ROBOMECH Journal

RESEARCH ARTICLE Open Access

# Analysis of cooking recipes written in Japanese and motion planning for cooking robot

Masahiro Inagawa, Toshinobu Takei* and Etsujiro Imanishi

**Abstract**
Many cooking robots have been developed in response to the increasing demand for such robots. However, most existing robots must be programmed according to specific recipes to enable cooking using robotic arms, which requires considerable time and expertise. Therefore, this paper proposes a method to allow a robot to cook by analyzing recipes available on the internet, without any recipe-specific programming. The proposed method can be used to plan robot motion based on the analysis of the cooking procedure for a recipe. We developed a cooking robot to execute the proposed method and evaluated the effectiveness of this approach by analyzing 50 recipes. More than 25 recipes could be cooked using the proposed approach.

**Keywords:** Recipe analysis, Cooking robot, Motion planning

## Introduction
In recent years, the demand for cooking robots has increased rapidly to address the workforce scarcity resulting from declining birthrates and an aging population as well as to enhance operational efficiency. Cooking robots have already been developed and commercialized in various countries [1–5]. In general, such robots can cook programmed recipes. However, the programming of recipes in the context of robotic arms may be challenging for users without considerable robotics or coding expertise. The most widely used method to program robotic arms is the teaching playback method, in which each angle of the robot arms is taught to the robot by users through a device known as a teaching pendant [6]. The implementation of this approach involves considerable experience and technical skill because the robot habits must be visually confirmed and identified.

To address this problem, various teaching methods for robots have been developed, such as a method that defines each angle of the robot arms in a simulator through a graphical user interface (GUI) [7] and one that transforms hand and arm motions to robot motions using motion capture systems [8]. These methods are offline teaching methods because teaching can be performed even in the absence of the actual robot. In such methods, the robot motion is implemented through human motion in accordance with the recipe. However, users may require considerable time to learn how to operate the software, and the teaching devices may be expensive. As an alternative approach, researchers have attempted to realize automatic teaching by reusing manually generated primitive robot motions (sequence data of robot motion for each cooking operation) [9]. However, this alternative approach could not be generalized for all tasks and could not fully automate the teaching process. To perform motion planning for a cooking robot, Bollini et al. developed an approach to interpret recipes written in human languages [10]. The robot completed a cooking task by executing primitive cooking motions following recipe steps. However, the researchers only considered recipes that were easy to understand and involved simple sentences.

*Correspondence: takei@hirosaki-u.ac.jp
Faculty of Science and Technology, Hirosaki University, Hirosaki, Aomori, Japan

An approach to plan robot motion by considering recipes must be able to correctly understand the cooking procedure for a recipe. Herein, the cooking procedure reflects specific information, such as cooking operations including mixing and pouring, movement degree, tools required, and ingredients. However, to analyze the cooking procedure accurately, the following problems must be addressed. A cooking procedure may involve unnecessary information, especially in the case of recipes shared on the Internet. In general, the background for a recipe is not relevant in the cooking procedure. Furthermore, in the Japanese language, subjects and objects may be omitted and replaced. Thus, different authors may express the same cooking procedure in different ways.

To address these issues, the cooking procedure must be determined accurately based on uniform expressions. In this regard, we believe that the motion of robots can be planned by suitably extracting and organizing the cooking operations pertaining to different recipes. To this end, we propose a method to analyze recipes written in Japanese to achieve the automated teaching of stationary-type cooking robots [11, 12]. This method can enable motion planning for cooking robots in accordance with the robot system and cooking environment by generating the cooking procedure upon analyzing general recipes. A recipe analysis algorithm was developed, and the proposed approach was applied to an actual cooking robot system [11, 12]. We evaluated the proposed algorithm by testing it on several recipes randomly selected from different websites.

## Research approach

The key objective of this study was to develop an algorithm that analyzes Japanese recipes and performs automated motion planning for a cooking robot.

The target object was a cooking robot system with a multi-axis industrial manipulator, which is often used commercially. The robot was expected to be surrounded by cookware, heating equipment, and a cooking table, on which the robot could place ingredients and perform cooking tasks.

The target recipes were selected from among those uploaded to recipe-sharing websites on the Internet. The format of the recipes was first normalized because the recipe information was presented differently on different websites.

Cooking procedures often involve unique expressions, Internet slang, personal opinions and feelings of the authors, and omitted verbs and cookware information. As this information may lead to improper robot motion planning, it was necessary to extract only the necessary information and eliminate information irrelevant to the cooking process.

In Japanese grammar, the subject-predicate relation is determined by particles, and the context remains clear even when sentences are interchanged. Therefore, it was necessary to standardize the positional relations of the information, as the manner of expression differs among authors. To this end, when extracting the necessary information from the cooking procedure, the words were classified into certain categories and organized through an encoding process. It was assumed that if a sufficiently large database of such categorized words could be generated, the proposed word-classification system would be reasonably effective.

Once the cooking procedures were organized correctly based on the aforementioned processes, the robot motion was planned to execute primitive cooking operations for the procedures.

To plan the motion of an actual robot, we built a simulation environment that corresponded to an actual cooking environment and was of a similar scale as the actual environment. Furthermore, regardless of the type of motion, the cooking operation depends on parameters such as the cookware and ingredients; therefore, it was necessary for the motions to be generated identically, even if the cookware arrangement differed among recipes. We attempted to generate the motion by defining primitive motions that record the relative coordinates of the cookware and ingredients and then sequentially referring to those coordinates.

## Recipe analysis algorithm

The process flow of the proposed algorithm is divided into two stages. The first stage involves the recipe analysis, which is illustrated in the upper part of Fig. 1. In this stage, the information necessary to plan the robot motion is extracted from the recipe and sorted to generate the cooking procedure. The second stage involves the motion-planning process, which is illustrated in the lower part of Fig. 1. In this stage, the robot motion, including the avoidance of obstacles, is planned based on the output data of the recipe analysis process.

The details of these processes are as follows:

1. *Recipe analysis process*

   In this process, the cooking procedure is generated to plan the robot motion from recipes written in Japanese.

   In this process, the cooking procedure is generated to plan the robot motion from recipes written in Japanese. The proposed recipe analysis method can be used to analyze recipes written in Japanese that are uploaded on the Internet. Some authors write recipes

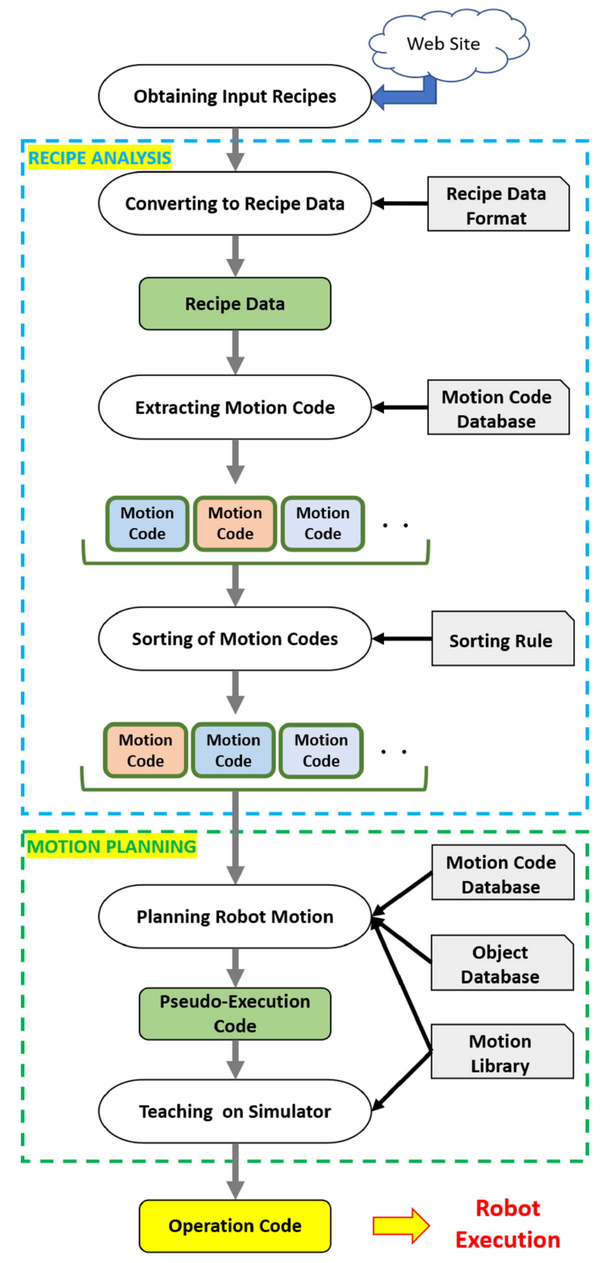


**Fig. 1** Recipe analysis algorithm for a cooking robot. The algorithm involves two processes,as indicated by the two flowcharts surrounded by dotted lines

in the style of diaries, in which descriptions of cooking procedures are incomplete. Therefore, we define prerequisites to evaluate whether the cooking robot can cook using the available recipe content. Our method analyzes recipes satisfying these prerequisites, which are described in Section "Recipe analysis".

**Table 1** Motion code classification

| code | Meaning |
|---|---|
| E | Large cookware |
| U | Small cookware |
| T | Tools |
| A | Seasonings |
| F | Ingredients |
| V | Operation |
| C | Change |
| S | Additional information |

It is necessary to address the omission of subjects and objects and the complexity of the grammatical structure during recipe analysis. Thus, the order of robot motions is specified according to the order of the cooking elements in the decomposed recipe, and any abbreviations are defined. These processes are realized using a generated database and sorting processes.

2. *Motion planning process*
   The robot motion is planned according to the cooking procedure defined in the recipe analysis stage. Using a simulator, the actual robot motion is planned by creating paths that connect the end point of the robot hand at the current moment to the end point generated automatically from the cooking procedure, as well as the subsequent motion pertaining to the cooking procedure. The planned motion is output as an execution code and converted into data that can be executed on an actual robot.

## Database for the algorithm

This section describes the three databases prepared for use in the algorithm.

### Motion code database

A motion code consists of symbol-based data corresponding to cooking information, such as nouns, verbs, degree of movement, and unique identifiers. The different types of motion codes are listed in Table 1. For example, for "E[0]," the letter "E" indicates large cookware, such as a heater, and "0" indicates the unique identifier in the motion code database (in this example, the E code database). For the word-by-word classification of recipes, Mori et al. [13] developed a corpus of tagged terms describing the motion and state of a cooking process and the ingredients. Although this classification method can identify the state and motion of the ingredients, finer classification is required to plan automated robot motion. Therefore, in our approach, eight categories were defined

**Table 2** Motion code database for the E code

| ID | Object shape | Current position | Base position | Heat permission | Grabbing permission | ... |
|---|---|---|---|---|---|---|
| 0 | Pan.obj | (0,0,0,0) | (12,15,0,0) | True | True | ... |
| 1 | Cooker.obj | (0,0,0,0) | (-8,-28,0,0) | True | False | ... |
| ... | ... | ... | ... | ... | ... | ... |

**Table 3** Object Database

| ID | Motion code type | Motion code ID | Grabbed | Current position | Last operation | ... |
|---|---|---|---|---|---|---|
| 0 | E | 4 | No | (0,5,0,0) | Release | ... |
| 1 | T | 12 | Yes | (−2,−8,4,16) | Hold | ... |
| ... | ... | ... | ... | ... | ... | ... |

based on the classification labels: E, U, T, A, F, V, C, and S. Each type of database contains cooking information. With regard to the words related to seasonings (A) and ingredients (F) included in these eight categories, nouns in the recipe sentences are automatically extracted and are classified into A or F, before being registered in the database. Nouns belonging to the other six categories are manually extracted and classified from the recipe sentences and registered in the database. The number of registered words is 8207, of which 2715, 5406, 5, 12, 14, 28, 3, and 24 are classified into A, F, C, E, S, T, U, and V, respectively. For example, the E and T code databases contained the shape data, reference points for use in the motion libraries, and heating availability information, whereas the F and A databases contained data pertaining to the initial positions. Table 2 presents sample data of the E code database.

## Object database

The object database contains information pertaining to the change of positions of cookware and their states (e.g., placed on fire or filled with water) during motion planning. This database is temporarily and automatically generated during the motion planning process and is used to plan the robot motion. If a motion code indicates an object, such as an ingredient (F) or cookware (E or T), the related object data are added to the object database. Information about the state of the object, such as the current position, parent node of the robot hand, and previous movement, is updated as the planning progresses. A sample object database is presented in Table 3. In the considered example, the current position and angle of the object, data of the object held by the robot, heating state, previous motion data, and other relevant information are registered as items, and the information is sequentially updated as the cooking progresses.

**Table 4** Motion library structure

| ID of V | Dependent code | Target points of robot based on position of an object | ... |
|---|---|---|---|
| 0 | A, F, S | (E \| U,center,0),(E \| U,center,+90)... | ... |
| 1 | EMPTY | (Gripper, open) | ... |
| ... | ... | ... | ... |

The ID of the code V indicates the motion, the dependent code type indicates the information required for the cooking operation, and the base motion indicates the time-series data of the actual robot motion

## Motion library

In the motion database, the dependency of motion code V on the other codes, according to the ID of V, and the primitive robot motions of the robot arm are registered, as indicated in Table 4. The motion library is a group of programs that convert the motion code V into specific robot motions. Multiple specific and primitive motions are manually registered in the motion library. In the motion planning, the library selects a specific motion associated with V from the motion group and its argument motion codes (e.g., A or F). Subsequently, the library generates time-series positions of the hand in three-dimensional coordinates and the joint angles to provide control commands for actuators in the robot.

For example, the word V[6], that is, "mix," is registered in the motion library as a mixing motion performed clockwise in a container containing ingredients. Here, V exhibits a dependency on E (container), T (cooking tools), and A (seasonings), and the primitive motion is clockwise mixing. This library is used to determine the dependencies of a code or the necessary information corresponding to the code V (see Table 4). Furthermore, this library is used to generate the motion of an actual robot. The primitive motion of the robot corresponding to the code V is described based on the motion reference points of the relevant objects. Therefore, the robot motion can be defined in the same library, even if the type and location of the cookware and ingredients change.

## Recipe analysis

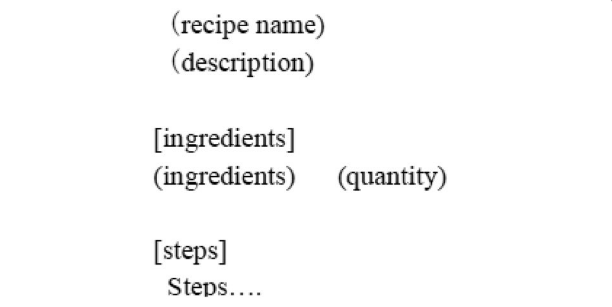
This section describes the algorithm used to determine the cooking procedures in the recipe analysis process based on the analysis of recipes written in Japanese. This process entails the analysis of recipes and extraction of the corresponding cooking procedures using the previously described motion code databases.

### Prerequisite

The prerequisites for the recipe analysis are defined as follows. Users evaluate whether the input recipes satisfy these prerequisites.

- The input is a Japanese-language recipe in the specified recipe format (Fig.2).
- The input recipe consists of general sentences that satisfy the following conditions:

  1. Each sentence should contain at least one word related to cooking (e.g., 野菜を炒める [fry vegetables]).
  2. The recipe clearly indicates the specific cooking procedures for the target dish and does not reference any other recipes.
  3. The cooking procedures do not change depending on conditions such as the size and softness of the ingredients.
  4. 4 No spelling or grammatical errors are present in any sentence.


(recipe name)
(description)

[ingredients]
(ingredients) (quantity)

[steps]
Steps….

**Fig. 2** Recipe data format

### Conversion of recipes to recipe data

As the expression style for recipes may differ among recipe-sharing websites (such as Cookpad or Allrecipes.com), we define a recipe data format, as shown in Fig.2. The data in this format are termed the recipe data, which are used as the target data for the analysis. Recipes extracted from the Internet are converted to this recipe data format.

### Extraction of motion code

To obtain the necessary information from the recipe data, words such as nouns and adjectives pertaining to the cookware and ingredients and verbs related to cooking are extracted and converted to the corresponding motion codes in the order of occurrence.

To execute this process, we use MeCab [14], an open-source morphological analysis engine, to divide the sentences into words and parts of speech. If the original form of a word exactly matches the motion code databases, the word is encoded into the motion code; otherwise, it is omitted.

### Sorting and insertion of motion codes

As the order of words in Japanese differs depending on the writer, a sorting process is used to standardize the order. The motion codes are treated as array data and are sorted and deleted in the last step in the recipe analysis stage. The sorting rules are as follows:

1. If S is encountered in the text before A or F, remove S.
2. If E is encountered, move E to the last cell of the array of the sorted motion code.
3. If U is encountered, move U to the last cell of the array of the sorted motion code.
4. If V that does not depend on any other code is encountered, move V to the last cell of the output array.
5. If V that depends on any other code is encountered, move V to the last cell of the output array; subsequently, move the codes that V is dependent on to the last cells of the output array.
6. If A, F, T, and S that do not correspond to V are encountered, move the codes to the last cell of the output array after complementing V' using the motion code database and motion library.
7. Otherwise, move the code to the last cell of the output array.

An example of this process is illustrated in Fig. 3. First, we remove the additional information (S) associated with the seasonings (A) and ingredients (F) in the behavior

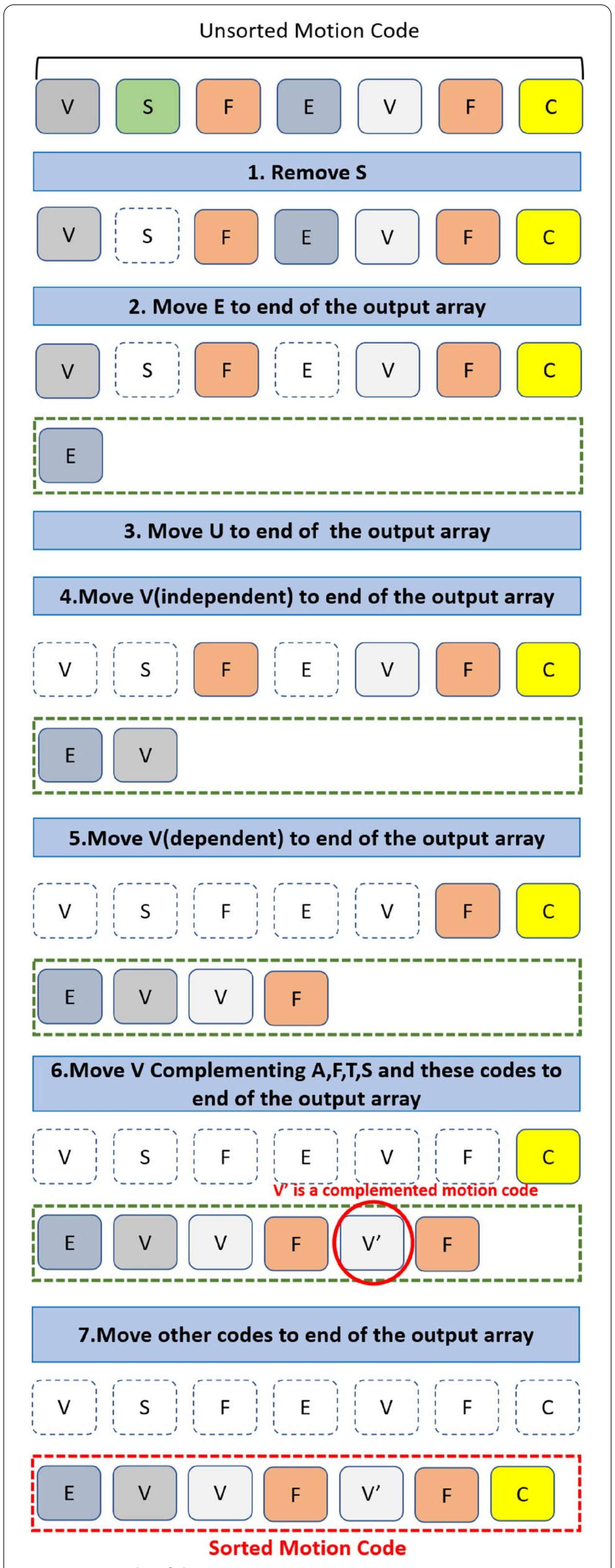

Unsorted Motion Code
V S F E V F C
1. Remove S
2. Move E to end of the output array
3. Move U to end of the output array
4.Move V(independent) to end of the output array
5.Move V(dependent) to end of the output array
6.Move V Complementing A,F,T,S and these codes to end of the output array
V' is a complemented motion code
7.Move other codes to end of the output array
E V V F V' F C
Sorted Motion Code

**Fig. 3** Example of the sorting process

code, which is not necessary for the motion generation. Next, the cookware data (E and U) are moved to the last cell of the array of the output motion code.

Two types of cooking operations (V) exist: operations (V-independent) that can be executed independently of the ingredient and cookware (T) and those (V dependent) dependent on other motion codes. If a motion is V-independent, it is moved to the last cell of the array of the output motion code. In the case of V-dependent motion, the code is moved to the last cell of the sorted array. Subsequently, the motion codes dependent on V are moved to the last cells of the sorted array. If a motion code is available only for A, F, T, and S, motion planning is not possible, because the motion code does not include the cooking operations. Therefore, it is necessary to complement these codes. V' is the complement of V and consists of operations missing in the cooking procedure of the recipe;it is constructed by referring to the operations required for the execution of A, F, T, and S. This complement is constructed by referring to each motion code database, and it completes the motion. Fig. 3 demonstrates the construction of the complement V', which indicates the cooking operation required to execute F, from the motion code database. Finally, the remaining operation codes are directly moved to the end of the sorted operation code array.

This sorting process simplifies motion planning as it groups the necessary information near the code V which indicates the cooking motion.

## Motion planning

This section describes the formulation of the operating procedure using the output motion codes. The operating procedure involves two functions, as described in Section "Planning robot motion": identifying the cooking procedure and complementing the cooking operations from the motion codes to plan the actual robot motion through offline teaching methods. This process generates the operation code, as indicated in the final process in Fig. 1.

### Planning robot motion

The motion codes are used to define the operating procedure by planning the motion of the cooking robot according to the cooking procedure through the motion library. This operating procedure is represented by pseudo-execution code, which involves the following two functions:

- SET $Arg^1$(Ingredient or Cookware), which sets the target points of the arm to $Arg^1$
- EXECUTE $Arg^1$(Cooking Operation), which executes $Arg^1$

The SET function sets the target points of the robot arm to generate the motion trajectory of the arm, and the EXECUTE function executes the cooking operation registered in the motion library.

Normally, a motion code is assigned to each argument. In certain cases, proper nouns related to the robot hand, such as the names of the ”heater” or ”gripper”, may be assigned. The names and actual processing contents are defined in the motion code database and motion library, as described in Section "Database for the algorithm".

The pseudo-execution code represents the process flow of motion codes shown in Fig. 4. By matching the motion codes with each database for each type of motion code, a pseudo-execution code can be created for the given code based on information such as the location and state of the object and the relevance of the other codes. Note that the generated operating procedure includes only the order in which the cooking motion is executed and not the actual robot motion path. In the actual program, the operating procedure is included in the motion generation process, as described in the next section.

### Simulator to train the robot motion

The actual motion of the robot can be generated via offline teaching using a cooking robot simulator (Fig. 5) developed using the Unity service [15].

In particular, the robot motion can be generated by determining a path connecting the starting point and a target point for the tip of the robot hand. Each point can be derived based on the operating procedure. Various methods have been proposed to identify the path connecting the starting and target points. In this study, the rapid-exploration random tree strategy was employed [16]. In this strategy, when the obtained paths involve collision with obstacles, a motion is generated with passage points placed around the obstacle. After the motion is generated, it is converted to an executable code, which is the final output that can be executed by the robot. The execution code is presented in Fig. 6.

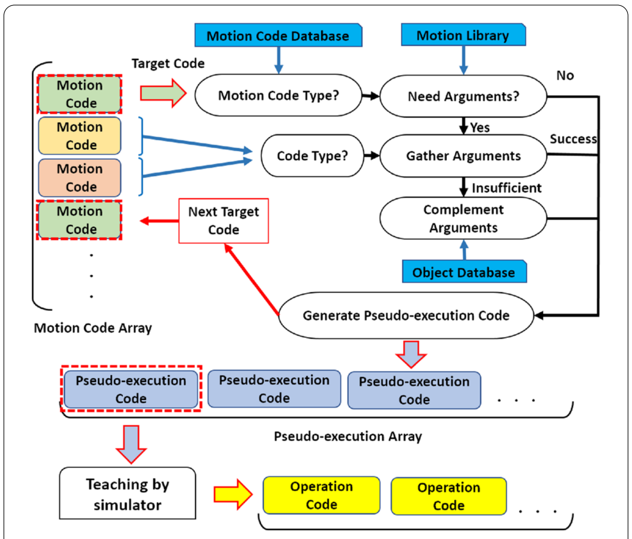


**Fig. 4** Process flow to create the operating procedures

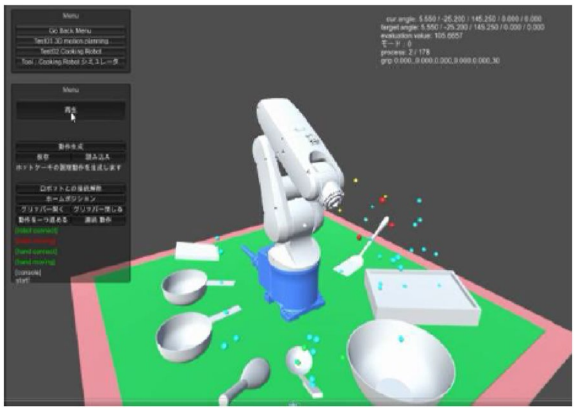


**Fig. 5** Cooking robot simulator

## Experiment

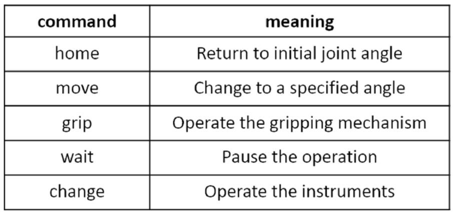
The main objective of the experiment is to confirm whether the robot can cook a recipe written in general Japanese by analyzing and planning the operation using the proposed method. Since the robot used was reproducible, sequence control was performed under the assumption that the cooking motion could be reproduced once it had been defined. To that end, the positions of the cookware and ingredients were precisely fixed.

### Experimental environment

The experimental environment had the following characteristics.

1. *Target robot*

   A VP-6242 robot manufactured by DensoWave was used. To handle several pieces of cookware, the robot hand used two robot grippers from Makerobot as a

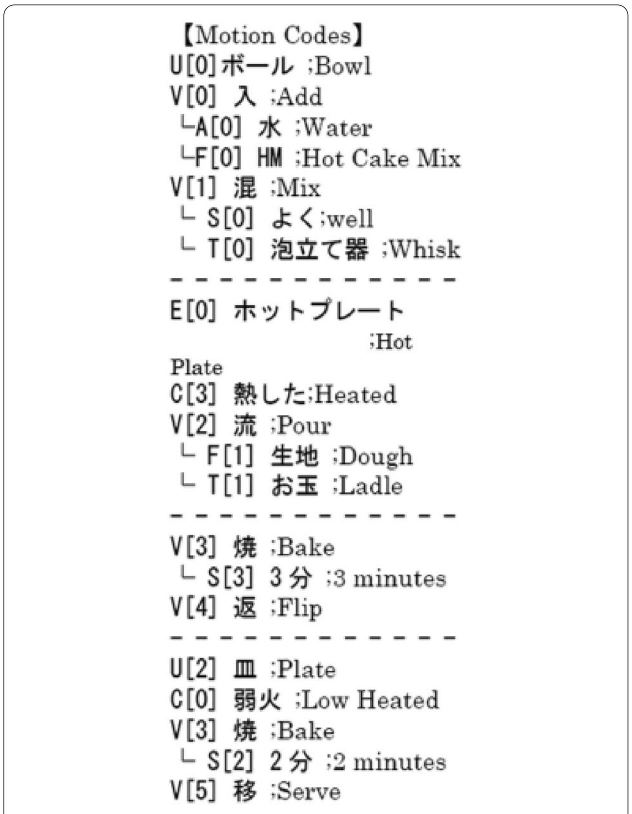


| command | meaning |
|---|---|
| home | Return to initial joint angle |
| move | Change to a specified angle |
| grip | Operate the gripping mechanism |
| wait | Pause the operation |
| change | Operate the instruments |

**Fig. 6** Types of execution code

single hand. The hold strength was determined with respect to a threshold value by increasing or decreasing the current value recorded by the sensor.

2. *Cooking environment*
   We arranged small bowls with the necessary amount of ingredients for the recipe and set the cookware around the target robot. A similar cooking environment was replicated in the simulation.
3. *Input data*
   As the recipe data, we used a pancake recipe that included the cooking motions of adding ingredients, mixing, scooping, pouring, and flipping. The recipe data are presented in Fig. 7.
4. *Database*
   Motion code databases and a motion library were required for analyzing the input data and planning the motion of the robot. The motion code databases were created by categorizing and inputting words such as those related to the ingredients and cookware required for a human to prepare the recipe. Seven primitive motions, namely grabbing, releasing, moving, pouring, adding, mixing, and flipping, were defined and input to the motion library. Each of these primitive motions was created manually using a robot simulator and then confirmed to be executed accurately by the actual robot.



```
【Motion Codes】
U[0]ボール ;Bowl
V[0] 入 ;Add
 └A[0] 水 ;Water
 └F[0] HM ;Hot Cake Mix
V[1] 混 ;Mix
 └ S[0] よく;well
 └ T[0] 泡立て器 ;Whisk
- - - - - - - - - - - -
E[0] ホットプレート
                  ;Hot
Plate
C[3] 熱した;Heated
V[2] 流 ;Pour
 └ F[1] 生地 ;Dough
 └ T[1] お玉 ;Ladle
- - - - - - - - - - - -
V[3] 焼 ;Bake
 └ S[3] 3 分 ;3 minutes
V[4] 返 ;Flip
- - - - - - - - - - - -
U[2] 皿 ;Plate
C[0] 弱火 ;Low Heated
V[3] 焼 ;Bake
 └ S[2] 2 分 ;2 minutes
V[5] 移 ;Serve
```

**Fig. 8** Recipe analysis results

## Result

1 *Recipe analysis*

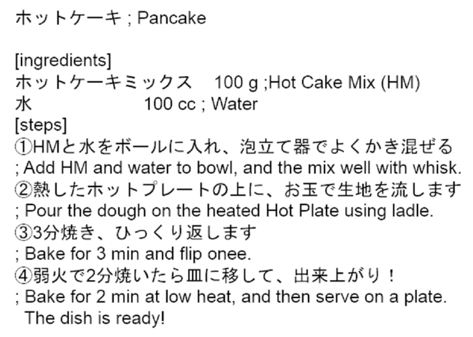
The results of recipe analysis using the proposed approach are shown in Fig. 8. According to the pancake cooking procedure, the words related to cooking were extracted as accurate motion codes without unnecessary information.

2 *Action generation*

The robot motion was planned based on the motion code in the simulator. The operating procedure is not presented here, as it was generated during the planning process. The output executable code is shown in Fig. 9. Overall, 169 executable codes were obtained.

3 *Actual execution*

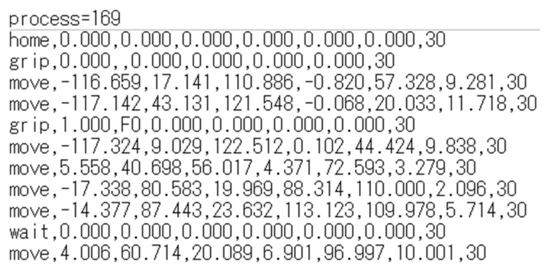
Fig. 10 shows a series of images depicting the robot cooking pancakes during the experiment. The first image shows the robot in starting position, prepared to cook. The second and third images show the robot picking up a small bowl with hot cake mix, pouring it into a mixing bowl, and placing it on the cook-



ホットケーキ ; Pancake

[ingredients]
ホットケーキミックス　100 g ;Hot Cake Mix (HM)
水　　　　　　　100 cc ; Water
[steps]
①HMと水をボールに入れ、泡立て器でよくかき混ぜる
; Add HM and water to bowl, and the mix well with whisk.
②熱したホットプレートの上に、お玉で生地を流します
; Pour the dough on the heated Hot Plate using ladle.
③3分焼き、ひっくり返します
; Bake for 3 min and flip onee.
④弱火で2分焼いたら皿に移して、出来上がり！
; Bake for 2 min at low heat, and then serve on a plate.
The dish is ready!

**Fig. 7** Pancakes recipe data



```
process=169
home,0.000,0.000,0.000,0.000,0.000,0.000,30
grip,0.000,,0.000,0.000,0.000,0.000,30
move,-116.659,17.141,110.886,-0.820,57.328,9.281,30
move,-117.142,43.131,121.548,-0.068,20.033,11.718,30
grip,1.000,F0,0.000,0.000,0.000,0.000,30
move,-117.324,9.029,122.512,0.102,44.424,9.838,30
move,5.558,40.698,56.017,4.371,72.593,3.279,30
move,-17.338,80.583,19.969,88.314,110.000,2.096,30
move,-14.377,87.443,23.632,113.123,109.978,5.714,30
wait,0.000,0.000,0.000,0.000,0.000,0.000,30
move,4.006,60.714,20.089,6.901,96.997,10.001,30
```

**Fig. 9** Sample output executable code

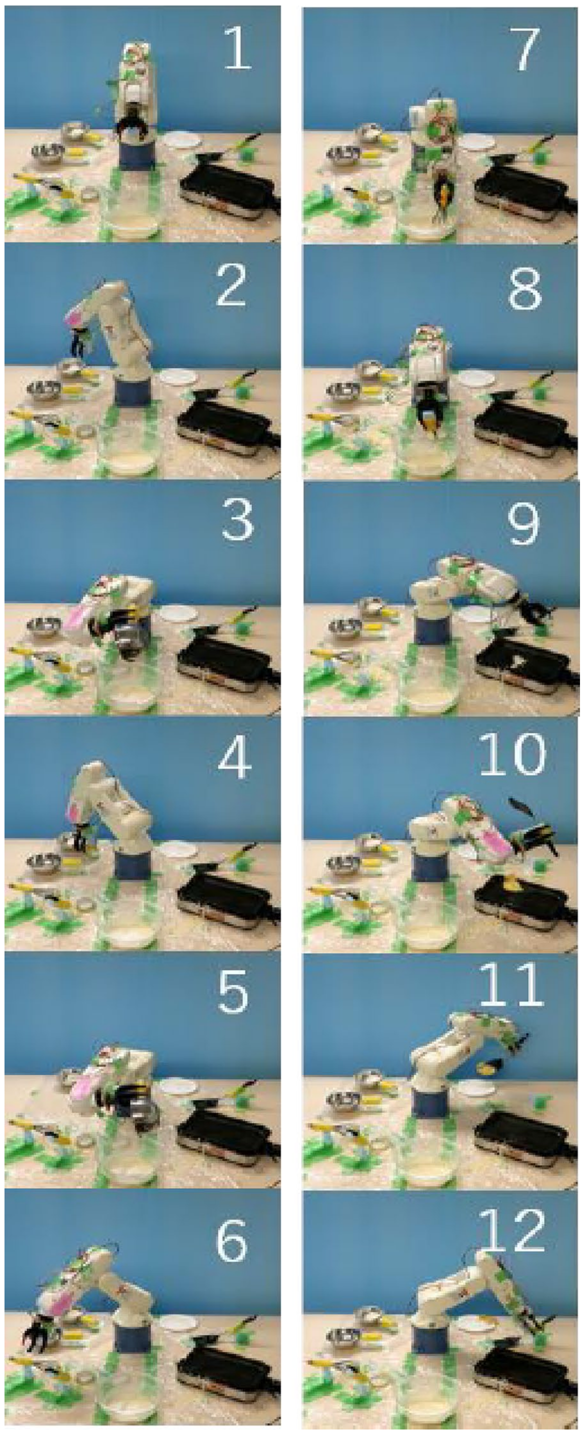


**Fig. 10** Actual robot preparing pancakes

ing workspace. The fourth and fifth images show the robot picking up a small bowl containing water, pouring it into a mixing bowl, and placing it on the cooking workspace. The sixth image shows the robot picking up a whisk. The seventh and eighth images show the robot mixing the ingredients in the bowl using the whisk. The ninth image shows the robot pouring dough into a hot plate. The tenth and eleventh images shows the robot turning the pancake over and moving it to the plate, respectively. Finally, the twelfth image shows the robot cleaning up the workspace.

Our recipe analysis method does not depend on the robot system. Differences between hardware components can be resolved using the motion library during motion planning. An example of the application of our method to different hardware has already been published in SII2021 [17]. The robot can execute seven motions based on the motion library prepared for the experiment: grabbing, releasing, moving, pouring, adding, mixing, and flipping. From the results of this experiment, we confirmed that the robot could execute grabbing, releasing, moving, pouring, adding, and mixing without any problems. However, pouring and flipping could not be performed in some cases, depending on the characteristics (softness) of the ingredients.

## Evaluation of the recipe analysis appropach

The aim of the aforementioned experiment was to verify whether the proposed approach could be applied to plan robot motion based on a recipe written in Japanese. Therefore, we used a relatively easy recipe consisting only of words known to be registered in the motion code database.

In this study, we validated only the recipe analysis process shown in Fig. 1 to verify the applicability of the proposed method. In the verification experiment, we did not evaluate the specific motion plan of the actual robot.

### Preparation

For the verification, it was necessary to increase the amount of data in each database. In particular, the motion code database is the most critical database in the recipe analysis and indicates the number of data types that can be recognized for an input recipe.

Ideally, the database should contain all the words related to cooking that are currently available. However, adding all such words is not realistic; therefore, we attempt to diversify the vocabulary for two motion code types: A and F. These data were obtained by randomly extracting approximately 1,000 recipes from a recipe-sharing website called Cookpad [18], converting them into recipe data, extracting the ingredient-related data from the recipes, and manually mapping the data to the motion code types A and F. The employed database contained 8123 names of ingredients and seasonings. For other databases, we entered the minimum required data.

For example, for the code type V, we entered a total of 24 verbs, including "put," "mix," and "bake."

The total number of data points in the motion code databases was 8207. Furthermore, information such as that related to the object shape and heating availability, which was not particularly relevant to the recipe analysis process, was omitted at the time of creating the databases because the validation experiment involved steps only until the motion code conversion.

### Input recipe data

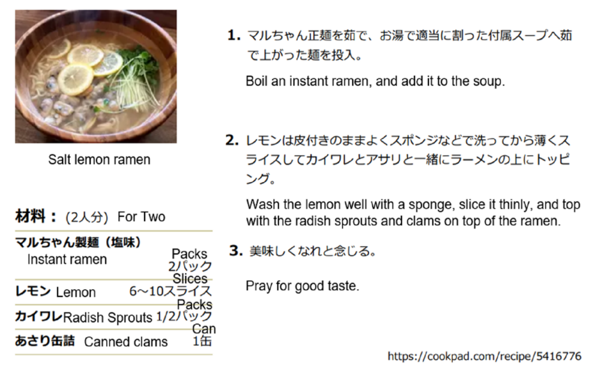
In the experiment, 50 recipes were randomly extracted from a search for side dishes in Cookpad [18] and converted to the recipe data format.

The recipe data that satisfied the following conditions were excluded from the input because they did not satisfy the prerequisites for the recipe analysis method:

- Sentences not including verbs and nouns or cooking information (an example is shown in Fig. 11)

  Example: 美味しくなるように念じる

  Meaning: Pray for good taste
- Conditional statements
  Example: もし味が薄ければ、塩を入れる
  Meaning: If the taste is bland, add salt
- Sentences not pertaining to the specific cooking procedure
  Example: エビフライを作る
  Meaning: Make fried shrimp.
- Spelling or grammatical errors


Salt lemon ramen

材料：(2人分) For Two

| | |
|---|---|
| マルちゃん製麺（塩味） Instant ramen | Packs 2パック |
| レモン Lemon | Slices 6～10スライス |
| カイワレ Radish Sprouts | Packs 1/2パック |
| あさり缶詰 Canned clams | Can 1缶 |

1. マルちゃん正麺を茹で、お湯で適当に割った付属スープへ茹で上がった麺を投入。

   Boil an instant ramen, and add it to the soup.

2. レモンは皮付きのままよくスポンジなどで洗ってから薄くスライスしてカイワレとアサリと一緒にラーメンの上にトッピング。

   Wash the lemon well with a sponge, slice it thinly, and top with the radish sprouts and clams on top of the ramen.

3. 美味しくなれと念じる。

   Pray for good taste.

https://cookpad.com/recipe/5416776

**Fig. 11** Example of sentences not including verbs and nouns or cooking information

  Example: 入れる→入る
  Meaning: put → pu (Spelling error for 'put')

After exclusion, recipe data for 33 recipes remained and were used in the experiment.

### Evaluation method

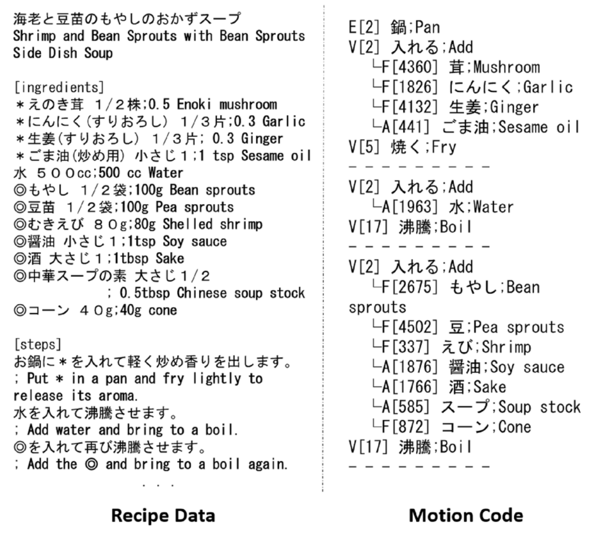
The proposed recipe analysis algorithm was used to convert the recipe data into motion codes. The order of coincidence of the output motion codes and expected codes was examined. The correct data were provided by an individual who was familiar with the recipes.

### Result

In the experiment, 19 recipes were successfully converted, with a success rate of approximately 57.6%. A sample recipe that could be transformed into motion codes is shown in Fig. 12. The causes of failure in motion code conversion and their relative share are shown in Fig. 13. The inability to address unique and ambiguous expressions was the most frequent cause (40%), followed by a lack of content in the motion code database (33%), redundant content processing (13%), and word segmentation failures in the recipe data (13%).

## Discussion

Although the recipes used in the experiments contained insufficient sentences, the robot motion planning could be satisfactorily realized using the proposed method. Examples of the insufficient sentences are "ボウルに入れる, Put in a bowl" and "小麦粉を入れる, Put the flour in," in which the objects (flour and bowl, respectively) are omitted. "ボウルに小麦粉を入れる, Put the flour in a bowl" is the correct sentence corresponding to those insufficient sentences.


```
海老と豆苗のもやしのおかずスープ
Shrimp and Bean Sprouts with Bean Sprouts
Side Dish Soup

[ingredients]
＊えのき茸 1/2株;0.5 Enoki mushroom
＊にんにく(すりおろし) 1/3片;0.3 Garlic
＊生姜(すりおろし) 1/3片; 0.3 Ginger
＊ごま油(炒め用) 小さじ1;1 tsp Sesame oil
水 500cc;500 cc Water
◎もやし 1/2袋;100g Bean sprouts
◎豆苗 1/2袋;100g Pea sprouts
◎むきえび 80g;80g Shelled shrimp
◎醤油 小さじ1;1tsp Soy sauce
◎酒 大さじ1;1tbsp Sake
◎中華スープの素 大さじ1/2
                 ; 0.5tbsp Chinese soup stock
◎コーン 40g;40g cone

[steps]
お鍋に＊を入れて軽く炒め香りを出します。
; Put ＊ in a pan and fry lightly to
release its aroma.
水を入れて沸騰させます。
; Add water and bring to a boil.
◎を入れて再び沸騰させます。
; Add the ◎ and bring to a boil again.
                    . . .
```

**Recipe Data**

```
E[2] 鍋;Pan
V[2] 入れる;Add
  └F[4360] 茸;Mushroom
  └F[1826] にんにく;Garlic
  └F[4132] 生姜;Ginger
  └A[441] ごま油;Sesame oil
V[5] 焼く;Fry
- - - - - - - - - - -
V[2] 入れる;Add
  └A[1963] 水;Water
V[17] 沸騰;Boil
- - - - - - - - - - -
V[2] 入れる;Add
  └F[2675] もやし;Bean
sprouts
  └F[4502] 豆;Pea sprouts
  └F[337] えび;Shrimp
  └A[1876] 醤油;Soy sauce
  └A[1766] 酒;Sake
  └A[585] スープ;Soup stock
  └F[872] コーン;Cone
V[17] 沸騰;Boil
- - - - - - - - - - -
```

**Motion Code**

**Fig. 12** Example of a successful motion code conversion

In addition, using the motion code and motion library, paths leading to individual motions can be generated.

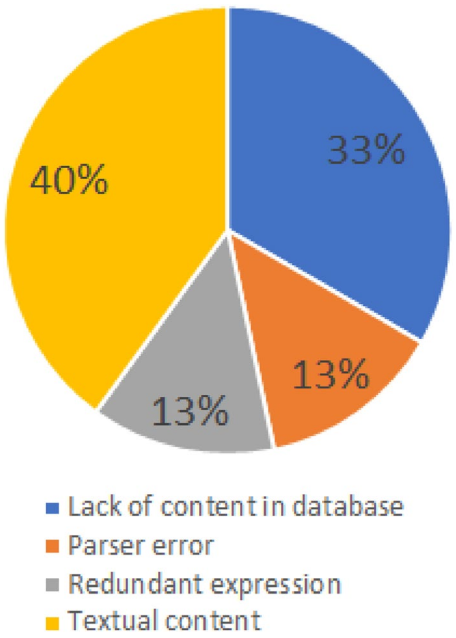


**Fig. 13** Causes of failure in the motion code conversion and their relative share

Because the proposed approach can define robot motions that can be executed by an actual robot, it was considered that the approach can be effectively applied to cooking robots.

However, this method involves several databases for the recipe analysis and robot motion planning; therefore, the scope of application is dependent on the databases. For example, the motion code database can extract the words necessary for cooking and classify them; however, words that are not included in the database cannot be evaluated. In addition, the motion library can generate motions independent of the position and type of the tools; however, the library must be able to consider all possible conditions, and malfunctions may occur in the case of improper inputs. Therefore, to generalize the method, it is necessary to develop a reasoning algorithm for unknown keywords and to create a motion library that is correctly defined for each primitive motion, in addition to expanding the classification database for each motion code type.

It is necessary to consider the amount of data required for the motion code database and the motion library. For the motion code database, we predict that the amount of data will depend on the cuisine to be analyzed, as different cuisine uses different ingredients, cooking utensils, and cooking motions. We believe that these parameters will need to be considered in the future.

To execute the simulation on an actual robot, considerable information processing is required in advance, such as the adjustment of cookware and other aspects, to make the actual layout consistent with that in the simulation. In the experiments, the robot could not handle the pancake batter effectively in certain cases. In particular, when we used the motion library to generate the motion, it was assumed that the ingredients and cookware always behave in a certain manner. The ingredients were likely improperly handled because of a slight shift in the position of the cookware in the actual situation.

It is difficult to maintain the same environment, including food (or ingredient) properties such as position and shape, whenever a recipe is cooked. Therefore, for fully automating a cooking robot, the robot system must be more sophisticated, with the ability to recognize the actual environment using external sensors, control motions based on sensor feedback, and so on. We intend to focus on this aspect in future work.

In our future studies, we will also consider the evaluation of various specific cooking robot motions. In the present study, we found that the proposed method could correctly analyze about 58% of the 33 recipes considered. We could successfully solve certain problems in recipes written in Japanese, such as redundant expressions, omission of objects, and word-order discrepancies among writers, by using the proposed recipe analysis method.

However, there were some recipes to which the proposed method could not be applied. The failure observed in the verification experiment was caused by the following factors.

1. *Inability to address unique and ambiguous expressions*

   Depending on the manner of expression used in the recipe, the effectiveness of word extraction may vary. In Example 1, the adverb may have multiple meanings, such as heating time, degree of heating, or degree of movement. Moreover, when casual expressions represent motion, the meaning cannot be captured accurately. Furthermore, the method cannot address statements that cite other recipes or involve difficult phrasing, as indicated in Example 2.

   (Example 1)「ガンガン煮る」

   (Meaning 1) boil (well / roughly / for a long time)

   (Example 2)「今回は冷凍のものを使ったが生のものでも良い」

(Meaning 2) I used frozen ones this time, but raw ones are also acceptable

2. *Insufficient content in the motion code database*
When the recipe data involve cookware or ingredients that are not registered in the database, the reordering process cannot be performed suitably, because the sorting of the words fails. In Japanese, words can be written in three different scripts—hiragana, katakana, and kanji. Even if the words convey the same meaning, they cannot be coded if the equivalent notation is not registered.
3. *Redundant content in recipe data*
As shown in Example 3, if the recipe contains comments that are not directly related to the recipe content, the motion coding fails because statements unrelated to the recipe cannot be processed.
(Example 3) 「食べる時にすりゴマかけても美味しい」
(Meaning 3) It's good with sesame when you eat
4. *Failure of word segmentation*
In the actual process, the default setting of MeCab [14] is employed. Therefore, if the word separator is inaccurate, the coding may fail.
(Example 4)「ミートボール」→「ミート」「ボール」
(Meaning 4) meatballs → meat, balls

Considering these aspects, to improve the generalization performance of this method, it is necessary to diversify the data in the motion code database and to implement modifications to address unique and ambiguous expressions. This limitation was expected, to a certain extent, before the experiment. To apply the approach to recipes on the Internet, which may involve more colloquial expressions and expressions requiring more types of databases, it is necessary to address the coding of casual phrases, especially because writers often employ contractions and verb paraphrasing. In this regard, it is also necessary to revise the sorting process by, for example, enhancing the adverbial phrases that can be used and adding judgment-based processing to modify phrases with clauses.

In certain failure cases, redundant recipe content could not be removed. Such redundancies are often found in recipes posted by ordinary Internet users, who separate words with arrows or write comments in between the steps of a cooking procedure. The proposed algorithm was developed to eliminate emotional expressions and impressions. Because the proportion of this cause of failure was lower than that of the other causes, the algorithm is considered to have succeeded in removing redundancy to a certain extent. However, when the redundant part contains words related to cooking, the information is recognized as information to be processed, which may lead to errors. In this scenario, it is necessary to determine whether the expression is redundant at the sentence level instead of at the word level.

The proposed algorithm employs MeCab [14], which is a library for morphological analysis; however, in certain cases, failure occurred in the word segmentation stage owing to the use of the default settings. Therefore, it is necessary to add more cooking- and ingredient-related terms to the MeCab [14] database.

The applicability of the proposed method is limited owing to the considered assumptions. Thus, 34% of the 50 recipes randomly selected could not be coded. To alleviate this restriction, it is necessary to perform a language-proofing process before the recipe analysis is conducted. It is considered that the method can be generalized further by correcting language errors with reference to other recipes when specific parts of a recipe are not present.

It must be noted that a multi-axis industrial manipulator, which is used in many commercial machines, was adopted as the cooking robot system in this study. However, the proposed method is independent of the recipe analysis and motion generation processes. Therefore, by changing the motion generation process, the system can be used for various types of cooking robots. In future work, the proposed analysis method will be used to evaluate a different cooking robot system.

## Conclusion

In this study, we developed a method to plan the motion of a stationary cooking robot based on the results of analyzing recipes written in Japanese.

Considering a simple pancake recipe, we performed experiments on an actual machine and confirmed that the correct motion procedure for the recipe could be defined to generate the robot motion.

To investigate the general performance and applicability of the proposed method, we coded general recipes extracted from the Internet and verified that more than half of the recipes could be converted correctly without any modification.

Using a motion code database, the algorithm could analyze approximately 60% of 33 recipes. Nearly 70% of the failures were due to a lack of unique expressions and proper nouns in the database. The results indicate that randomly extracting the names of ingredients and seasonings did not improve the performance noticeably. Although the number of verbs was less than the number of ingredient and seasoning names, the verbs did not appear to lead to failure and likely enhanced the performance.

To enrich the database, it must be made more effective through the prioritization of duplicate verbs in the input and relevant comparisons with multiple recipes. In terms of proper nouns, it is inefficient to increase the number of randomly generated nouns. Therefore, it may be necessary to design a more sophisticated database mechanism, such as one that enables keyword classification for similar expressions.

In future studies, we intend to reevaluate the obtained experimental data and improve the algorithm and broaden its applicability. Furthermore, it is necessary to improve the recipe analysis algorithm and add a text-proofreading process to improve the generalization performance.

**Acknowledgements**
Not applicable.

**Authors' contributions**
The first author conducted the study under the supervision of the second author. The first and second authors wrote the manuscript. All authors read and approved the final manuscript.

**Funding**
Not applicable.

**Availability of data and materials**
Not applicable.

## Declarations

**Ethics approval and consent to participate**
Not applicable.

**Consent for publication**
All authors consent for the publication of the manuscript in Robomech Journal.

**Competing interests**
The authors declare no competing financial interests.

Received: 12 October 2020 Accepted: 31 May 2021
Published online: 10 June 2021

**References**
1. Spyce. https://www.spyce.com/. Accessed 20 Aug 2020
2. Jing Dong X Future Restaurant. https://prtimes.jp/main/html/rd/p/000000049.000034149.html. Accessed 12 Sept 2020.
3. Moley. http://www.moley.com. Accessed 15 May 2020
4. OctChef. https://connected-robotics.com. Accessed 4 May 2020
5. Bot Chef. https://techable.jp/archives/94801. Accessed 15 Jan 2020
6. Yoshihiro Kusuda, Takashi Yagi. A Practical Guide to Introducing Industrial Robots in Illustration. Nikkan Kogyo Shimbun, Japan. 1999. **[Translated from Japanese]**
7. Yuta S, Daisuke S, AI W, Masahiko I, Takeo I. Cooky: Development of a Cooking Order Interface and Cooking Robot. The 17th Workshop on Interactive Systems and Software (WISS2009), pp 75-80. 2009
8. Yuuta F, Yukinobu H. Experiment of motion control of KHR-2HV by using a motion capture system 2. Proceedings of the Japan Joint Automatic Control Conference 2010, Volume 53, The 53rd Japan joint automatic control conference, session ID 126, Pages 24, Released February 03, 2011
9. Takamune S, Tatsuhiko S, Keiichi S. Research of Off-line Teaching Support System for Automatic Programming of Welding Robot (OS-1, S-14, 15, 16, 19 Work Robot). Proceedings of the Kansai Section Conference, Session ID 209. 2008
10. Bollini M, Tellex S, Thompson T, Roy N, Rus D (2013) Interpreting and executing recipes with a cooking robot. Experimental robotics. Exp Robotics Springer Tracts Adv Robotics 88:481–495
11. Masahiro I, Toshinobu T, Etsujiro I. Interpreting and Motion Generation for a Cooking Robot. ROBOMECH2019 The Robotics and Mechatronics Conference 2019, 1P2-C11. 2019
12. Masahiro I, Toshinobu T, Etsujiro I. Japanese Recipe Interpretation for Motion Process Generation of Cooking Robot. 2020 IEEE/SICE International Symposium on System Integration, Paper We2E.5. 2020
13. Shinsuke M, Yoko Y, Tetsuro S, Hirokuni M. Definition of flow graphs for recipe texts. Report of Information Processing Society of Japan, Report of the Society for Natural Language Processing 2013-NL-214(13), 1-7, 7 November 2013
14. MeCab: Yet Another Part-of-Speech and Morphological Analyzer. http://taku910.github.io/mecab/. Accessed 20 Jan 2020
15. Unity. https://unity.com/. Accessed 20 Jan 2020
16. S.M.LaValle. Rapidly-exploring random trees: A new tool for path planning. Computer Science Dept. Oct. vol.98. no.11. 1998
17. Masahiro I, Toshinobu T, Etsujiro I. Development of a Tower-type Cooking Robot. Proceedings of the 2021 IEEE/SICE International Symposium on System Integration Iwaki, Fukushima, Japan, January 11-14, 2021
18. Cookpad. https://cookpad.com. Accessed 21 Mar 2020