Recipe Analysis Method and Implementation for Motion Planning of Cooking Robot

Masahiro Inagawa

Hirosaki University

Toshinobu Takei (✉ takei@hirosaki-u.ac.jp)

Hirosaki Daigaku <https://orcid.org/0000-0002-3270-1034>

Etsujiro Imanishi

Hirosaki University

Research Article

Keywords: Recipe analysis, Cooking robot, Motion planning

Posted Date: October 21st, 2020

DOI: <https://doi.org/10.21203/rs.3.rs-93682/v1>

License: © ⓘ This work is licensed under a Creative Commons Attribution 4.0 International License.

[Read Full License](#)

Recipe Analysis Method and Implementation for Motion Planning of 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 of the existing robots must be programmed according to the recipes to enable cooking through the robotic arms, which requires considerable time and expertise. Therefore, this paper proposes a method to allow a robot to cook by analyzing the recipes available on the internet without any programming. The proposed method can plan robot motion by analyzing the cooking procedure of a recipe. We developed a cooking robot to execute the proposed method, and evaluated the effectiveness of the approach by analyzing 50 recipes. More than 25 recipes could be implemented using the proposed approach.

Keywords: Recipe analysis; Cooking robot; Motion planning

1 INTRODUCTION

In recent years, the demand for cooking robots has increased rapidly to address the problem of workforce scarcity owing to the declining birthrate and aging population and to enhance the operational efficiency. Cooking robots have already been developed and commercialized in various countries [1–5]. In general, cooking robots can cook programmed recipes. However, the programming of recipes in the context of robotic arms may be challenging for personnel without considerable robotics or coding expertise. The most widely applied method to program robotic arms is the teaching playback method, in which each angle of the robot arms is taught using a device known as the teaching pendant [6].

Nevertheless, 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 those in which each angle of the robot arms is defined in a simulator through GUI [7] and hand and arm motions are transformed to robot motions using motion capture systems [8]. These methods are known as offline teaching methods because teaching can be realized even in the absence of the actual robot. In such methods, the robot motion is implemented through human motion in accordance with the recipes. However, considerable time may be required to learn to operate the software and the cost of the teaching devices may be high. As an alternative approach, researchers attempted to realize automatic teaching by reusing primitive robot motion (act) generated manually [9]. Nevertheless, the approach could not be generalized for all tasks and could not fully automate the teaching process. To realize the motion planning of cooking robot, Bollini et al. attempted to develop an approach to interpret recipes written in human languages. The robot completed the cooking task by executing primitive cooking motions following the recipe steps. However, the researchers only considered recipes that were easy to understand and involved simple sentence, and complex sentences were not considered.

To plan robot motion considering recipes, an approach must be able to correctly understand the cooking procedure of a recipe. Herein, the cooking procedure reflects the specific information, for instance, cooking operations such as mixing or pouring, movement degree, tools, and ingredients.

However, to accurately identify the cooking procedure, the following problems must be addressed. A cooking procedure may involve unnecessary information, especially in the case of recipes shared on the internet websites. In general, the background for a recipe is not relevant in the cooking procedure. Furthermore, in the Japanese language, the subjects and objects may be omitted and/or replaced. Thus, the expressions of the same cooking procedure may differ for different authors.

^{*}Correspondence: takei@hirosaki-u.ac.jp

Department of Mechanical Science and Engineering, Hirosaki University, , Hirosaki, Japan

Full list of author information is available at the end of the article

[†]Equal contributor

To address these issues, the cooking procedure must be extracted accurately, in the manner of 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 recipes.

To this end, we propose a method to analyze recipes written in Japanese characters to realize the automated teaching of stationary-type cooking robots [11, 12]. The method can realize the motion planning of cooking robots according to the robot system and cooking environment by generating the cooking procedure by analyzing general recipes. A recipe analysis algorithm was developed, and the approach was applied to an actual cooking robot system [11, 12]. Furthermore, we evaluated the proposed algorithm by considering many recipes randomly selected from websites.

2 RESEARCH APPROACH

The key objective of this study was to develop an algorithm to analyze Japanese recipes to realize automated motion planning of 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 the cooking tasks.

The target recipes were those uploaded to a recipe sharing site on the internet. The format of the recipes was required to be normalized, because the recipe information is presented differently on different recipe sharing sites.

In general, 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 is necessary to extract only the necessary information and eliminate the information irrelevant for the cooking process.

In addition, in Japanese grammar, the subject–predicate relation is determined by the particles, and the context is clear even when the sentences are interchanged. Consequently, it was necessary to standardize the positional relation of the information, as the manner of expression may differ across 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 the categorized words could be generated, the proposed word classification system would be reasonably effective.

Once the cooking procedures have been organized correctly by implementing the aforementioned processes, robot motion can be planned to simply execute

the primitive cooking operations according to the procedures.

To plan the motion of an actual robot, we built a simulation environment, which corresponded to the actual cooking environment and was of a similar scale as that of the actual environment.

Furthermore, because the cooking operation, regardless of the type of motion, depends on the parameters such as the cookware and ingredients, it was necessary for a motion to be generated identically, even if the cookware arrangement changed according to the recipe. Therefore, we attempted to generate the motion by referring to the coordinates sequentially by defining primitive motions recording the relative coordinates of the cookware and ingredients.

3 RECIPE ANALYSIS ALGORITHM

The process flow of the proposed algorithm can be divided into two stages. The first process involves the recipe analysis, as illustrated in the upper part of Fig. 1. In this stage, the necessary information to plan the robot motion is extracted from the recipe and sorted to generate the cooking procedure. The second stage corresponds to the motion planning process, as illustrated in the lower part of Fig. 1. In this stage, the robot motion is planned, including the avoidance of the obstacles, through the output data of the recipe analysis stage. The details of these stages are as follows:

1 Recipe Analysis Stage

The cooking procedure is defined to plan the motion of the robot, considering the input data, which include recipes written in Japanese characters. In this scenario, it is necessary to account for the omission of the subjects and objects and complexity of the grammatical structure. To this end, the order of robot motion is specified according to the order of the cooking elements in the decomposed recipe. Any abbreviations are complemented. These processes are realized using a generated database and sorting processes.

2 Motion Planning Stage

The robot motion is planned according to the cooking procedure defined in the recipe analysis process. Using a simulator, the actual robot motion is planned by creating paths that connect the point at the end of the current robot hand orientation to the starting point, and 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.

4 DATABASE FOR THE ALGORITHM

This section describes the three databases prepared to be used in the algorithm.

Motion Code Database

A motion code corresponds to symbol-based data containing cooking information, including nouns, verbs, degree of movement, and unique identifier. The definitions of the elements in the motion codes are listed in Table 1. For example, in the sample motion code “E[0] T[2]”, “E[0]” indicates large stationary cookware, such as a heater, with “0” being the identifier for the cookware in the motion code database, herein, the E code database. In general, to realize the word-by-word classification of recipes, Mori et al. [13] developed a corpus of tagged terms describing the motion and state of the cooking process and ingredients. Although this classification can identify the state and motion of the ingredients, a finer classification is required to plan automated robot motion. To this end, in the proposed approach, eight subdivisions were generated with reference to these classification tables, labeled as E, U, T, A, F, V, C, and S. Each type of database contained the cooking process information. 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. Sample entries of the E code database are presented in Table 2.

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

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	...
...	...	...	...	...	...	...

Object Database

The object database is a variable and dynamic database used to plan the robot motion. If a motion code indicates an "object", such as ingredients (F) or cookware (E or T), the object data related to the object is added to the object database. The information of the state of the object, such as the current position, parent node of the robot hand, and previous movement, are 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 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	...
...	...	...	...	...	...	...

Motion Library

In the operation database, the dependency relationships with other codes, according to the ID of the operation code V, and several basic operations of the robot arm are registered, as indicated in Table 4.

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 relationship with E (container), cooking utensil (T), and seasonings (A), and the basic operation is "mix clockwise".

This library is used to determine the dependencies of a code or necessary information in response to the V code. Furthermore, this library is used to generate the motion of an actual robot. The primitive motion of the robot corresponding to the motion content of the V code 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.

5 RECIPE ANALYSIS

This section describes the algorithm used to obtain the cooking procedures in the recipe analysis stage by analyzing the recipes written in Japanese letters.

Table 4 Motion library structure: The ID of the V code indicates the motion; the dependent code type indicates the dependent code type, and the base motion indicates the time series data of the actual robot motion.

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)	...
...	...	...	...

5.1 Prerequisite

The following assumptions were made:

- The input is a Japanese recipe in the recipe format (Fig. 2).
- The input recipe consists of general sentences that satisfy the following conditions.
 - 1 At least certain verbs and nouns are included in each sentence.
 - 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 the conditions, such as status of the ingredients.
 - 4 No spelling or grammatical errors are present in any sentence.

5.2 Conversion of Recipes to Recipe Data

Because the expression style of the recipes may be different depending on the recipe sharing site (such as Cookpad or Allrecipes.com), we define a recipe data format, as shown in Fig. 2. The data in this format is termed as the recipe data, which are used as the target data for the analysis. The recipe extracted from the website is transformed to the recipe data.

(recipe name)	
(description)	
[ingredients]	
(ingredients)	(quantity)
[steps]	
Steps....	

Figure 2 Recipe Data Format

5.3 Extraction of Motion Code

To organize the necessary information from the recipe data, words such as those pertaining to the cookware,

ingredients, and verbs related to cooking are extracted and converted into the motion codes in the order of occurrence.

To implement this process, we use McCab [14], an open source morphological analysis engine. The software converts the nouns, verbs, adjectives, and adverbs extracted from a sentence into motion codes by comparing the results with the motion code database.

5.4 Sorting and Insertion of Motion Codes

The motion codes are treated as array data and 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 of the array of sorted motion code.
- 4 If V that does not depend on any other code is encountered, move V to the last cell of the array of the sorted motion code.
- 5 If V that depends on any other code is encountered, move V to the last cell of the array of the sorted motion code. Subsequently, move the codes that V is dependent on to the last cells of the sorted array.
- 6 If A, F, T, and S that do not correspond to V are encountered, move the code to the last cell of the array of the sorted motion code after complementing V using the motion code database and motion library.
- 7 Otherwise, move the code to the last cell of the array of the sorted motion code.

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

Two types of cooking operations (V) exist: The operations (V-independent) that can be executed independent of the ingredient and cookware (T), and the operations (V-dependent) that are 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. Among the remaining codes, the A, F, T, and S codes need to be complemented by the V code, as the associated verbs may be omitted.

Finally, the remaining operation codes are moved to the end of the sorted operation code array as is.

This process can extract the unnecessary information and minimize the grammatical differences pertaining to different writers. In addition, this process can simplify the subsequent robot motion planning phase.

6 MOTION PLANNING

This section describes the formulation of the operation procedure by using the output motion codes.

The operation procedure involves two functions (as described in subsection 6.1) of identifying the cooking procedure and complementing the cooking operation 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 stage in Fig.1.

6.1 Planning Robot Motion

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

- **SET Arg¹(Ingredient or Cookware)**
Set the target points of the arm to Arg¹
- **EXECUTE Arg¹(Cooking Operation)**
Execute Arg¹

The SET function is 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, such as heater and gripper, corresponding to the robot hand, may be assigned. The names and actual processing contents are defined in the motion code database and motion library (as described in Section 4).

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

6.2 Simulator to Train the Robot Motion

The actual motion of the robot can be generated by implementing offline teaching through the 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 at the tip of the robot hand. Each point can be derived considering the operation procedure. Various methods have been proposed to identify the path connecting the starting point and target point. In this study, the rapidly exploring random tree strategy was employed [16]. In this approach, when the obtained paths are the paths that involve a collision with the obstacles, the motions are generated with passage points placed around the obstacle. After the motion is generated, it is converted into an executable code, which corresponds to the final output data that can be executed by the robot. The execution code is illustrated in Fig. Fig.6.

7 EXPERIMENT

The proposed method was evaluated considering general recipes written in Japanese characters to verify if motion planning of a cooking robot could be realized.

7.1 Experimental Environment

The experimental environment had the following characteristics:

Target Robot

The VP-6242 robot from DensoWave was applied. To handle several cookware, the robot hand used two robot grippers from Makerobot as a single

Figure 5 Cooking Robot Simulator

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

Figure 6 Types of execution code

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

Cooking Environment

We arranged the necessary amount of ingredients for the recipe and the cookware to be placed around the target robot. In addition, a similar cooking environment was constructed on the simulation environment.

Input Data

For the recipe data, we considered a pancake recipe that included the cooking motions of adding ingredients, mixing, scooping, pouring, and flipping. The recipe data is presented in Fig. 7.

Database

The database was filled with the keywords and behaviors involved in the input recipe. Seven primitive motions, namely, grabbing, releasing, moving, rotating, adding, mixing, and flipping, were defined and input to the motion library.

7.2 Result

1 Recipe Analysis

The results of the recipe analysis using the pro-

posed approach are shown in Fig. 8. According to the pancake cooking procedure, the words related to cooking were extracted as the motion codes without unnecessary information, and accurate motion codes were generated.

2 Action Generation

The robot motion was planned using the motion code in the simulator. The operating procedure is not presented herein, 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

After the output execution codes were sent to the actual robot, the robot performed a series of cooking operations following the cooking procedure. A series of pictures of the robot making pancakes in the experiment is shown in Fig. 10. However, at certain instances, the robot could not pour the batter in a certain position or turn the pancake over.

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!

Figure 7 Pancakes Recipe Data

8 EVALUATION OF THE RECIPE ANALYSIS APPROACH

The aim of the aforementioned experiment was to verify if the proposed approach could plan robot motion considering a recipe presented in Japanese characters. Consequently, we considered a relatively easy recipe consisting only of the words known to be registered in the motion code database.

Subsequently, a validation experiment was conducted to determine the scope of application of the recipe analysis method.

8.1 Preparation

For this verification, it was necessary to increase the data in each database. In particular, the motion code

【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

Figure 8 Recipe Analysis Results

```
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
move,-116.686,22.220,106.457,-0.303,62.746,7.655,30
```

Figure 9 Sample output executable code

Figure 10 Actual robot making pancakes

database is the most critical database in the analysis process and indicates the number of 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, because this scenario is not realistic, we attempt to diversify the vocabulary for two motion code types, A and F. These data were obtained by extracting approximately 1,000 recipes by randomly searching web pages on Cookpad [17]—a recipe sharing site on the internet—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 and 24 verbs.

For other datasets, we entered the minimum required data. For example, we entered a total of 24 verbs, such as “put”, “mix”, and “bake”.

Information such as that of the object shape and heating availability, which was not particularly relevant to the recipe analysis process, was omitted at the time of input, because the verification experiment involved steps only until the motion code conversion. It was considered that if the motion code conversion is accurate, the subsequent processing is uniquely determined.

8.2 Input Recipe data

In the experiment, 50 data points were randomly extracted from a search for “side dish” in Cookpad [17] 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 assumptions.

- Sentences not including verbs and nouns or cooking information
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
Example) 入れる→入る
Meaning) put → pu

After exclusion, 33 data remained, which were used in the experiment.

8.3 Evaluation Method

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

8.4 Result

It was observed that 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. 11.

The percentage of the causes of failure of the motion code conversion is shown in Fig. 12. The inability to address the unique and ambiguous expressions was the most frequently occurring cause (40%), followed by the lack of content in the motion code database (33%), and redundant content processing and word segmentation failures in the recipe data (13%)

<p>本日の晩ご飯</p> <p>[ingredients]</p> <p>大根 20cm</p> <p>ゆで卵 3個</p> <p>こんにゃく 1枚</p> <p>ごぼう天 3本</p> <p>厚揚げ 3個</p> <p>里いも 1/3袋</p> <p>顆粒出し 1/2袋</p> <p>酒 ひとまわし</p> <p>醤油麹 小さじ2</p> <p>醤油 色づくぐらい</p> <p>[steps]</p> <p>野菜類を農薬除去液にくぐらせ流す。</p> <p>大根も同じく。ゆで卵を作る。</p> <p>圧力鍋に入るだけ先に煮る。</p> <p>抗酸化だしとミネラルを一滴ずつ入れる。</p> <p>圧力が取れたら、他の材料も入れ煮る。</p> <p>ガンガン煮る。出来上がり。</p>	<p>E[2] 鍋</p> <p>V[2] 入れる</p> <p> ^LF[4360] 茸</p> <p> ^LF[1826] にんにく</p> <p> ^LF[4132] 生姜</p> <p> ^LA[441] ごま油</p> <p>V[5] 焼く</p> <p>-----</p> <p>V[2] 入れる</p> <p> ^LA[1963] 水</p> <p>V[17] 沸騰</p> <p>-----</p> <p>V[2] 入れる</p> <p> ^LF[2675] もやし</p> <p> ^LF[4502] 豆</p> <p> ^LF[337] えび</p> <p> ^LA[1876] 醤油</p> <p> ^LA[1766] 酒</p> <p> ^LA[585] スープ</p> <p> ^LF[872] コーン</p> <p>V[17] 沸騰</p> <p>-----</p>
Recipe Data	Motion Code

Figure 11 Example of a successful motion code conversion

9 DISCUSSION

Although the recipes used in the experiments involved problematic sentences, such as those involving the omission of subjects and objects and unspecified cookware, the robot motion planning could be satisfactorily realized using the proposed method. In addition, by using the motion code and motion library, the method could generate paths leading to individual motions. Furthermore, because the approach could define robot motion that could be executed on 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 generation; 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 input in the database cannot be evaluated. In addition, the motion library can generate motion independent of the position and type of the tools; however, the library must be able to take into account all possible conditions, and malfunctions may occur in the case of improper inputs. Therefore, to apply the method in a more generalized manner, 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. To execute the simulation on an actual robot, considerable information processing is expected to be realized in advance, such as the adjustment of the cookware and other aspects, to unify the associated layout with that in the simulation.

In the experiments, in certain cases, the robot could not effectively handle the pancake batter. 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 reality. Consequently, it may be necessary to try to generate motion

considering the actual environment. Moreover, the process can be improved by adding an external field sensor to provide real-time feedback control during execution. From the results of the validation experiments, it is possible to analyze more than half of the recipes, which is randomly selected from the general recipes at the web sites, by using this method. The following text discusses the causes of failure in the verification experiment.

1 Inability to address unique and ambiguous expressions

Depending on the manner of recipe expression, the effectiveness of the 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 well. 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良い」

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

2 Insufficient Content in the Operation Code Database

When the recipe data involve cookware or ingredients that are not registered in the database, the reordering process cannot be performed well because the sorting of the words fails. In addition, there exist many notations, such as those of 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 The 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 not related to the recipe cannot be processed in the approach.

(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 in the future, it is necessary to diversify the data of the motion code database and to implement modifications to address unique and ambiguous expressions. This result was expected to a certain extent before the experiment, and the experiment results confirmed this prediction. To apply the approach to recipes on the internet, which involve more colloquial expressions and aspects requiring more types of databases, it is necessary to implement the coding of casual phrases, especially because the writers often employ contractions and verb paraphrasing. In this regard, it is also necessary to revise the sorting process in the future, such as by enhancing the adverbial phrases that can be used and adding judgmental processing to modify phrases with clauses.

In certain failure cases, the redundant recipe content could not be removed. Such redundancies are often involved in the recipes posted by ordinary Internet users, who separate words with arrow symbols or write comments between a cooking procedure. The proposed algorithm was developed to eliminate emotional expressions and impressions. Because the ratio of this cause of failure is lower than that of the other causes, the algorithm is considered to have succeeded in removing a certain amount of redundancy. 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 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. In this case, it is necessary to add more cooking and ingredient terms to the MeCab [14] database.

The applicability of the proposed method is limited owing to the considered assumptions, and 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 process is implemented. It is considered that the method can be made more generalized by correcting mistakes in the authors' written language or by referring to other recipes when the specific content of the recipe is not indicated.

It must be noted that a multi-axis industrial manipulator, which is used in many commercial machines, was adopted for 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.

10 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.

Using a simple pancake recipe, we performed experiments on an actual machine and confirmed that the correct motion procedure according to 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 the recipes could be converted correctly without any modification.

Using this database, the algorithm could analyze approximately 60% of the 33 recipes. Nearly 70% of the cause for failure pertained to the lack of unique expressions and proper nouns in the database. The results indicated that obtaining the names of ingredients and seasonings randomly did not increase the performance notably. 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 enhance the database, it must be made more effective with respect to the prioritization of duplicate verbs in the input and the relevant comparison 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 that enabling keyword classification for similar expressions.

In future studies, we aim 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.

References

1. Spyce. <https://www.spyce.com/>. Accessed 20 August 2020.
2. Jing Dong X Future Restaurant. <https://prtimes.jp/main/html/rd/p/000000049.000034149.html>. Accessed 12 September 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 January 2020.
6. Kusuda, Yagi. A Practical Guide to Introducing Industrial Robots in Illustration, Nikkan Kogyo Shimbun, 1999[Translated from Japanese.]
7. Yuta Sugiura, Daisuke Sakamoto, Ai Withana, Masahiko Inami, Takeo Igarashi. 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 Fukuoka, Yukinobu Hoshino. 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 Shinkawa, Tatsuhiko Sakaguchi, Keiichi Shirase. 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. Mario Bollini, Stefanie Tellex, Tyler Thompson, Nicholas Roy, Daniela Rus. Interpreting and Executing Recipes with a Cooking Robot. Experimental Robotics. Experimental Robotics, Springer Tracts in Advanced Robotics, vol 88, pp481-495, 2013.
11. Masahiro Inagawa, Toshinobu Takei, Etsujiro Imanishi. Interpreting and Motion Generation for a Cooking Robot. ROBOMECH2019 The Robotics and Mechatronics Conference 2019, 1P2-C11. 2019.
12. Masahiro Inagawa, Toshinobu Takei, Etsujiro Imanishi. Japanese Recipe Interpretation for Motion Process Generation of Cooking Robot. 2020 IEEE/SICE International Symposium on System Integration, Paper We2E.5. 2020.
13. Shinsuke Mori, Yoko Yamashawa, Tetsuro Sasada, Hirokuni Maeda. 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 January 2020.
15. Unity. <https://unity.com/> Accessed 20 January 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. Cookpad. <https://cookpad.com>. Accessed 21 March 2020.

Figures

Figure 1

Recipe analysis algorithm for the cooking robot. The algorithm involves two process indicated by the two flowcharts surrounded by dotted lines: The top and bottom processes correspond to the recipe analysis and motion planning, respectively.

(recipe name)

(description)

[ingredients]

(ingredients) (quantity)

[steps]

Steps....

Figure 2

Recipe Data Format

Figure 3

Example of the sorting process

Figure 5

Cooking Robot Simulator

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

Figure 6

Types of execution code

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!

Figure 7

Pancakes Recipe Data

【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

Figure 8

Recipe Analysis Results


```
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
move -116.686 22 220 106 457 -0.303 62 746 7 655 30
```

Figure 9

Sample output executable code

Figure 10

Actual robot making pancakes

本日の晩ご飯

[ingredients]

大根 20cm

ゆで卵 3個

こんにゃく 1枚

ごぼう天 3本

厚揚げ 3個

里いも 1/3袋

顆粒出し 1/2袋

酒ひとまわし

醤油麹 小さじ2

醤油 色づくぐらい

[steps]

野菜類を農薬除去液にくぐらせ流す。

大根も同じく。ゆで卵を作る。

圧力鍋に入るだけ先に煮る。

抗酸化だしとミネラルを一滴ずつ入れる。

圧力が取れたら、他の材料も入れ煮る。

ガンガン煮る。出来上がり。

Recipe Data

E[2] 鍋

V[2] 入れる

└F[4360] 茸

└F[1826] にんにく

└F[4132] 生姜

└A[441] ごま油

V[5] 焼く

V[2] 入れる

└A[1963] 水

V[17] 沸騰

V[2] 入れる

└F[2675] もやし

└F[4502] 豆

└F[337] えび

└A[1876] 醤油

└A[1766] 酒

└A[585] スープ

└F[872] コーン

V[17] 沸騰

Motion Code

Figure 11

Example of a successful motion code conversion

Figure 12

Percentage of causes of failure in motion code conversion