017**, p. 20.
17. Stavridis, S.; Doulgeri, Z. Bimanual assembly of two parts with relative motion generation and task related optimization. In Proceedings of the Proc. of 2018 IEEE/RSJ Int.l Conf. on Intelligent Robots and Systems, 2018, p. 7131–7136.
18. Moriyama, R.; Wan, W.; Harada, K. Dual-arm assembly planning considering gravitational constraints. In Proceedings of the Proc. of IEEE/RSJ Int. Conf. on Intelligent Robots and Systems, 2019, pp. 5566–5572.
19. Yamazaki, K.; Watanabe, Y.; Nagahama, K.; Okada, K.; Inaba, M. Recognition and manipulation integration for a daily assistive robot working on kitchen environments. In Proceedings of the 2010 IEEE Int. Conf. on Robotics and Biomimetics, 2010, pp. 196–201.
20. Mu, X.; Xue, Y.; Jia, Y.B. Robotic cutting: Mechanics and control of knife motion. In Proceedings of the 2019 Int. Conf. on Robotics and Automation (ICRA), 2019, pp. 3066–3072.
21. Yamaguchi, A.; Atkeson, C.G. Stereo Vision of Liquid and Particle Flow for Robot Pouring. In Proceedings of the 2016 IEEE-RAS 16th Int. Conf. on Humanoid Robots (Humanoids), 2016, pp. 1173–1180.
22. Inagawa, M.; Takei, T.; Imanishi, E. Japanese Recipe Interpretation for Motion Process Generation of Cooking Robot. In Proceedings of the 2020 IEEE/SICE Int. Symposium on System Integration, 2020, pp. 1394–1399.
23. Beetz, M.; Klank, U.; Kresse, I.; Maldonado, A.; Mösenlechner, L.; Pangercic, D.; Rühr, T.; Tenorth, M. Robotic roommates making pancakes. In Proceedings of the 2011 11th IEEE-RAS Int. Conf. on Humanoid Robots, 2011, pp. 529–536.
24. Lisca, G.; Nyga, D.; Bálint-Benczédi, F.; Langer, H.; Beetz, M. Towards robots conducting chemical experiments. *2015 IEEE/RSJ Int. Conf. on Intelligent Robots and Systems* **2015**, pp. 5202–5208.
25. Kazhoyan, G.; Beetz, M. Programming Robotic Agents with Action Descriptions. In Proceedings of the IEEE/RSJ Int. Conf. on Intelligent Robots and Systems (IROS), 2017.
26. Chen, H.; Tan, H.; Kuntz, A.; Bansal, M.; Alterovitz, R. Enabling robots to understand incomplete natural language instructions using commonsense reasoning. In Proceedings of the 2020 IEEE Int. Conf. on Robotics and Automation (ICRA), 2020, pp. 1963–1969.
27. Paulius, D.; Huang, Y.; Milton, R.; Buchanan, W.D.; Sam, J.; Sun, Y. Functional object-oriented network for manipulation learning. *2016 IEEE/RSJ Int. Conf. on Intelligent Robots and Systems* **2016**.
28. Paulius, D.; Jelodar, A.B.; Sun, Y. Functional Object-Oriented Network: Construction Expansion. *2018 IEEE Int. Conf. on Robotics and Automation* **2018**.
29. Paulius, D.; Dong, K.S.P.; Sun, Y. Task Planning with a Weighted Functional Object-Oriented Network, 2021, [arXiv:cs.RO/1905.00502].
30. Takata, K.; Kiyokawa, T. Ramirez-Alpizar, I.; Yamanobe, N.; Wan, W.; Harada, K. Efficient Task/Motion Planning for a Dual-arm Robot from Language Instructions and Cooking Images. In Proceedings of the Proc. of IEEE/RSJ Int. Conf. on Intelligent Robots and Systems.
31. CaboCha.