Intention-Sensing Recipe Guidance via User Accessing Objects

Atsushi HASHIMOTO\textsuperscript{1}, Jin INOUE\textsuperscript{1}, Takuya FUNATOMI\textsuperscript{2}, and Michihiko MINOH\textsuperscript{1}

\textsuperscript{1}Kyoto University
\textsuperscript{2}Nara Institute of Science and Technology,

May 18, 2016

Sensing intention of a user’s next action is a necessary function for the systems assisting human physical activity. In this paper, we investigate a strategy for recipe guidance systems that can predict a sub-task that the user intends to do next, in a task of cooking a dish. For this purpose, we focus on user’s access to objects, namely actions of touching and releasing objects. Touching might indicate the start of the next sub-task, and releasing the end. The main difficulty lies in the fact that, firstly, humans may move objects just because they are in the way, and secondly, humans may use cooking tools that are unforeseen for the assistive systems. In these cases, the accessed object should not indicate the next sub-task. As our contribution, the proposed method tracks the progress of a task based on the history of object access. This enables us to eliminate any accesses to objects that are out of context. Simultaneously, the method predicts the next sub-task based on a combination of the progress and materials rather than tools and materials. In the experiments, we developed a guidance system that runs as a web service, and observed real cooking activities navigated by the system. The Wizard of OZ method is utilized to simulate a system that detects accesses to objects. As a result, we achieved 73.6\% of accuracy on displaying information selection. This result supports availability of “access to objects” to achieve an intention sensing systems in a practical situation.

1 Introduction

Recent progress in mobile and multi-sensing technologies has increased the opportunity for computers to assist human’s physical activity in the real world. Different from the human-computer interaction on a deskwork, the people working on physical tasks do not constantly look at a computer interface, but instead concentrate on their own physical task. “Intention sensing” is a necessary function in the context of assisting physical tasks.

*ahasimoto@mm.media.kyoto-u.ac.jp*
A typical example is the exosuits, one type of which aims to aid in nursing care. In the implementation of exosuits, myoelectric signals play an important role in sensing wearer’s intention of the next motion. This enables to control motors automatically without any interface operations, which greatly helps users concentrating on the nursing care tasks.

We aim to achieve the function of “intention-sensing” for more complicated tasks, such as cooking activity (Fig. 1). A cooking task is organized by many sub-tasks, e.g. cutting onions, tearing lettuce, and dressing the mixture of them. The order for executing the sub-tasks is not strictly defined in nature (Hamada, Okabe, Ide, Sakai, & Tanaka, 2005), and can be described as a workflow in which one sub-task corresponds to one node (Fig. 2). As the workflow description is often used for many kinds of physical tasks, the scenario of recipe guidance can be applied to those other tasks. More favorably, small mistakes in cooking task are hardly critical. This enhances human natural behavior. In this sense, cooking task meets to our aim of achieving intention-sensing in a practical situation.

The problem of intention sensing in the cooking context can be defined as estima-
tion of the next sub-task on the workflow graph. To solve this problem, we use “object access”, namely actions of touching and releasing objects, as a substitute for the myoelectric signal. The worker naturally accesses various objects while cooking activity. Accessed objects are good clues for predicting the next sub-task intended by the user. Predicting the next sub-task enables us to help a user by displaying multimedia information along with what the user will do, and will free the user from frequent interface operations that interfere with the user’s concentration.

On the other hand, different from myoelectric signal, an object access does not always correspond to the user’s intention. A user may grab objects just to set it aside. The user may also use an unforeseen tool to execute a sub-task. In this sense, it is a challenge to predict the next sub-task in a practical situation even based on access to objects. The contribution of this paper is the proposal of an algorithm that deal with the above two difficulties; we call them “deceptive access” and “unforeseen tools”, respectively. The algorithm tracks the progress of task by the history of object access. This enables us to eliminate deceptive accesses, which will be out of context. Simultaneously, the method predict the next sub-task based on a combination of progress and materials rather than tools and materials. Even when tools used in a sub-task are unforeseen, materials are enough informative under the known progress.

2 Related work

Better eating habits are important not only for promoting people’s health and well being but also for reducing the cost of health care in our society. To guide people toward better habits, it is important to assist them in preparing healthy, delicious and economical dishes. In recent years, many such recipes have become available on the Web. Video and still images facilitate users’ understanding of various cooking techniques, and voice announce makes it easier for users not to miss necessary steps. Such multimedia content has a trade-off problem; namely, the more multimedia contents a recipe contains, the more operations are required to search and play the contents.

To improve the accessibility to the multimedia contents, many researchers have proposed various interfaces (Ju, Hurwitz, Judd, & Lee, 2001; Bradbury, Shell, & Knowles, 2003; Hamada et al., 2005; Nintendo, 2006; Miyawaki & Sano, 2008; Uriu et al., 2012; Matsushima, Funabiki, Zhang, Nakanishi, & Watanabe, 2013). Similar interfaces have been also proposed for assembling tasks (Zauner, Haller, Brandl, & Hartmann, 2003; Tang, Owen, Biocca, & Mou, 2003; Yuan, Ong, & Nee, 2008; Siltanen et al., 2007). The main difference between cooking and assembling tasks is the availability of radio-frequency identification (RFID) tags and visual markers; they are unavailable for food-stuffs. Nonetheless we should take the assembling task into consideration because it can also be represented by the workflow graph shown in Fig. 2.

Many systems were developed in the early 2000 (Ju et al., 2001; Nintendo, 2006; Zauner et al., 2003; Tang et al., 2003). Those systems recognize the progress of a cooking/assembling task by tags, markers or simple user operations; however, the recipes in those systems are described as fully ordered sub-tasks, and the workers must obey them strictly. In such a situation, the next sub-task is always uniquely defined, and there is no space for workers to arrange the scenario according to their own intentions.
Hamada et al. (Hamada et al., 2005) pointed out that the flexibility of a cooking recipe can be represented by a workflow graph shown in Fig. 2. Their system intends to assist users in preparing a hot meal for the family. For this purpose, the system recommends the full-order of sub-tasks by an online scheduling algorithm so as to cause the user to finish up dishes all together. The partial-order defined by the workflow graph is an important condition for the online scheduling algorithm. This work focused more on the scheduling algorithm, and less on the interface. This system helps users a great deal, especially when they are unfamiliar with the cooking procedures. On the other hand, the absence of intention-sensing functionality will always cost a user operating the system’s interface whenever the user wanders from the plan recommended by the system. This will lead to lack of experienced user’s motivation even though the scheduling is a helpful function. Our purpose is to maintain the user’s initiative by sensing his/her intention while providing various kinds of assistance by computer, including the scheduling assist.

There are some studies taking the user’s intention into consideration. Yuan et al. developed an interface (Yuan et al., 2008) that enables users to signal the end of sub-task execution and to choose the next sub-task by themselves. Similarly, some cooking-assistive systems use special devices to access information freely while executing a task (Bradbury et al., 2003; Matsushima et al., 2013). Clearly, interface operation can frequently interrupt work, and interfere with the user’s concentration.

Instead of explicit interface tools, Miyawaki et al. proposed a system that uses real objects as implicit interfaces for choosing sub-tasks (Miyawaki & Sano, 2008). Their system achieved automatic recipe navigation without any explicit interfaces. To achieve full automation, they preliminarily form a correspondence between sub-tasks and the objects on the cooking counter, which are mainly containers. The system identifies the next task according to which container the user accessed.

This approach is similar to ours; however there is an essential difference. They avoid the ambiguity of the next sub-task by assigning unique objects to each sub-task. In nature, no unique correspondences are guaranteed between sub-tasks and objects. The user will easily forget the unnatural correspondences, and this approach will collapse when the user uses an unforeseen object. Our method also relies on the correspondence between sub-tasks and objects, but the correspondence is given by the nature of each sub-task so that the user does not need to memorize the assignment. In this setting, the objects assigned to sub-tasks are no longer unique. Instead of relying on the unique correspondences, we proposed an algorithm that focuses on identifying the intended sub-task from contextual information and the combination of accessed objects.

Schneider simulated humans execution of cooking task on a dynamic Bayesian network (DBN) (Schneider, 2009). In the DBN, a state corresponds to progress in cooking, and a transition corresponds to a sub-task that the user has executed. In this problem setting, the system aims to find out what recipe the user is working on rather than what sub-task the user will do next. Even the system aims recipe identification, the model has a capability of predicting the next sub-task substantively.

In their work, instead of observing real cooking, they randomly generated observation data as a sequence of symbols in the simulation. The DBN was also generated randomly and no human factors are counted. Moreover, the number of possible states
of the DBN explodes combinatorially if the system allows the user to execute sub-tasks in a recipe flexibly (see Appendix B). We deal with the flexibility of the recipe, and pay more attention to the real cooking activity. The heuristics of our algorithm reveals the sparseness of human state transition behavior even in the combinatorially exploded number of states.

3 Sensing intention via access to objects

3.1 Workflow graph representation of a recipe

Before discussing the detail of our approach, we define the workflow representation of the graph more strictly. Let $G(V, E)$ be a workflow graph shown in Fig. 2, where $v \in V$ corresponds to a sub-task, and $e \in E$ is a definition of order relationship between two sub-tasks. The edge is given in the following manner; we express an order relationship between two sub-tasks $v_1$ and $v_2$ by operator $<$. Namely, $v_1 < v_2$ means that $v_1$ must be completed before starting $v_2$. To avoid any redundant edges to represent the partial order definition in $G$, we add directed edge $[v_1, v_2]$ to $E$ if and only if $v_1 < v_2$ and there are no other sub-task $v$ such that $v_1 < v$ and $v < v_2$.

For example, the graph shown in Fig. 2 consists of sub-tasks $V = \{a, b, c, d, e\}$, and order definitions $E = \{[a, b], [b, d], [c, d], [d, e]\}$. The graph has two paths $a < b < d < e$ and $c < d < e$. Each path of sub-tasks on the flow tracks one material, and each conjunction of paths corresponds to a sub-task of mixing the materials. As $G$ is intended to represent the partial order relationship of a task, there are no order definitions between sub-tasks $a$ and $c$. In other words, the users have leeway to arrange the order of these sub-tasks.

To utilize object access to sense user’s intention, we associate object labels with each sub-task in $V$. Let $O_v$ be the labels linked to sub-task $v$. We also maintain the division between materials and tools by denoting them respectively by $O^m_v$ and $O^t_v$ ($O^m_v \cup O^t_v = O_v$, and $O^m_v \cap O^t_v = \emptyset$). In cooking activity, it is not obvious whether a seasoning belongs to $O^m_v$ or $O^t_v$. To rule out this ambiguity, we define that an object is a material if and only if it is processed alone in any of $v$. Otherwise it is a tool. For example, oil is material if $G$ has a sub-task $v$ in which the oil is solely heated up, but not when it is just added to other materials. This definition avoids an appearance of $v$ with no materials in linked object labels.

From the nature of workflow graph, the materials are further categorized into solo material and mixture of materials. The labels of the tools and solo materials can be preliminary assigned to $v$ by hand. The labels are succeeded until they are mixed. For example, we set $O^m_d = \{A, B\}$ at sub-task $d$ in Fig. 2 because $A$ and $B$ are involved to the process one-by-one at $d$. After a sub-task that mixes two or more materials, the labels of mixed materials are bound as one label, and treated as a mixture of materials. In Fig. 2, after the process at $d$, $A$ and $B$ are mixed and treated as a material represented by $\{A, B\}$, and we set $O^m_e = \{A, B\}$. In this manner, $O^m_v$ is obtained automatically from the assigned raw materials $A$ at sub-task $a$, $B$ at $c$, and the structure of $G$. A more precise definition of the algorithm to assign $O^m_v$ for each sub-task is given in A.

A recipe can be represented by various granularities of sub-tasks $V$. For example, a
writer of recipe can represent a direction for tomato puree by just one sub-task that says "puree tomato", or several sub-tasks of "remove the stems of tomatoes", "boil them", "put them into a pot of cold water", "peel the skin", "dice them", and "puree them in a food processor". The granularity has an influence on usability of the system. The level of detail given is generally up to the writer. For beginners, more detail is preferred; for experienced cooks, a rougher explanation may be better.

Basically, more finely the sub-tasks are divided, more detailed information the system can display for users. A fine division may be kind for the users who are unfamiliar to the recipe. For those familiar with it, too much information does not annoy them when their intention is sensed by the system accurately and the displayed information is switched automatically. Because our goal is the system with intention sensing, we suppose finely divided sub-tasks that helps users cook unfamiliar recipe, and does not annoy them while cooking familiar recipes. To make the recipe more compatible with our strategy of object access, we divide a recipe into sub-tasks so that each sub-task has the smallest $O_v$. $O_v$ is the smallest when a sub-task includes no processes that use different tools. Thus, the smallest $O_v$ is obtained easily by dividing any sub-tasks that include two processes using different tools.

3.2 Basic problems with sensing intention from object access

To achieve the functionality of sensing intention, we use object access. Let $O^h$ be a set of objects in hand. Intention-sensing via object access is basically considered as a problem of finding $v$ intended as the next sub-task, based on comparison of $O^h$ with $O_v$ for all $v \in V$. In practice, there are some problems that should be taken into account. Firstly, when sub-tasks $u$ and $v$ have the same object labels ($O_u = O_v$) and matched to $O^h$, we must consider any external evidences to choose either of $u$ and $v$ that is more likely to be performed next. For this problem, we choose the one that is on the same path with previously performed sub-task. Secondly, $O^h$ and $O_v$ are not always reliable due to deceptive access and use of unforeseen tools.

Deceptive access is an access that is not related to the user’s intention of the next action. It appears typically when the user dislocates certain objects. This is a common act in many kinds of tasks, and those acts make a naive rule-based system, such as (Miyawaki & Sano, 2008), work improperly.

Unforeseen tools are not negligible in practical situations because it is obviously impossible to list up all possible tools for $v$ preliminary. When an unforeseen tool is used as a substitute for a tool in $O_v$, $O_v$ and $O_h$ become different. Hence, the system must accept a certain difference of tools in $O_v$ and $O_h$.

Against the deceptive access, we maintain the progress of the task on $G$, and reject any access that is not suitable to the situation. To deal with the unforeseen tools, we designed a score function that is capable of handling a certain amount of mismatch between $O_v$ and the actually accessed objects.

3.3 Enhancing robustness against deceptive access

To eliminate the deceptive accesses from the intention-sensing process, it is helpful to track the progress on a workflow graph $G(V, E)$. The progress is described by a set of
completed sub-tasks. We refer to the set as $V^- (\subseteq V)$. $V^-$ is obtained by detecting every completion of $v \in V$ through the observation. The sub-tasks at the ready on execution are obtained as the sub-tasks whose ancestors are all in $V^-$, where $u$ is an ancestor of $v$ if $u < v$. We refer to such sub-tasks as $V^+$ (Fig. 3).

A simple idea for eliminating the deceptive accesses is to ignore any accesses when the accessed object is not listed in $\bigcup_{v \in V^+} O_v$. This strategy will eliminate deceptive accesses effectively, but is practically too naive against inaccurate $V^+$. When a single mistake has occurred in completion detection, $V^-$ goes inaccurate and thereby $V^+$ will be disturbed, too.

To discuss the situation that the accurate $V^-$ is different from that of the system believes, let us consider a set of sub-tasks $\hat{V}^-$ that have been estimated to be completed by the system, and distinguish it from $V^-$, which is an accurate set of completed sub-tasks. Similarly, $\hat{V}^+$ is the set induced by $\hat{V}^-$. Note that, for each path in $G$, there is only one sub-task that is on the path and belongs to $V^-$. We refer to $\hat{v}$ as a sub-task that is in $\hat{V}^+$ and on the same path with $v \in V^+$. When $\hat{v}$ and $v$ are different sub-tasks, the accesses to objects in $O_v$ are ignored, and the system fails in displaying the instruction for $v$.

For the purpose of enhancing robustness against failure in $\hat{V}^-$, we extend $\hat{V}^+$ into two directions, and get modified contexts $\hat{V}^-$ and $\hat{V}^+$. $\hat{V}^-$ consists of nodes in the forward direction of directed edges in $G$ from those in $\hat{V}^+$, and $\hat{V}^+$ in the backward direction (Fig. 4). $\hat{V}^- \cup \hat{V}^+ \cup \hat{V}^-$ will cover more nodes in $V^+$, which is the accurate context, than $\hat{V}^-$ alone. Therefore, involving accesses related to $\hat{V}^-$ and $\hat{V}^+$ in addition to those related to $V^+$ will reduce the risk of eliminating access to $v \in V^+$.

It is important to consider how to expand $\hat{V}^+$. Wider the context is, more loss of contextual information occurs. On the other hand, too narrow expansion leads the same result with the case of using $V^+$. On the other hand, narrower the context is, more naive against the failure. For deciding our strategy, we focused on the difference of
risks between two types of failures on \( \hat{v} \in \hat{V}^- \): the case of \( \hat{v} < v \) and \( v < \hat{v} \), where we
call the two types of the cases respectively left-unattended error and overrunning error.

Once \( \hat{v} \in \hat{V}^+ \) has fallen in the condition of \( \hat{v} < v \), the system keeps waiting the
accesses to \( O_v \). However, in this case, \( \hat{v} \) is a completed sub-task and objects in \( O_{\hat{v}} \) are
hardly accessed. In this situation, the system will not recover from the left-unattended error automatically. In contrast, when \( v < \hat{v} \), the user proceed the task and, in the
meanwhile, accesses to the objects in \( O_{\hat{v}} \) to work on \( \hat{v} \). In this case, the system surely
has a chance to recover from the overrunning error.

To avoid the left-unattended error with a certainty, we prepare a completion detector for \( \hat{V}^- \) so that it achieves a high recall rate, thereby it hardly misses the completion.
This strategy leads many overrunning errors. To reduce the bad effect from the over-
running errors, we expand the context \( \hat{V}^- \) in only one sub-task (\( \hat{V}^- \)). Conversely, we
expand the context widely in the backward direction so that \( v \) tends to lie in \( \hat{V}^- \) while \( v < \hat{v} \) (Fig. 4).

We provide a mathematically rigorous description of the expansion shown in Fig.
4 as follows. First, \( \hat{V}^+ \) is obtained as \( \{v | u, v \in E \land u \in \hat{V}^+ \} \). Second, we add \( u \) to
\( \hat{V}^- \) if and only if \( u < v \) for any \( v \in \hat{V}^+ \) and there is at most one conjunction on the
path from \( u \) to \( v \). This backward expansion accepts accesses to both materials mixed in
the sub-task at the conjunction, and mixture of the materials produced by the sub-task.
Adding \( \hat{V}^- \) and \( \hat{V}^+ \) obtained in this manner to \( \hat{V}^+ \) loses some contextual restriction,
but is more robust against corrupting \( \hat{V}^- \), which may contain errors.

### 3.4 Dealing with unforeseen tools

Different from the system by Miyawaki (Miyawaki & Sano, 2008), our system does
not force users to obey any one-to-one correspondences between tools and sub-tasks.
Instead, we list up alternative tools in \( O_v \) for each \( v \). Since the alternative tools are not
used together, we prepare \( O_v \) for each alternative tool. Namely, \( O_v = \{O_i^v | 0 \leq i < N_v \} \)
where \( N_v \) is the number of alternative combination of tools. For example, if a sub-task
is "peel the potato skin," then the tool can be a knife or a peeler. In that case, \( O_v \) consists of
\( O_0^v = \{"knife","potato\}" and \( O_1^v = \{"peeler","potato\}".

When any of \( O_i^v \) is equal to \( O_0^v \), it is easy to sense the intention; however, as
discussed above, \( O_0^v \) and \( O_1^v \) are often not equal. The user may execute sub-task \( v \) with an
unforeseen tool, or keep holding some unnecessary tools during a sub-task. In those
cases, \( O_0^v \) differs from any of \( O_i^v \).

To match \( O_i^v \) to \( O_0^v \) while allowing such difference in tools, we designed a heuristic
scoring function \( L(v; O) \), and select the sub-task of the highest score. The score for \( v \) is
calculated as

\[
L(v; O^h) = \begin{cases} 
0 & v \notin \hat{V}^- \cup \hat{V}^+ \cup \hat{V}^- \\
\max_{\hat{O}_i^v} L(O_i^v; O^h) & \text{otherwise} 
\end{cases} 
\]

\[
L(O_i^v; O^h) = \sum_{o \in O_i^v} \sigma(o) \mathbb{I}(o \in O^h) - s|O^h \setminus O_i^v|, 
\]

where \( \sigma(o) \) is an elemental score assigned to each object in \( O_i^v \). We assign a larger
score to materials in $O^i$ and a smaller one to tools because the material is a dominant factor in sensing the intention. $\mathbb{I}(o \in O)$ is an indicator function that returns 1 if $o \in O$ is true, otherwise 0. $\epsilon$ is a penalty score for extra objects, and $|O|$ is the number of elements in set $O$.

We show our setting of elemental score $\sigma(o)$ and penalty score $\epsilon$ in Table 1. These values were given in the following heuristic manner. Basically, the score for tools should be used only to differentiate the sub-tasks treating the same material. An exception is the case where only tools are accessed, but no materials. A typical example is seasonings, which are often sprinkled on foodstuffs, and the foodstuffs are not accessed at all. The seasonings will be the dominant factor in such cases. Hence, we set score $\sigma(o)$ a large value for the tools if $O^i$ contains no materials.

### 3.5 Overall algorithm

Here, we present the details of the algorithm. The overall algorithm is given as Alg. 1. In the algorithm, $v'$ is the sub-task whose instructional multimedia contents are displayed in front of the user.

There are two external functions $R$ and $C$. $R$ is a recommendation function. This function chooses a sub-task in the order that the system believes to be the best, without considering the user’s intention. One example is the scheduling algorithm proposed by Hamada et al. (Hamada et al., 2005). Because the design of $R$ is not the focus of this paper, we simply use a static, preliminary determined order as the return value of $R$. $R$ is called only when our algorithm has no clue how to estimate the intended next sub-task (lines 2 and 24).

$C(o)$ detects completion of $\hat{V}^-$. This function is called whenever an object is released. The details of these functions are given in 3.6.

The procedure at line 5 was provided in 3.3. The procedure from lines 7 to 14 maintains $O^h$ according to detection of each object access. Lines 15 to 24 estimate the intended sub-task. Since $\hat{V}^-$ is used mainly for recovery, we give priority to $\hat{V}^+ \cup \hat{V}^-$. When several sub-tasks mark the highest score, we adopt the one that first appears in the order recommended by $R$.

### 3.6 Design of completion detector $C(v)$

$\hat{V}^-$ is maintained by detecting completion of sub-tasks whenever the worker releases objects. There are many action recognizers, and some of them are specialized for culinary tasks (Rohrbach, Amin, Andriluka, & Schiele, 2012; Packer, Saenko, & Koller, 9
Algorithm 1: Entire process for forecast.

1: \( \mathcal{O}^h = \emptyset, \hat{\mathcal{V}}^+ = [s|v \in \text{source vertices in } G], \hat{\mathcal{V}}^- = \emptyset \)
2: \( v^c = \mathcal{R}(\hat{\mathcal{V}}^+, \mathcal{O}^h) \)
3: \( \textbf{while } \hat{\mathcal{V}}^- \neq \mathcal{V} \textbf{ do} \)
4: \( \text{display } v^c \)
5: \( \text{maintain } \hat{\mathcal{V}}^+, \hat{\mathcal{V}}^-, \text{ and } \hat{\mathcal{V}}^\rightarrow \text{ along with } \hat{\mathcal{V}}^- \)
6: \( \text{wait until any object } o \text{ is touched or released} \)
7: \( \text{if } o \text{ is newly touched then} \)
8: \( \mathcal{O}^h = \mathcal{O}^h \cup \{o\} \)
9: \( \text{else if } o \text{ is newly released then} \)
10: \( \mathcal{O}^h = \mathcal{O}^h \setminus \{o\} \)
11: \( \text{if } C(v^c) \text{ is true then} \)
12: \( \hat{\mathcal{V}}^- = \hat{\mathcal{V}}^- \cup v^c \)
13: \( \text{end if} \)
14: \( \text{end if} \)
15: \( v^+ = \arg\max_{v \in \mathcal{V}^+ \cup \mathcal{V}^-} L(v; \mathcal{O}^h) \)
16: \( v^- = \arg\max_{v \in \mathcal{V}^-} L(v; \mathcal{O}^h) \)
17: \( \text{if } L(v^+; \mathcal{O}^h) \geq 1.0 \text{ then} \)
18: \( v^c = v^+ \)
19: \( \text{else if } L(v^-; \mathcal{O}^h) \geq 1.0 \text{ then} \)
20: \( v^c = v^- \)
21: \( \text{else if } L(+) > 0.0 \text{ then} \)
22: \( v^c = v^+ \)
23: \( \text{else} \)
24: \( v^c = \mathcal{R}(\hat{\mathcal{V}}^+, v^c) \)
25: \( \text{end if} \)
26: \( \text{end while} \)

2012; Lei, Ren, & Fox, 2012; Iscen & Duygulu, 2013; Shimada, Kondo, Deguchi, Morin, & Stern, 2013; Kuehne, Fraunhofer, Arslan, & Serre, 2014). These methods can be a good solution for the problem of completion detection. Because it is not the purpose of this paper to develop an accurate completion detector, instead of implementing those action recognizers, we implemented a simple detector based on only two clues: the time elapsed since \( v^c \) was displayed, and confidence of the displayed information \( L(v^c; \mathcal{O}^h) \). Both of them are obtained from the calculation of Alg. 1.

Let \( n \) be an iteration index of the loop at line 3 in Alg. 1, and refer the variables in the algorithm with a subscript \( n \), e.g. \( v^c_n \) is \( n \)-th displayed sub-task. We also refer to \( t_n \) as the time when \( v^c_n \) starts to be displayed. Then \( t - t_n \) represents the elapsed time at time \( t \) after display of \( v^c_n \).

As the first condition, \( C \) returns false if \( t - t_n < \theta_t \). This is simply because any sub-task will take a certain amount of time. We set \( \theta_t = 1.0 \) second for any sub-task. We also disregard the touching of object \( o \) when \( o \) has been released in 1.0 second. In that case, we reset all of the parameters in Alg. 1 to the state before \( o \) has been touched,
and recalculate the procedure while ignoring the access to $o$.

If the elapsed time is longer than 1.0 second, we check the second condition of the highest record of $L(v; O^h)$ during term $[t_v, t]$. This is done by the following calculation.

$$C(v^c) = \begin{cases} 
  \text{true} & L_{\text{max}} \geq \theta_1 \\
  \text{true} & \theta_1 > L_{\text{max}} \geq \theta_2 \land R(\hat{V}_{n-1}^+, v_n^c) = v_n^c \\
  \text{false} & \text{otherwise},
\end{cases} \quad (3)$$

where $L_{\text{max}} = \max_{[t_v, t]}(L(v^c; O^h))$. $\theta_1$ and $\theta_2$ are respectively higher and lower thresholds obtained by the elemental scores in Table 1. Let $\sigma_{w_1}^m$, $\sigma_{w_0}^m$, $\sigma_{w_1}^a$, and $\sigma_{w_0}^a$ be respectively a material with tools, a material without tools, a tool with materials, and a tool without materials, which corresponds to the cases in Table 1. Then $\theta_1$ is given by $\sigma(\sigma_{w_1}^m) + \sigma_{w_0}^a$, $\theta_2$ must be lower than $\sigma(\sigma_{w_1}^m)$, and we set it 0.5.

$C$ returns true if $L(v^c; O^h)$ has reached higher than $\theta_1$. Even when $L(v^c; O^h)$ scores lower than $\theta_1$, $C$ returns true when the score is more than $\theta_2$ and $v_n^c$ corresponds to $R(\hat{V}_{n-1}^+, v_n^c)$, where $\hat{V}_{n-1}^+$ is the value of $\hat{V}^+$ at $(n-1)$-th iteration. This condition derives from the idea that $v_n^c$ should be executed consecutively after $v_n^c$. $C$ returns false in any other case.

In the update phase at line 12, we push $v^c$ and any sub-task $v$ that satisfies $v < v^c < \hat{V}^+$ into $\hat{V}^+$. This completion-detecting algorithm may be combined with any other sophisticated action recognizers. This remains as future work.

4 Interface Implementation

4.1 Web-based platform CHIFFON

To achieve an interface that is controllable from the intention-sensing system, we implemented a recipe display system using a web application framework, called CHEF’s Interface For Food preparation (CHIFFON). CHIFFON has server-side and client-side programs, and two-way communication is established between them (the blue arrows in Fig. 5); client-to-server messages deliver all user operations to the system, and server-to-client messages are orders for clients rendering the instructional multimedia contents on a web browser. Figure 6 shows the graphical user interface (GUI) rendered on the web browser.

The server-side program is also capable of receiving a message from any external systems, including the object sensor module in Fig. 1. When the server gets any messages from the external system, it generates and sends a rendering order to the target client web browser (the red arrows in Fig. 5).

We designed a recipe format that is capable of describing the workflow structure, text instructions, links to multimedia contents, and $O_i$ for each sub-task $v$. We also provided an application programming interface (API) for the external systems.

The recipe format, called Hand-Work Markup Language (HWML), is an XML document whose structure and content pattern are defined in the format of Relax NG schema (Murata, 2001). This format allows users to describe the workflow structure of
sub-tasks. Each sub-task can own several triggers. A trigger is a set of conditions, and we describe $O_i$ as the trigger in this study.

The API receives various orders as an HTTP GET request as long as the request is written in the format specified by the API. Once the server-side program has accepted the order through the API, it sends a message to the client-side program. Therefore, any external systems can control the client-side program in a sidelong way through this API.

The intention-sensing algorithm proposed in the previous section is implemented in the server-side program. It was called when the object sensor module sends a message that notifies an object access through the API. Whenever the algorithm called, it calculates the most likely intended sub-task, and sends a rendering order to the target client web browser.

4.2 Object Sensor Implementation

The last module we have yet to mention in Fig. 1 is the object sensors. This module detects the user’s access to objects. An object access is described by a tuple $[t, o, a(o)]$, where $t$ is time-stamp, $o$ is an object label, $a(o)$ is a label of object-access, namely
“touched” or “released”. The output of this module is sent to the intention-sensing algorithm on the server through the CHIFFON’s API.

There are many choices to implement this module: RFID (Nakauchi, Fukuda, Noguchi, & Matsubara, 2005), RFID + Load sensors (Chang et al., 2006), AR Markers (Miyawaki & Sano, 2008; Ueda, Funatomi, Hashimoto, Watanabe, & Minoh, 2011), RGB-D cameras (Klompmaker, Nebe, & Fast, 2012), Multi-modal signal processing (Hashimoto et al., 2010, 2012; Yasuoka, Hashimoto, Funatomi, & Minoh, 2013). Furthermore, it will let the system predict user’s intention earlier, and possibly more accurate, to replace touch detection by touch forecasting through gaze estimation (Nakazawa & Nitschke, 2012), grabbing motion estimation (Prasad, Kellokumpu, & Davis, 2006), or observing activity with wrist-mounted camera (Ohnishi, Kanehira, Kanezaki, & Harada, 2015). Each method has pros and cons, and there may be a good combination for achieving an optimal solution.

Our focus is on the design of an intention-sensing system, and developing such an optimal solution is beyond the scope of this study. Instead, we adopted the Wizard of OZ method (WOZ) (Kelley, 1984) which has been widely accepted for the last 30 years for evaluating various kinds of intelligent interface prototypes.

The WOZ method uses a human as a wizard, who works as a substitute for an intelligent module. In our case, the wizard provides an alternative to the object sensor via an interface shown in Fig. 7. The wizard stays in the background of the system so that subjects do not notice the presence of the wizard. One big advantage of this method is that interface designers do not have to wait for the relevant technologies to mature sufficiently.

5 Experiments

There are two important factors that affect a system’s proper intention-sensing: non-unique $O_i$ because of the complexity of the recipe, and the nature of human activity discussed in 3.2. To evaluate each factor independently, we designed two different
Figure 7: Interface for wizards. When a button for object $o$ is pushed, $a(o) = 'touched'$ is sent with the timestamp. Then, the button turns red. $a(o) = 'released'$ is sent after pushing the red button again.

Cut napa cabbage  Boil in broth  Stir
Cut Japanese radish  mix starch with water
Cut carrot  drain water from tofu
Cut ginger  Slice scallion

Figure 8: Overview of ginger soup. Nodes colored in darker blue are mixing sub-tasks.

experiments. In the first experiment, we restricted users to not having any deceptive accesses and using only the tools listed in $O_v$, and compared the accuracy of displayed information between two recipes of different complexities. In the second experiment, we removed the restrictions and evaluated the system with cooking activities in a natural setting.

5.1 Robustness against recipe’s complexity

Since several sub-tasks possibly have the same combination of objects as $O_v$, the algorithm must be robust against such recipes. To evaluate the robustness, we prepared two different recipes: “ginger soup” and “fried rice” as shown in Figs. 8 and 9. The number of sub-tasks is 19 and 30 for each recipe.

“Ginger soup” is a recipe that is easy to guide by object access because $O_v$ for $v \in V$ is unique. In such a recipe, $v$ can be identified uniquely without any contextual information. In contrast, “fried rice” has many sub-tasks that share the same set of objects as $O_v$. They are indicated by the colored border in Fig. 9. There are four pairs...
Reheat rice  
Slice scallion  
Tear lettuce  
Beat eggs  
Heat oil  
Stir-fry  
Season the food  
Stir  
Heat sesame oil  
Mince ginger  
Make a broth  
mix starch with water  

Figure 9: Overview of fried rice with starchy sauce. The same boundary color indicates the duplicated $O_v$.

(orange, red, blue and green) and one triplet (pink) of sub-tasks that share the same $Q_v$.

Before the experiments, we gave the subjects the following restrictions.

1. Do not add any extra sub-tasks to the process; only do what the recipe instructs.
2. Do not touch any object unless it relates to what the subject intends to do.
3. Do not use a tool that is not called for the recipe.
4. Do not release the objects while the sub-task is ongoing.

Two subjects are employed in this experiment, and both are skilled amateur cooks. It is too difficult to cook with the above restrictions without practices. Therefore, the subjects rehearsed several times.

During the rehearsals and the real part of the experiment, the information was displayed automatically. At the same time, they were allowed to operate the interface manually on a touch display when they felt any inconvenience. The number of manual operations indicates the number of critical failure for the subjects. Hence, counting user operations enables us to count such failures separately from the other failures that do not affect the interface’s usability seriously.

Table 2 shows the accuracy of the displayed information and the number of user operations while cooking. Despite the duplicated $O_v$ in the fried rice recipe, the accuracy maintains a high value in both recipes. From this result, we confirmed the algorithm works with a difficult recipe as well as an easy recipe.

It can also be said that the system was scarcely operated by the subjects in every case, even though accuracy did not reach 100%. The subjects were mostly concentrated on their task of cooking, and did not check the display all the time. As a result, displaying out-of-context information barely affected the user as long as the right information was displayed at the right moment.

### 5.2 Evaluation with human activities in nature

To evaluate the system in a natural human activity, we used the recipe for “fried rice,” which is more challenging and practical than “ginger soup.” Different from the first
Table 2: Accuracy of displayed information and number of user operations.

<table>
<thead>
<tr>
<th>Subject</th>
<th>Ginger Soup</th>
<th>Fried Rice</th>
</tr>
</thead>
<tbody>
<tr>
<td>A</td>
<td>118</td>
<td>121</td>
</tr>
<tr>
<td>B</td>
<td>202</td>
<td>120</td>
</tr>
<tr>
<td># of displaying correct sub-tasks</td>
<td>16</td>
<td>18</td>
</tr>
<tr>
<td># of displaying false sub-tasks</td>
<td>7</td>
<td>6</td>
</tr>
<tr>
<td>Accuracy</td>
<td>88.1%</td>
<td>94.5%</td>
</tr>
<tr>
<td># of user operations</td>
<td>0</td>
<td>0</td>
</tr>
</tbody>
</table>

Table 3: Number of manual operations while cooking recipe for fried rice with starchy sauce.

<table>
<thead>
<tr>
<th>Subjects</th>
<th>A</th>
<th>B</th>
<th>C</th>
<th>D</th>
<th>E</th>
<th>G</th>
<th>H</th>
</tr>
</thead>
<tbody>
<tr>
<td>Manual Operation (times)</td>
<td>2</td>
<td>2</td>
<td>9</td>
<td>1</td>
<td>2</td>
<td>5</td>
<td>3</td>
</tr>
</tbody>
</table>

experiment, we set no restrictions this time.

Seven subjects are recruited for this experiment, three males and four females. They cook at least once a week, and thereby have a certain level of culinary skill. To enhance the naturalness of the activity, we gave subjects ten minutes to check the entire recipe before starting observation, and instructed them to make a plan for cooking the dish on an optimal schedule.

We also compensated for the unfamiliar cooking environment. Because the subjects did not know where the things were stored, it was highly expected that this situation would restrict user activity. In particular, cookware tends to be used more often when stored in an easy-to-find location. To cancel this effect, we asked the subjects to determine what kind of cookware they would need beforehand, and placed it in plain view before starting the task.

The result of this experiment is shown in Tables 3 and 4. In Table 3, subjects C and F were very sensitive to false display; they checked the display at every sub-task, and corrected the error nine and five times in total, respectively. It was revealed that our system is not very effective for this type of users. On the other hand, the remaining four subjects rarely operated the system. This indicates that the object access can be a strong clue to sense the user’s intention.

We examined the accuracy of displayed information for subject A to E precisely. In all data, there were relatively more falsely displayed sub-tasks than the result in Table 2; however, subjects A, B, D, and E were hardly confused by them. A typical error occurred at mixing sub-tasks; \( v \) went forward and backward repeatedly between the two sub-tasks, mixing sub-task and the next one (\( d \) and \( e \) respectively in Fig. 2). This happened in the following scenario. First, one of the target materials was touched, and the instruction for the mixing sub-task was displayed. Then, the material was released, and the completion detector falsely regarded the release as the completion of the mixing sub-task even when there were some other materials that should be mixed. After that, touching of the other materials caused the system to recover from the error. This continued until all the materials in the sub-task were mixed up.

In most cases, the failures described above were ignored by the subjects, and the
Table 4: Accuracy of displaying steps and number of manual error corrections.

<table>
<thead>
<tr>
<th>Subjects</th>
<th>A</th>
<th>B</th>
<th>C</th>
<th>D</th>
<th>E</th>
</tr>
</thead>
<tbody>
<tr>
<td>a) expected access</td>
<td>99</td>
<td>101</td>
<td>72</td>
<td>98</td>
<td>77</td>
</tr>
<tr>
<td>b) deceptive access</td>
<td>92</td>
<td>106</td>
<td>73</td>
<td>76</td>
<td>95</td>
</tr>
<tr>
<td>c) # of displaying correct sub-tasks</td>
<td>128</td>
<td>162</td>
<td>108</td>
<td>135</td>
<td>121</td>
</tr>
<tr>
<td>d) # of displaying false sub-tasks</td>
<td>63</td>
<td>45</td>
<td>37</td>
<td>39</td>
<td>51</td>
</tr>
<tr>
<td>e) (d) at (a)</td>
<td>23</td>
<td>15</td>
<td>11</td>
<td>17</td>
<td>19</td>
</tr>
<tr>
<td>f) (d) at (b)</td>
<td>40</td>
<td>30</td>
<td>26</td>
<td>22</td>
<td>32</td>
</tr>
</tbody>
</table>

Accuracy: \(\frac{(c)}{(c)+(d)}\)

<table>
<thead>
<tr>
<th></th>
<th>A</th>
<th>B</th>
<th>C</th>
<th>D</th>
<th>E</th>
</tr>
</thead>
<tbody>
<tr>
<td>Accuracy: (\frac{(c)}{(c)+(d)})</td>
<td>67.0%</td>
<td>78.3%</td>
<td>74.5%</td>
<td>77.6%</td>
<td>70.3%</td>
</tr>
</tbody>
</table>

algorithm recovered successfully in subsequent object accesses. In this sense, it was confirmed that our design of the interface and the error recovery mechanism worked well even against human activity in nature.

Other types of failures happened when objects were released in the middle of execution. This kind of situation was observed, for example, when a heating sub-task was executed in parallel with other sub-tasks, and when the subjects took a short break.

All of the above failures were caused by failures of detecting completion of sub-tasks. These failures can bother users who react as subject C and F did. To confirm the completion of each sub-task more accurately, we should involve other clues besides simply object access. Addressing this problem remains as future work.

5.3 Analysis of inputs from wizards

In previous sections, we discussed the strategy of the proposed algorithm from the aspect of natural human activity. As a matter of practice, the strategy also works robustly against negative effects by failures at object sensor modules, as well as deceptive access and use of unforeseen tools. We discuss this property in this subsection.

A user accesses to object frequently during a cooking task, and it is not a rare case to access three or four different objects in a second. Therefore, it is hard to avoid false recognition of access to objects and false operation on the interface for wizards even under the experiments by the WOZ method. These are regarded as failures on the object sensor modules.

Table 5 shows the failures in experiments in subject A to E. These failures are confirmed after a careful survey of video and operational logs. A false positive is a case that a wizard input a touch/release of untouched/unreleased object. Inversely, a false negative is a lack of input for touched/released objects. The precision and the recall of input by wizard were respectively 94.4% and 73.5%. Generally speaking, precision and recall are in a trade-off relationship, but the priority can be controlled by some parameters of a classifier. In this sense, the wizards, which are the object sensor in the experiment, are a classifier controlled to have a higher precision and a lower recall.

Table 6 shows the breakdown of false positive inputs and the number of displaying false sub-tasks caused by them. From this table, we can see that false positive inputs
Table 5: Failures in wizard’s inputs.

<table>
<thead>
<tr>
<th>Subjects</th>
<th>A</th>
<th>B</th>
<th>C</th>
<th>D</th>
<th>E</th>
</tr>
</thead>
<tbody>
<tr>
<td># of false positive</td>
<td>15</td>
<td>9</td>
<td>2</td>
<td>10</td>
<td>9</td>
</tr>
<tr>
<td># of collect input</td>
<td>176</td>
<td>199</td>
<td>143</td>
<td>164</td>
<td>163</td>
</tr>
<tr>
<td># of total input</td>
<td>191</td>
<td>208</td>
<td>145</td>
<td>174</td>
<td>172</td>
</tr>
<tr>
<td># of false negative (missing touch)</td>
<td>50</td>
<td>47</td>
<td>11</td>
<td>11</td>
<td>34</td>
</tr>
<tr>
<td># of false negative (missing release)</td>
<td>52</td>
<td>44</td>
<td>11</td>
<td>11</td>
<td>33</td>
</tr>
</tbody>
</table>

Table 6: The comparison of number in false positive inputs and displaying false sub-tasks caused by them.

<table>
<thead>
<tr>
<th>Subjects</th>
<th>A</th>
<th>B</th>
<th>C</th>
<th>D</th>
<th>E</th>
</tr>
</thead>
<tbody>
<tr>
<td>Touch on the table in hand</td>
<td>6</td>
<td>2</td>
<td>1</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>Release on the table in hand</td>
<td>7</td>
<td>3</td>
<td>2</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>Touch on the table</td>
<td>0</td>
<td>1</td>
<td>3</td>
<td>0</td>
<td>1</td>
</tr>
</tbody>
</table>

5.4 Discussion

The problem of maintaining $V^-$ is regarded as that of identifying progress state of the task. There is a study by Schneider (Schneider, 2009) that models the progress transition by a dynamic Bayesian network (DBN), and simulate a cooking activity in the simulation on randomly generated DBNs. In the DBNs, each node corresponds to one progress state, and in their experiments, the number of node in DBN are 100, which is clearly too small to simulate a real cooking activity.

The progress state $V^-$ can be coded as a binary assignment of completion label to each $v \in V$, namely $\text{label}(v) = 1$ if $v \in V^-$ and vice versa. If calculating the number of possible progress states in $G$ along with Appendix B, there are respectively 376 and 2940 different progress states on the graph of the ginger soup (Fig. 8) and the fried rice (Fig. 9). It is difficult to treat such a large number of states for a probabilistic graphical model such as DBN.

Instead of considering the probability in such a large-scale state space, we modeled the progress as a definite context $V^+$, and expanded it to $V^-$ and $V^+$ for dealing with the uncertainty of human activity. This expansion strategy provided a good scope to search and track the accurate progress state in the large-scale state space.

Not only the number of nodes, there are many edges that represent a transition from one state to another. In a real human activity, a worker chooses to which state he/she transit along with many biases unconsciously. The objects in hand are a type of bias, and it is fully utilized in our method. Another bias is the structure of graph $G$. For
example, the worker can execute sub-task $c$ of Fig. 2 after finishing $a$ in theory, but more possibly, the worker will do $b$, which is the next sub-task on the same path with $a$, in actual activity. This bias is reflected in our system as the preliminary defined order of the sub-tasks provided by $R$. The order is not random, but holds some kinds of regularity along each path. This helps the system to predict the next sub-task whenever there is no clue from the object access. The proposed method worked successfully in real human activity because it took these biases into consideration.

6 Conclusion

In this paper, we proposed an intention-sensing interface for guiding cooking recipes. The system senses intention via a user’s access to objects. Since touching and releasing actions happen naturally in human activity, use of those actions can be a solution to assist humans working on a sophisticated physical tasks organized by many sub-tasks in a workflow. The technical problems for utilizing object access are deceptive access and use of unforeseen tools. Both of them often happen in human activities in nature. We focused on such human factors, and developed an algorithm that is less deceived by the deceptive access and unforeseen tools.

In the experiments, we first compared the accuracy of intention-sensing with two different recipes: “ginger soup”, which has unique object set $O_v$ for any $v \in V$, and “fried rice”, some of whose sub-tasks share the same set of objects as $O_v$. For the purpose of removing the human factors of deceptive accesses and unforeseen tools, we restricted human activity strongly. The former recipe was guided precisely by the system while there were no deceptive accesses or unforeseen tools. The latter recipe, which was expected more difficult to be guided, was also guided precisely. As a result, we confirmed that our algorithm has a certain level of robustness against the complexity of recipes while identifying the intended sub-tasks.

The second experiment was performed with human activity in nature. Namely, we did not restrict subjects’ activity. Rather, we laid out the situation so as to provoke more unexpected activity. We used the “fried rice” recipe for this task as a more challenging scenario than “ginger soup.” As a result, the sub-tasks were displayed with 73.6% accuracy on average through the five observations for subjects A to E. From the viewpoint of human-computer interaction, it is more important how often the subjects operate the interface manually. In the experiments, four subjects needed to input only two manual operations at most, and three, five, and nine times for subjects G, F, and C respectively, against 30 sub-tasks in the recipe (thereby 30 operations are expected with the traditional systems).

Even though we confirmed the potential availability of object access for the problem of intention-sensing, it is still a pilot system and has many additional topics to be studied. First, we need to develop an object sensor for different tasks. The writing cost of recipe in the workflow graph representation is the second topic. There are some studies endeavoring to extract the structure of sub-tasks automatically from traditional text-style recipes (Mori, Maeta, Yamakata, & Sasada, 2014). Third, the recommendation function should also be further investigated to attain higher usability. Location and posture of hand (Song et al., 2013) and those of objects will be important evidences to
estimate human intention. This should eliminate deceptive accesses and enhance the accuracy of the proposed system. Lastly, for detecting completion of sub-tasks with practical cost, we need to consider any other effective clues in addition to object access. All of these tasks remain as future work.

A Algorithm for assigning materials to each node in workflow.

Let \( G(V, E) \) be a digraph representing a recipe workflow, and consider a sub-task \( v \) as a process that receives one or more materials as input, and output one material. The input materials correspond to \( O^m_v \), and \( o^m_v \) denotes the output material from \( v \in V \).

When \( v \) is a leaf node, we manually assign \( O^m_v \) (A to \( a \) and \( B \) to \( c \), respectively in Fig. 2). Otherwise, \( O^m_v \) is obtained as a set of output from the nodes that have an out-edge to \( v \).

Let \( u_i \) be a node that is adjacent to \( v \) by \( v \)'s \( i \)-th in-edge \( f_{u_i}; v \in E \). When \( v \) has only one in-edge (thereby \( i \) is always 0), \( O^m_v \) consists of only one unique element. Because we use the same label to a material as long as they are not mixed with others, \( o^m_v \) is equal to the unique input element at \( e \). Namely \((A, B) = O^m_e \).

This definition is compatible with systems that can track the materials over the sub-tasks, such as (Hashimoto et al., 2010), or those driven by RFID in an assembling context.

B Number of possible progress states on a flexible recipe

Let \( G(V, E) \) be a directed graph representing work flow of the recipe, where \( v \in V \) corresponds to a sub-task, and \([u, v] \in E \) is a directed edge. A sub-task \( u \) must be completed to start \( v \) if \([u, v] \in E \).

For counting possible progress states (PPSs), we consider a local progress state around \( v \). Namely,

\[
N_v = \{ u | [u, v] \in E \} \quad (4)
\]

\[
S(v) = \begin{cases} 
\{ l(u) | u \in N_v \} & \text{if } N_v \neq \emptyset \\
\emptyset & \text{if } N_v = \emptyset 
\end{cases} \quad (5)
\]

where \( l(u) = 0, 1 \) is a Boolean variable indicating the completion of sub-task \( u \) (\( l(u) = 1 \) for \( u \)'s completion). \( v \) is ready to be executed if all Boolean variables in \( S(v) \) are true (or \( S(v) = \emptyset \)).

For simplicity, we assume that \( G \) is a tree. This assumption is true for most recipes. We note that the following estimation will not decrease drastically even if there are a few acyclic closed pathes in \( G \). Let us consider subtree \( G_v \) cut out from \( G \) by node \( v \),
namely a subgraph induced from \( v \) and all the ancestor nodes of \( v \). Then, the number of PPSs in subtree \( G_v \) is counted up by function \( \#(S(v)) \) as

\[
\#(S(v)) = \begin{cases} 
1 + \Pi_{u \in N_v}(\#(S(u))) & N_v \neq \phi \\
2 & N_v = \phi,
\end{cases}
\]

where \( 1 \) is added to count the following two cases differently: the cases where \( v \) is ready but not completed, and where \( v \) has been completed. The same cases are also counted when \( N_v = \phi \).

The entire number of PPSs in \( G \) is obtained as \( \#(S(v')) \), where \( v' \) is the sink node of workflow graph \( G \). Because of Eq. (6)'s direct product property, \( \#(S) \) increases combinatorially at each conjunction on the workflow \( G \), and cannot be treated by general DBN models.

**Acknowledgements**

This work was supported by JSPS KAKENHI Grant Numbers 24240030, 26280084 and 16K16099.

**References**


Işcen, A., & Duygulu, P. (2013). Knives are picked before slices are cut: Recognition through activity sequence analysis. In *Proc. of the 5th international workshop on multimedia for cooking eating activities* (pp. 3–8).


