Adaptive Task Planner for Performing Home Service Tasks in Cooperation with a Human

Seung-Jae Lee, Jin-Man Park, Deok-Hwa Kim and Jong-Hwan Kim

Abstract—To perform a home service task through cooperation with a human in a real environment, a robot needs to deal with the environmental changes and accordingly plan appropriate behavior sequence. For this purpose, in this paper, we propose an adaptive task planner which is based on memory and reasoning. A robot perceives user behaviors and objects using an RGB-depth and thermal sensor. The robot stores a temporal sequence of behaviors for performing a task in its episodic memory that is realized by a sequence to sequence network. When the user command is given, the episodic memory is used to retrieve the behavior sequence to carry out the command. On the other hand, when the robot perceives user behaviors, the robot postpones its behavior till his/her behavior is stopped. Once stopped, the episodic memory retrieves the behavior sequence to conduct a task that the user has intended. A task scheduler schedules the behavior sequence from the memory and sends it to an internal simulator. The internal simulator confirms the behavior sequence to be executable and then if executable, it sends the next executable behavior to the execution module. If a behavior fails in the internal simulation test, fast forward planner generates an alternative behavior sequence to resolve the failed behavior problem. The effectiveness and applicability of the proposed planner is demonstrated by a wheel-based humanoid robot.

I. INTRODUCTION

As science and technology have advanced, people’s perspectives on robots have changed a lot [1]. Robots are considered as partners who can fulfill requested tasks intelligently and dexterously for human beings. As a partner, robots need to be able to perceive the environment and conduct tasks autonomously without detailed description of the tasks from users.

To perform a task, determining and executing a sequence of primitive behaviors is an important feature to be considered. Various task planning techniques have been presented over the years [2], [3]. Robots recognized a grasp point and a location point respectively to stably pick and place objects [4]. It learned picking and placing skills from human demonstrations by using the Markov decision process. Techniques to execute piling up boxes were also presented [5]. An architecture for behavior selection and execution for conducting a task was also developed [6]. By using perceived information online, a robot’s primitive behaviors are dynamically organized to perform a task. In [7], they reduced search time by using a hierarchical approach, and a robot can plan and perform a task efficiently. An architecture that combines task planning and path planning was also devised and tested for humanoid robots [8].

To retrieve appropriate primitive behaviors in a task planning problem, a robot needs its own memory model to store and retrieve what it has experienced. Researchers have proposed various artificial memory models that have been inspired by human episodic memory. Episodic memory stores a temporal sequence of behaviors as an episode, where the behaviors can be explicitly stated. A research showed that episodic memory is required the capability to capture the spatio-temporal characteristics [9], [10]. Episodic memory is used to memorize past experiences of a robot and human task manipulation episodes [11]. Using the memory, robots can retrieve memorized experiences of manipulation tasks. In [12], storing, retrieving, and forgetting of episodic memories were implemented by using a generalized model of fusion adaptive resonance theory (F-ART) network. Conditional random fields (CRFs) trained with a dataset of task descriptions in natural language and the corresponding robot behavior logs are also used as the robot memory [2]. Due to variations in the environment in the dataset, the robot was able to perform given tasks context-sensitively. Despite the good performance of these works, they have the disadvantage of lacking generalization ability. This means that even a small change in the environment may cause serious errors.

To perform a task in a real environment, a robot needs a modular framework that develops task intelligence more efficiently and systematically. For this purpose, the intelligent operating architecture (iOA) was developed, which is an integrated architecture consisting of five modules: perception, internal state, memory, reasoning, and execution [13]. Each module performs its own role and exchanges its information, and thus it helps a robot to perform its task efficiently [14], [15]. The episode memory model was designed using Deep ART as memory module to store temporal sequences of behaviors as task episodes and to retrieve one of them according to a current environment situation [16]. As an reasoning module, a developmental episodic memory-based mechanism of thought (DEM-MoT) was presented, which helps to select a most appropriate behavior for the current situation. In an execution module, Q-RRT* was used, which is a sampling based motion planning algorithm. It has fast convergence rate in a complicated environment. The task planning methods and memory models can be applied to the modular framework above. The application has shown good performance in limited environments or tasks, but it is not able to handle dynamic environments. In [17], a method to adaptively plan a task in an unstructured environment was
devised to avoid a collision. If a robot cannot perform a task using an behavior sequence generated by a memory module due to an obstacle, the robot uses a fast forward (FF) planner to generate an alternative behavior sequence for the task. However, even in this case, when a behavior sequence for the task has been determined, it cannot cope with situations such as human interruptions or changes in the surrounding environment.

On the other hand, robots have to interact with human beings in the home environment. For efficient human-robot interaction (HRI), especially from the human-robot cooperation point of view, robots should be able to read human intentions to better aid humans by providing efficient and cooperative behaviors. In [18], information perception was studied for HRI. Coordination mechanisms for action observation, action coordination, task sharing, etc. were developed to support human-robot cooperation [19]. In [20], a learning-based controller was designed for human assistance by focusing on mimicking human behaviors that are related to objects. In [21], human intention was defined as “desired behavior of humans using objects” and affordance network and map to infer human intention from environment information was developed. Also a Control Architecture for the Dynamics of Embodied Natural Coordination and Engagement (CANDENCE) was designed [22]. For cooperation with a robot, turn-taking style is proven effective in performing a task. CANDENCE helps a social robot to control turn-taking autonomously. For this purpose, interruptible modality actions, resource monitoring and dynamic scheduling supervisor are needed. Based on these studies on human-robot cooperation, some elements are applied to our proposed planner for a home service robot.

This paper proposes an adaptive task planner for a home service robot through cooperation with a human. A robot perceives the environment using an RGB-depth and thermal (DT) sensor, and a gaze control algorithm helps perceive the environment with attention. We define a user behavior, which is used to predict user intention, as a human behavior that is related to objects in the workspace, like ‘A user grasps a cup.’ User behaviors and environment, including map and objects, are recognized by the perception module, and they are delivered to the adaptive task planner. A natural language user command, obtained by speech to text API, can also be delivered to the adaptive task planner. The adaptive task planner consists of an episodic memory, a task scheduler, an internal simulator and a FF planner. The episodic memory, in this paper, refers to sequence to sequence (seq2seq) neural network which is trained through 50,000 behavior sequence execution data, each of which consists of a user command and a list of objects as input and a behavior sequence as output. This network receives a natural language user command and a list of nearby objects as input, and then generates a robot behavior sequence of behaviors. There are two ways in which the network creates a behavior sequence, for a user command and a list of perceived objects and user behaviors and a list of perceived objects. Through either way, the generated behavior sequence is passed to the task scheduler. Then the task scheduler plans behavior sequence to be executed by the robot. The internal simulator tests whether the first behavior in the behavior sequence can be performed, decides which behavior to conduct with an object, and sends it to the execution module. If the first behavior is turned out not executable through the internal simulation, the information is delivered to the FF planner which generates an alternative behavior sequence. Then, the task scheduler adds the local plan to the behavior sequence to perform the task successfully. The execution module is responsible for the robot’s behavior. The proposed architecture is implemented to a humanoid robot, Mybot, a humanoid robot developed in the RIT Laboratory at KAIST to verify its effectiveness. The experiments demonstrate that the robot with the proposed architecture could carry out tasks successfully through cooperation with a human in a real situation.

The key contributions of this paper are as follows. An adaptive task planner for home service robot is proposed, which adaptively generates behavior sequence to meet the human intention. A novel neural network model as an episodic memory, is proposed to generate a robot behavior sequence to a user command in natural language and a list of surrounding objects.

The remainder of this paper is organized as follows. Section II presents the overall architecture and Section III proposes the adaptive task planner. Section IV describes the environment and parameter settings for the experiment and discusses experimental results. Finally, concluding remarks follow in Section V.

II. OVERALL ARCHITECTURE

Fig. 1 shows the overall architecture which consists of the perception module, adaptive task planner and execution module. Task performance processes from perception to execution are integrated to conduct tasks autonomously through cooperation with a human.

The perception module perceives objects and user behaviors by using robot’s RGB-DT sensor. We assume that all objects can only exist either on a flat plane or in a hand. With the normalized RGB image segment, a pre-trained neural network is used to recognize objects [23]. The temporal pose changes of a hand and an object are considered in
determining a behavior to be executed. The user hand is perceived in the same way as an object is detected, but a thermal sensor helps perceive the user hand more accurately by matching an image portion with a certain range of temperatures in the thermal image to an RGB-D sensor image. The perception module delivers the recognized user behaviors to both the behavior sequence (BS) generator for user behavior of the adaptive task planner and the execution module. A user behavior contains object information which is related to the behavior. The perception module also delivers environment information to the adaptive task planner. The environment information includes the position and orientation of objects as well as map information.

Adaptive task planner consists of episodic memory including two BS generators, task scheduler, internal simulator and FF planner. Details of each part are described in the adaptive task planner section. When the user command is delivered, episodic memory retrieves a behavior sequence which is needed to accomplish the user command. Then, the BS generator for user command sends the retrieved behavior sequence to the task scheduler. Once the user behaviors are recognized and completed, the episodic memory predicts the next behaviors. Then, the BS generator for user behavior sends these behaviors as the user dependent behavior sequence to the task scheduler.

The task scheduler generates a behavior sequence to be executed by the robot based on the received behavior sequences, and delivers the behavior sequence to the internal simulator. The internal simulator pre-tests whether the robot can perform the first behavior of the behavior sequence. If the behavior is executable, the simulator sends this behavior to the execution module. If the first behavior is not executable, the internal simulator sends this failed behavior to the FF planner as a failed behavior planner. At this time, a domain file and a problem file, which are created based on Planning Domain Definition Language (PDDL), are generated based on the environment information and the current robot state, and then local planning is performed through the FF planner [17]. The FF planner generates an alternative behavior sequence to resolve the failed behavior problem and delivers the alternative behavior sequence to the task scheduler to be added to the existing behavior sequence that is yet to be performed.

In the execution module, the robot executes the behavior and the gaze control based on the robot behavior and perceived user behavior. Fig. 2 shows a state machine of the gaze control, where $O_u$ is an object related to a user behavior and $O_r$ is an object related to a robot behavior. The robot controls its gaze, arms and a torso in two different cases, depending on the situation.

In the case when the executable behavior is received from the internal simulator and there is no user behavior, the behavior generator in execution module forms the trajectories of the arms and torso. After the trajectory generation, the robot selects the hand behavior, e.g., ‘grasp’, ‘release’ or ‘none’. The robot’s gaze is controlled based on the robot behavior. The robot’s gaze focuses on $O_r$ for grasping, moving, pouring, etc., a target object or a target place. In locating and releasing behavior, the robot’s gaze is on the current grasping hand. In the case when the user behavior is being presented, only the gaze is controlled and the robot arms are left immobile. The robot’s gaze is controlled based on the perceived user behavior. If a user behavior related to an object is detected, the robot controls its gaze to $O_u$.

The perception module, the adaptive task planner, and the execution module are designed to use the turn-taking method for cooperation with humans. The perception module monitors environment information and human information, and the execution module can be interrupted by the perception module or the adaptive task planner.

III. ADAPTIVE TASK PLANNER

The adaptive task planner is based on memory and reasoning, in which behavior sequence scheduling can be flexibly performed according to a robot state and surrounding environment information. As memory, episodic memory is employed to store the behavior sequence. Reasoning is for task scheduling and confirming it through the internal simulator.

A. Episodic Memory

As shown in Fig. 3, a behavior sequence for a perceived object list and user command is generated by a sequence to sequence network which is commonly used in neural machine translation [24]. In order to generate output behavior sequence by combining two inputs (perceived object list and user command), two encoders, $E^{obj}$, $E^{com}$ and one generator, $G$ are constructed, and an attention mechanism [25] is applied to decide which part of each input to concentrate, so that the learning can proceed well even if the input or output sequence is very long. This network can be used in two ways for behavior sequence generation either by user command or by user behaviors.

1) Behavior Sequence Generation by User Command:
the generator RNN with the attention mechanism, and then a probability distribution for the candidate output symbols is generated through the softmax layer.

The generator $G$ decodes $h$ and $h'$ into a sequence of feature vectors, $e = e_1, ..., e_T$, based on the attention mechanism, where $e_t = AttentionMechanism(e_t^{in}, h, h')$. Here, $e_t^{in}$ is the embedded word vector from $y_{t-1}$, with the same embedding layer used in the encoders. With $c$, the generator $G$ generates a corresponding robot behavior sequence, $y$ such that $G(c) = y$. Let $g = g_1, ..., g_T$ denote the hidden states of the RNN cells in $G$. Each hidden state of the RNN cell $g_t \in \mathbb{R}^{n_g}$, where $n_g$ is the dimension of the hidden state, is computed as follows:

$$g_t = RNNCell(g_{t-1}, [e_t^{in}, c_t]).$$

(2)

The output behavior sequence symbol at time $t$, is computed as

$$y_t = Softmax(W_y g_t + b_y)$$

(3)

where $W_y \in \mathbb{R}^{n_y \times n_g}$ and $b_y \in \mathbb{R}^{n_y}$, and $n_y$ is the output vector dimension which equals to the number of output classes.

2) Behavior Sequence Generation by User Behavior:

a) Hallucination encoder: In this case, no user command is given as an input. So we introduce a new encoder $E_{hall}$, which uses an object list as input to produce output, similar to $E_{com}$ in the absence of a user command input. $E_{hall}$ is trained after the training of existing network. In order to reproduce the hidden state of $E_{com}$ without user command input, $E_{hall}$ is trained to minimize the following Euclidean loss between the last hidden states, $h'_{T_f}$ and $h''_{T_t}$, as shown in Fig. 4 (a):

$$L_{hall} = ||h'_{T_f} - h''_{T_t}||^2$$

(4)

where $h''_{T_t}$ is the last hidden state of $E_{hall}$. After the training of $E_{hall}$ is completed, $E_{hall}$ is used instead of $E_{com}$, as shown in Fig. 4 (b).

b) Behavior sequence generator: The feed-forward process of the generator $G$ is basically the same as that described previously except its input sequence. Here, as shown in Fig. 4 (b), the embedded vector $e_t^{in}$ is generated from $a_{t-1}$ if $t \leq T_u$, where $a_t$ is a user behavior symbol, obtained from perception module, at time-step $t$. Then, if $T_u < t \leq T_k$, the embedded vector $e_t^{in}$ is generated from $y_{t-1}$ that is the output of $G$ at time-step $t - 1$.

B. Task Scheduler and FF Planner

Based on the retrieved behavior sequence by the user command or the user dependent behavior sequence by the user behavior, task scheduler generates the behavior sequence and sends it to the internal simulator. Fig. 5 shows the change of the behavior sequence according to the situation. When the robot receives the performed behavior information from the execution module, the behavior is removed from the behavior sequence as in the blue behavior in the figure. If the behavior sequence comes from the episodic memory, the
behavior sequence is sorted and sent to the internal simulator like gray and yellow behaviors. In this case, the priority of the incoming behavior sequence is the most recent incoming sequence, and the existing behavior sequence is pushed backward. If the simulation result is not feasible in the internal simulator, the FF planner generates the alternative behavior sequence. Then, like the green colored subsequence in Fig. 5, the task scheduler puts the alternative behavior sequence before the existing behavior sequence. If the FF planner fails to create the alternative behavior sequence (for example, the object does not exist, the robot can not generate arm trajectory, or both arms of the robot hold objects), the task scheduler initializes the behavior sequence and sends ‘no executable behavior’ to the internal simulator.

C. Internal Simulator

The internal simulator keeps the behavior sequence up-to-date and makes a decision if the behaviors are executable. It simulates the first behavior in the behavior sequence based on the environment information and the current state of the robot. In this case, the robot determines which arm to use for the behavior based on the distance between the object and the hand and the state of the hand. If the behavior for the object is confirmed and the simulation result is feasible, the internal simulator delivers the executable behavior to the execution module. If the behavior is not executable, it is forwarded to the FF planner. When ‘no executable behavior’ is received from the task scheduler, the behavior to be delivered to the execution module is ‘go to the initial state of the robot’. In this case, if the object is in the robot’s hand, the robot places the object in an empty space with no obstacles on the table and moves to the initial state.

IV. Experiments and Results

A. Robot Settings

The proposed adaptive task planner was implemented in a wheel-based humanoid robot, Mybot, developed in the RIT Laboratory at KAIST. The humanoid robot model and its configuration for the experiment are shown in Fig. 6. Mybot consists of four parts: a head, two arms, a torso, and a lower body. Its width, height, length, and weight are 54.0 cm, 132.0 cm, 54.0 cm and 70.0 kg, respectively. The head has 17 degrees of freedom (DoFs), and it can provide gaze control and generate various emotional expressions. Each arm has 8 DoFs, with each hand having 3 DoFs. The torso has 2 DoFs, so the robot can bend and rotate its upper body. The lower body is an omni-wheel platform and the robot can move anywhere with no limitation in direction changes. An RGB-D sensor and a thermal image sensor are attached in the head to perceive the external environment. These help Mybot to perceive human behavior more accurately.

B. Dataset and Hyper-parameter Settings

To train the seq2seq network based BS generator, we manually created total 50,000 input-output pairs by hand, by repeatedly changing objects, user commands, and behaviors. A pair is in the form of \([w, w', y]\), where \(w\) is an object list, \(w'\) is a user command, \(y\) is a behavior sequence, as shown in Table I. All pairs are related to one of the four scenarios - “Arrange toys”, “Make me a toast”, “Water the flowers”, “Make me a cereal”. Among the 50,000 pairs, 45,000 pairs were used for training, and the remaining 5,000 pairs were used for tests. Note that the output behavior sequence in the dataset is a sequence of behaviors, where a behavior has the structure \([\text{object}_1 \text{ behavior object}_2]\). If a behavior needs only one object, then \(\text{object}_2\) is the same as \(\text{object}_1\). For example, ‘Grasp the blue cylinder toy’ can be expressed as ‘blue cylinder toy grasp blue cylinder toy’, whereas ‘Pour the low-fat milk into the purple bowl’ can be expressed as ‘low-fat milk pour purple bowl’. There are approximately a hundred of objects in the dataset, and a total of 10 robot behaviors are used in this dataset, such as grasp, throw, locate, release, move, pour, put, push, sprinkle, and squeeze. Other implementation details in the BS generator are listed in Table II.

C. Test Result of Behavior Sequence Generator

The training result of the hallucination encoder \(E^{\text{hall}}\) is shown in Fig. 7. Fig. 7 (a) is the Euclidean loss graph of the hallucination encoder in training set, while Figs. 7 (b), 7 (c) and 7 (d) are the results in test set. Fig. 7 (b) shows the change of test perplexity under the three conditions during the training of \(E^{\text{hall}}\), where the case \(E^{\text{com}}\) is used is indicated by the red line, and the case \(E^{\text{com}}\) is removed and zero vector is applied is denoted by the green line, and the case \(E^{\text{hall}}\) is used instead of \(E^{\text{com}}\) is denoted by the blue line. The test perplexity was approximately 30 times less when using the hallucination network than when not using it. Figs. 7 (c) and 7 (d) respectively show Exact match (EM) score and F1 score in test set. For the EM score...
TABLE I
DATASET FOR TRAINING THE SEQ2SEQ BASED BEHAVIOR SEQUENCE GENERATOR

<table>
<thead>
<tr>
<th>User Command to Behavior Sequence</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>User Command</strong></td>
</tr>
<tr>
<td>I want some wine sauce</td>
</tr>
<tr>
<td>Water the yellow pot</td>
</tr>
<tr>
<td>Give me something to eat with milk</td>
</tr>
</tbody>
</table>

TABLE II
IMPLEMENTATION DETAILS

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Value or Type</th>
</tr>
</thead>
<tbody>
<tr>
<td>The number of layers $l$</td>
<td>3</td>
</tr>
<tr>
<td>The dimension of hidden state $n$</td>
<td>256</td>
</tr>
<tr>
<td>The dimension of word embedding $e$</td>
<td>300</td>
</tr>
<tr>
<td>Max input length $m$</td>
<td>50</td>
</tr>
<tr>
<td>Batch size $B$</td>
<td>32</td>
</tr>
<tr>
<td>Learning rate $\alpha$</td>
<td>0.0001</td>
</tr>
<tr>
<td>The number of epochs</td>
<td>400</td>
</tr>
<tr>
<td>Pre-trained embedding layer type</td>
<td>Glove [26]</td>
</tr>
<tr>
<td>Optimizer</td>
<td>Adamax [27]</td>
</tr>
<tr>
<td>Attention mechanism</td>
<td>Bahdanau et al. [25]</td>
</tr>
<tr>
<td>RNN type</td>
<td>Simple Recurrent Unit [28]</td>
</tr>
</tbody>
</table>

Fig. 7. (a) The Euclidean loss graph of the hallucination encoder $E_{hall}$ in training set. (b) shows the change of test perplexity under the three conditions during the training of $E_{hall}$, where the case $E_{con}$ is used is indicated by the red line, and the case $E_{con}$ is removed and zero vector is applied is indicated by the green line, and the case $E_{hall}$ is used instead of $E_{con}$ is indicated by the blue line. (c) and (d) respectively show Exact match (EM) score and F1 score in test set.

and the F1 score, there were respectively 58.0% and 21.0% improvements when the hallucination method was applied, compared to the zero vector input.

**D. Experiments in Home Environment**

To show the effectiveness of the proposed adaptive task planner, human-robot cooperation experiments were conducted using a human-sized robot Mybot. Fig. 8 shows a snapshot of an experiment. The left of the screen shows the robot, the environment, and the user. The user command, the user behavior and behavior sequence are shown in the top right corner of the left screen.

Various information is displayed in the images on the right side of the screen. The top view part shows the table scene, and the object recognition part shows perceived objects and their boundary. The behavior recognition part shows a color change when there are some interactions between the object and the user. When the user grasps the object, the color is changed as blue. When the user pours something to the object, the color is changed as red. A depth image was used for object detection and the construction of an octo map. Thermal image was used for accurate human cooperative behavior detection. Depth and thermal images are essential because perception based on colored images has some limitations for object and human recognition. An octo map was used to check for possible collisions for the generated trajectories of the robot’s arm and torso. The top view is only a reference image.

A total of three experiments were performed. In each experiment, the workspace of the robot and the objects of the experiment environment were different. We trained additional behaviors in the episode memory as needed.

1) Experiment 1: Fig. 9 shows key snapshots from Experiment 1. Attached mp4 file is a video clip of this experiment. In the workspace of the robot, there were one table and red toys, a toy box, a cereal box, a milk carton and a bowl, which were placed on the table. The robot looked around and identified the environment information.

When the user commanded to the robot “Arrange toys”, then the memory module retrieved a corresponding behavior sequence and the behavior sequence was delivered to the internal simulator through the task scheduler. For each red toy, the behavior sequence like ‘grasp the toy’, ‘move the toy to the toy box’ and ‘release the toy’ was generated. Internal
Fig. 9. Key snapshots from Experiment 1. (a) Grasp the red square. (b) Put the red cylinder to toy box. (c) Look user behaviors (grasp the cereal). (d) Look user behaviors (pour the cereal to the bowl). (e) Pour the milk to the bowl. (f) Put the red triangle to the toy box.

When a user hand was suddenly visible in the view of the robot, the robot paused its behavior and observed the behavior of the user. If the behaviors of the user had been finished and there were no additional behaviors for a certain period of time, the episodic memory generated the user dependent behavior sequence and sent it to the task scheduler. User behaviors were ‘grasp the cereal box’, ‘pour the cereal box to the bowl’ and ‘locate the cereal box to the cereal box (original location)’. The user dependent behavior sequence was ‘grasp the milk carton’, ‘pour the milk carton to the bowl’ and ‘locate the milk carton on the milk carton (original location)’. Then, the user dependent behavior sequence was added before the existing behavior sequence. After finishing behaviors of the behavior sequence which is related with “Making the cereal”, the robot arranged the remaining red triangle toy.

2) Experiment 2: Fig. 10 (a) shows the environment of Experiment 2. In the workspace of the robot, there were a tray, a blue bottle, a pepper, a milk carton, a bowl, and a cereal box, which were placed on the table. The robot looked around and identified the environment information.

When the user commanded to the robot “Make me a cereal”, the episodic memory retrieved a corresponding behavior sequence, which consists of ‘grasp the cereal box’, ‘pour the cereal box to the bowl’, ‘locate the cereal box on the table’, ‘grasp the milk carton’, ‘pour the milk carton to the bowl’ and ‘locate the milk carton on the table’. Internal simulator delivered the executable behavior to the execution module and the robot continued the behavior sequence until an interruption occurs.

When the user did behaviors ‘grasp the cereal box’, ‘move the cereal box to the tray’ and ‘locate the cereal box to the tray’ to execute the task “Arrange the cereal box to the tray”, the robot looked the user behaviors. After the user behaviors were finished, the robot generated the behavior sequence for “Arrange the milk carton to the tray”. Then, the robot executed behaviors in the order ‘grasp the milk carton’, ‘move the milk carton to the tray’ and ‘locate the milk carton to the tray’. Fig. 10 (b) shows the execution of ‘locate the milk carton to the tray’.

3) Experiment 3: Figs. 10 (c) and 10 (d) show the environment of Experiment 3. In the workspace of the robot, there were two tables. A cup, a milk carton and a cereal box were placed on the one table, and a pepper, a wine sauce and a blue bottle were placed on the other table. The robot moved between the two tables and looked around to identify the environment information.

When the user located the cup in front of the robot on the table like Fig. 10 (c), the episodic memory of the robot retrieved a corresponding behavior sequence of ‘grasp the blue bottle’, ‘pour the blue bottle to the cup’ and ‘locate the blue bottle on the table’. Then, the behavior sequence was forwarded to the task scheduler and simulated in the internal simulator. However, ‘grasp the blue bottle’ was not executable in current situation, and thus the alternative behavior sequence was needed to be generated through the FF planner as a local planner. At first, the failed behavior made the robot move to the table on which the blue bottle existed. On this table, the blue bottle was hidden by the pepper, so the alternative behavior sequence for moving the
the pepper to the other place was generated. After putting the pepper on its side, the robot grasped the blue bottle and moved back to the table and pour the blue bottle to the cup. Fig. 10 (d) shows ‘pour the blue bottle to the cup’. In short, the robot executed the task, “Serve a beverage to the user”.

V. CONCLUSION

In this paper, we proposed the adaptive task planner for performing the home service tasks through cooperation with a human. Through the perception module, the robot perceived environment information including states of the robot and the user. The information was then forwarded to the adaptive task planner and execution module. The episodic memory generated either the retrieved behavior sequence to the user command or the user dependent behavior sequence to the user behaviors. The task scheduler generated the behavior sequence by using the received behavior sequences. Internal simulator confirmed the executability of the behavior sequence and delivered the executable behavior to the execution module. When the first behavior of the behavior sequence was not executable, the FF planner made an alternative behavior sequence to resolve the local failure. The execution module generated robot’s behavior trajectory and controlled robot’s gaze based on the situation. The effectiveness and applicability of the proposed adaptive task planner were demonstrated through the three experiments with Mybot. Mybot could handle various practical situations and performed tasks successfully while cooperating with the user.

ACKNOWLEDGMENT

This work was supported by Development of robot task intelligence technology, supported by the Ministry of Trade, Industry, and Energy and Institute for Information and communications Technology Promotion grant funded by the MSIT (No.2018-0-00677, Development of Robot Hand Manipulation Intelligence to Learn Methods and Procedures for Handling Various Objects with Tactile Robot Hands).

REFERENCES