Autonomous control of mobile robots using logical representation of map and inference of location

Megumi FUJITA
Graduate School of Humanities and Sciences
Nara Women's University
Nishimachi, Kitauoya, Nara
630-8506, Japan
Email: saboten@ics.nara-wu.ac.jp

Yuki GOTO
Graduate School of System Infomatics
Kobe Univesity
1-1 Rokkoudai-cho, Nada-ku, Kobe
657-8501, Japan
Email: 120x602x@stu.kobe-u.ac.jp

Naoyuki NIDE
Faculty of Human Life and Environment
Nara Women's University
Nishimachi, Kitauoya, Nara
630-8506, Japan
Email: nide@ics.nara-wu.ac.jp

Ken SATOH
Priciples of Infomatics Research Division
National Institute of Infomatics
2-1-2 Hitotsubashi, Chiyoda-ku, Tokyo 101-8430, Japan
Email: ksatoh@nii.ac.jp

Hiroshi HOSOBE
Faculty of Computer and Infomation Sciences
Hosei University
3-7-2 Kajino-cho, Koganei-shi, Tokyo 184-8584, Japan
Email: hosobe@acm.org

***Abstract*—We propose an action-decision method for autonomous mobile robots, in which a robot constructs a logical representation of a map of its surrounding environment from its perception and uses that map to determine a plan to logically reach its destination. We conducted an experiment in which a robot had a sub-goal to reach halfway to its destination and attempt to recognize that it has reached that sub-goal in order to proceed to the next goal. We first explain our experimental results then provide a discussion on these results and future work.**

I. Introduction

Techniques for identifying robot locations, such as SLAM, and moving procedures using an ordinary plan library have been recently developed. However, sometimes these techniques lack the ability to enable logical understanding of the positional relation between locations.

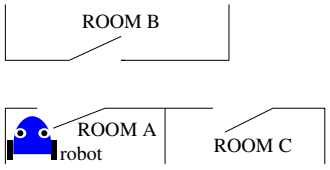

Fig. 1. Sketch map of building and robot's location

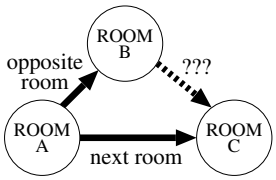

Fig. 2. Spatial relationship of rooms (robot's viewpoint)

For example, consider the situations in Figs. 1 and 2. First, we assume that a robot is in room A, and it knows how to get to room B from room A and to room C from room A. Suppose that we command the robot to go to room B and return to room A. The robot will then be able to move between rooms B and A. Then suppose that we command the robot to go to room C and return to room A. The robot will move similarly. Therefore, if we command the robot to go to rooms B and C without returning to room A, how will the robot move to accomplish this command?

If we use an ordinary plan library, since the robot does not have plans to move between rooms B and C, the robot will not be able to do such movement. On the other hand, if we use SLAM with a GPS sensor or laser range-finder device, the data obtained from this device are not the knowledge of the robot but the robot's geographical position; thus, the robot will not be able to understand the concept of the room unless it already had a record of these rooms' geographical positions.

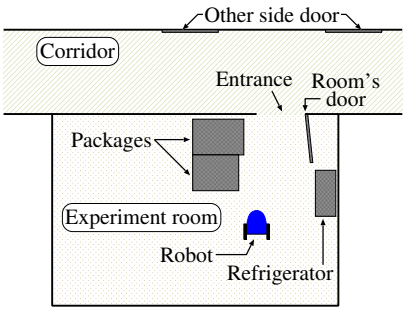

Fig. 3. Robot's environment in experiment room

Fig. 4. Q.bo Lite Evo

Our goal is to enable a robot to obtain new knowledge inferred from its perception and recognition. If the robot can recognize the relationships between rooms, it will be able to infer new relationships like those in Fig. 2. To achieve this goal, we have to know how to construct relationship knowledge and use it in the robot's moving process. To this end, we conducted an experiment involving a robot equipped with cameras and ultrasonic sensors that moves using its own knowledge on the relationship between a door and entrance (Fig. 3). We obtained successful and unsuccessful results, which we discuss later in this paper.

II. Implementation of our robot program

A. Robot for this study

For this study, we used Thecorpora's Q.bo Lite Evo robot (Fig. 4). This robot has one wheel in the front and two wheels in the rear. It has cameras as its eyes, which can capture the

978-1-5090-3931-9/16 $31.00 © 2016 IEEE
DOI 10.1109/ICA.2016.38

forward view. It also has two ultrasonic sensors near its front wheel that can detect obstacles.

B. BDI model

Our implementation is based on the BDI model, a model of autonomous agents that emulates the process of human goal achievement. A BDI agent, an agent based on the BDI model, first generates its goal using its beliefs obtained from its environments. By practical reasoning [1] using the beliefs, it selects a means (basically from its plan library) to achieve the goal, forms it as its intention, then attempts to maintain the intention and execute it (the top-left green box in Fig. 5).

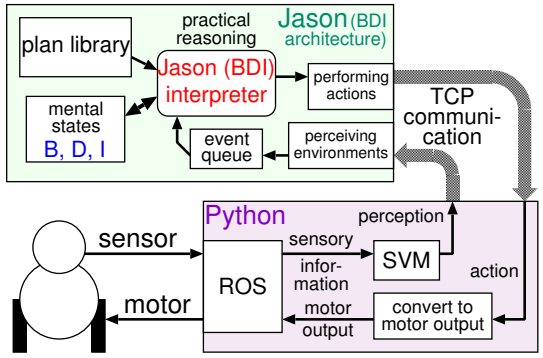

Fig. 5. BDI agent and implementation of robot's behavior

To make the robot have a goal and act while reasoning its sub-goals and plans to achieve them, we used the BDI model for the robot's action decision. To implement this, we used Jason[2], a platform for implementing the BDI architecture (the fundamental agent architecture on the BDI model). With Jason, practical reasoning can be described in a logic programming language approximately corresponding to Prolog.

Since the robot's goals and sub-goals are simple, the advantage of using the BDI architecture is not clear, except that we can describe goal-oriented actions in a Prolog-like language. However, to achieve our future goal of actualizing robots that attempt to reach their destinations by forming sub-goals, we assume that we can take advantage of the BDI model in which the robot can hold or modify its intentions and maintain its sub-goals in response to environmental changes or failures of actions in the real world[3].

C. ROS and Jason

The Q.bo Lite Evo robot has motors on its wheels and sensors, which can be controlled using ROS[4]. We use ROS from Python to make the robot move and obtain information from its sensors. By processing the visual information from the sensors using libSVM, the (non-)presence of a target object is determined, and the result is passed to Jason. Jason then acquires the result as a perception and uses it for practical reasoning to determine the robot's action, i.e., selects the plan for the current goal and passes an atomic action to the Python side. On the Python side, the motor output determined based on that action is passed to ROS, and the robot acts (Fig. 5).

We are currently focused on developing a mechanism for determining actions from goals as a practical reasoning based on logic programming. Hence, object recognition using libSVM currently involves a naive method, and its precision is not so good. Improving it is our future task.

D. Design of atomic actions

Atomic actions are designed at the level of, for example, 'proceed while avoiding obstacles', and implemented on the Python side. Plans using these actions as the smallest units of action are prepared on the Jason side. As a result, when writing plans for the BDI agent, we can plan using atomic actions, which are robust against obstacles.

We implemented the following atomic actions. The heading of each entry shows the name of the action. Actions `forward_Qbo` and `search_Qbo` were implemented in our previous research[5], [6].

- `looking_Qbo:` search for an object and take an argument as an ID of the target object. First, obtain 3 images from the camera by rotating the robot's head in different directions (covering about a 160° view) and splitting each image into 3×2 areas. Next, estimate whether the target object exists in each area using SVM. If, for some areas, the target is judged to exist at a certainty higher than some threshold, then the target is considered to exist in the direction of the area with the highest certainty. If not, the target is considered not to exist. The result is returned as a perception.
- `forward_Qbo:` take an argument as a base direction, turn to the specified base direction, and move ahead a constant length. The capability to detect an obstacle with ultrasonic sensors in parallel with moving is built into this action, and for detecting an obstacle, the robot stops. After this action, information of the robot's direction and whether the robot is facing an obstacle is received as perceptions (the same holds for `search_Qbo`).
- `search_Qbo:` take an argument as a base direction, turn to the direction that is close to the specified base direction, and move to where no obstacles are found. This first involves turning to the specified direction, and if there is any obstacle in that direction, turning around little by little until facing no obstacle.

E. Decision-making program for robots

We prepared a decision-making program for our robot as a plan of the BDI agent as follows. Currently, we assume that a robot is in a room whose door is opened inward. The final goal for the robot is 'to move out of the room', and we set a goal of 'reach the exit of the room' as an intermediate step. The plan to achieve this goal consists of the following sequence of sub-goals and written in Jason, as shown in Fig. 6.

i) Access the door of the room
ii) At that point, find the entrance of the room

```
@re
+!reach_exit
<-  !reach_object(door); // access the door
    !reach_object(entry). // find the entrance
```

Fig. 6. Plan for finding exit of room

To achieve these sub-goals, we prepared sub-plan '`ro`' shown in Fig. 7 in Jason. Note that a tag beginning with '`@`' at the top of each plan shows the name of that plan. This plan,

```
@ro
+!reach_object(Object)
<-  // initially set approximate direction to object
    !set_expected_target_direction(Object);
    !search_object(Object). // main process of search

@s_td1
+!set_expected_target_direction(door)
<-  // set expected direction of door given ad hoc
    -+expected_target_direction(0).

@s_td2
+!set_expected_target_direction(entry)
:   my_current_direction(Dir)
<-  // expected direction of door entry is separated
    // by 45 degrees from current direction
    -+expected_target_direction(Dir + 45).

@so1
+!search_object(Object)
:   // if not in front of object yet
    not front_of(Object)
<-  // receive perception caused by previous action
    !get_my_current_direction;
    !get_information_of_obstacle;
    // internally call looking_Qbo to find target object
    !around_search(Object);
    // decide next action and send it to robot
    !decide_action;
    // recursively try to achieve the goal
    !search_object(Object).

@so2
+!search_object(Object)
<-  true. // if already in front of object
```

Fig. 7. Sub-plans for finding object

```
@da1
+!decide_action
:   // if target is visible, and facing obstacle
    found_target(Object) & target_direction(Dir) &
    found_obstacle
<-  // find nearest direction with no obstacle
    tcp_write(search_Qbo(Dir)).

@da2
+!decide_action
:   // if target is visible, and not facing obstacle
    found_target(Object) & target_direction(Dir)
<-  // proceed toward the target
    tcp_write(forward_Qbo(Dir)).

@da3
+!decide_action
:   // if target is not visible, and facing obstacle
    found_obstacle & expected_target_direction(Dir)
    // expected_target_direction/1 returns approximate
    // direction of the target given initially
<-  // find direction with no obstacle
    // near initial direction
    tcp_write(search_Qbo(Dir)).

@da4
+!decide_action
:   // if target is not visible, nor facing obstacle
    expected_target_direction(Dir)
<-  // proceed toward initial direction
    tcp_write(forward_Qbo(Dir)).
```

Fig. 8. Sub-plans for deciding action

working with some sub-subplans, is prepared for reaching the front of a target object. It is commonly used in both i) and ii) above. In Fig. 7 (and for other figures referred in this section), a plan is in the form of '*triggering_event* : *precondition* <- *plan_body*' (where *precondition* can be omitted), and '!', '+!' denote a sub-goal and event of goal addition, respectively. Note that not all definitions of sub-goals are given in this paper. The symbol '-+' denotes a belief revision. Atoms that appear in *plan_body*, which do not begin with any special symbol such as '!' or '-+', are atomic actions; in Jason, atomic actions are implemented using Java. The symbol '//' denotes a comment. Sub-plan 'ro' executes the following processes.

1) Initially set the approximate direction of the target.
2) Receive perception.
3) Attempt to find the target object using looking_Qbo.
4) Depending on the perception caused by 3), decide the next action (forward_Qbo or search_Qbo) as follows (see Fig. 8), and send it to the Python side via TCP (by tcp_write).
   a) As a base direction, choose the direction toward the target if it is visible. If not, choose the direction initially given as an approximate direction to it.
   b) Select forward_Qbo to proceed if the robot is facing no obstacle, or search_Qbo to avoid the obstacle as the next action. The base direction chosen in 4a) is given as the argument of the action.
5) Repeat the process from 2) to 4) by recursive call.

Between i) and ii) mentioned above, to obtain the initial direction of the entrance, the robot uses an empirical rule that the expected direction to the room entrance is separated by $45°$ from the direction of the robot when the robot reaches the front of the door (plan 's_td2' in Fig. 7). This rule comes from the fact that the angle between the exit and door was about $90°$ in this experiment (Fig. 9).

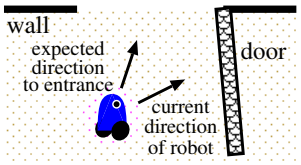

Fig. 9. Belief regarding direction to door and exit

F. Discussion on process of recognizing door

In this study, the robot recognized the door via its visual perception. However, due to the limit of its range of vision, the robot could only acquire a partial image of the door instead of that of the entire door. In this study, the room door had a horizontal slit under the doorknob, which the robot selected as a characteristic of the door in order to recognize it. We argue that the procedure described above is close to the concentration of mind in human object recognition. When a human opens a door, he/she concentrates his/her mind to a characteristic part of the door, such as the doorknob, to recognize the door and open it, instead of having awareness of the entire door. We assume that localized use of information is also useful for robots to recognize their environments.

III. Experiment and Discussion

We now discuss the results of our experiment.

A. Experimental results

First, the robot moved in the expected direction of the door given initially, found the door, and moved in the direction toward the front of the door. When it approached the front of the door, the context of plan 'so1' in Fig. 7 for reaching the front of the door was not satisfied under the following two conditions : recognizing the door by using SVM, and meeting the requirement of the ultrasonic sensors on its body, where we call these conditions "stop conditions" for plan 're' in Fig. 6. Then the goal !reach_object(door) (in the plan

're') was achieved through plan 'so2' in Fig. 7. Therefore, the robot's next goal was `!reach_object(entry)` (in plan 're') for reaching the entrance and exiting the room, and the robot moved in the direction by setting the rule of sub-plan 's_td2' in Fig. 7 to find the entry. With this knowledge, the robot moved to the entrance of the room and exited the room. This is a desirable result (Fig. 10).

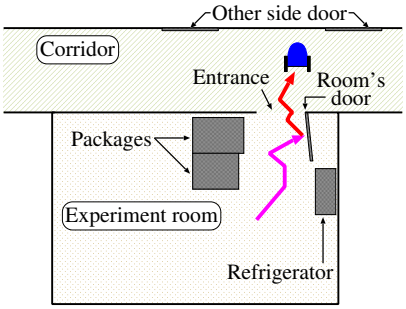

Fig. 10. Route of robot's movement during experiment (magenta line denotes executing plan for reaching front of door. Red line denotes plan for reaching entrance of room.)

Second, though the robot moved in the expected direction toward the door given initially, found the experiment room door, and moved in the direction toward the front of the door, like the first result, it found the opposite room door while it was moving toward the experiment room door. Therefore, the robot mistook the target as the opposite room door and moved toward the vicinity of the entrance of the room. At this point, the context of plan 'so1' for reaching the front of the door was not satisfied because when the robot approached the entrance of the room, the stop conditions were incorrectly satisfied by finding the opposite door and meeting the requirement of the ultrasonic sensors since its body was caught on the edge of the doorway. Then goal `!reach_object(door)` (in plan 're') was achieved with plan 'so2'. Therefore, the robot started to exit at that location by executing goal `!reach_object(entry)` (in plan 're'). As a result, the robot could not find the entrance and exit the room by using this knowledge (Fig. 11).

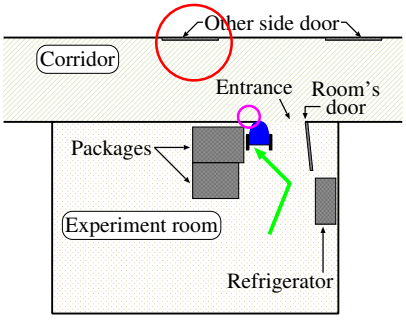

Fig. 11. Route of robot's movement during experiment (green line is executing plan for reaching front of door. Red circle is condition of recognizing door. Magenta circle is condition of meeting requirement of ultrasonic sensors.)

B. Discussion

From the second result in Sec. III-A, the robot did not move properly due to improperly implementing the stop conditions. In this experiment, we implemented the rule for the following two stop conditions : "The front of the door" by recognizing the door, and meeting the requirement of the ultrasonic sensors. In the real world, we can understand "the front of the door" empirically, but it is difficult to implement such a condition. Because the real world is a continuous space, it is difficult to uniquely determine such an abstraction.

IV. Related work

Nüchter et al. [7] proposed a method for understanding the space of the real world by labeling floors, ceilings, and walls using the inference rules of Prolog based on the vertical or parallel relationship on a three dimensional map obtained from a robot's 3D scan and using 6D SLAM [8]. However, they mentioned that using HTN planner [9] for determining a robot's movement using this logical information is a subject for future analysis. On the other hand, we used the BDI model and logic programming for determining a robot's movement, and the robot performed goal-oriented actions.

Lidoris et al. [10] developed the ACE project for developing a robot that can reach a destination without an existing map or GPS sensors and move outdoors. The main contribution of their paper was that the robot could reach a destination by asking pedestrians. Their robot is composed of a finite state machine consisting of three action-decision modules "Active, Inactive, and PriorityCheck". In contrast, we used the BDI model for the robot's action decision, and the robot could flexibly modify it's goal.

V. Conclusion

We proposed an action-decision method constructed in logical form and conducted an experiment. Through this experiment, we found the problem of stop conditions for logical representation and the necessity of appropriate stop conditions in the real world. For future work, we will devise a solution to the stop condition problem and conduct a verification experiment. We expect the robot's control of the logical representation will overcome this problem. We will also work toward our ultimate goal of actualizing robots that can logically understand the geographical relations between locations and use the knowledge in movement plans.

References

[1] M. E. Bratman, *Intention, Plans, and Practical Reason*. Harvard University Press, 1987.

[2] R. H. Bordini, J. F. Hübner, and M. Wooldridge, *Programming Multi-Agent Systems in AgentSpeak using Jason*. John Wiley & Sons, 2007.

[3] M. Fujita, H. Katayama, N. Nide, and S. Takata, "BDI robots who adapt to the diversity of the real world," *IPSJ Transactions on MPS*, vol. 5, no. 1, pp. 50–64, 2012, (In Japanese).

[4] ROS.org, "About ROS," http://www.ros.org/about-ros/.

[5] M. Fujita, Y. Goto, N. Nide, K. Satoh, and H. Hosobe, "Logic-based and robust decision making for robots in real world," in *Proc. of AAMAS '14*, 2014, pp. 1685–1686.

[6] ——, "An architecture for autonomously controlling robot with embodiment in real world," in *Proc. of Knowledge Representation and Reasoning in Robotics (workshop at ICLP 2013)*, 2013, pp. 59–71.

[7] A. Nüchter and J. Hertzberg, "Towards semantic maps for mobile robots," *Robotics and Autonomous Systems*, vol. 56, no. 11, pp. 915–926, 2008.

[8] A. Nüchter, K. Lingemann, J. Hertzberg, and H. Surmann, "6D SLAM — 3D mapping outdoor environments," *Journal of Field Robotics*, vol. 24, no. 8-9, pp. 699–722, 2007.

[9] M. Ghallab, D. S. Nau, and P. Traverso, *Automated Planning: Theory and Practice*. Morgan Kaufmann Publishers Inc., 2004.

[10] G. Lidoris, F. Rohrmuüller, D. Wollherr, and M. Buss, "The autonomous city explorer (ACE) project — mobile robot navigation in highly populated urban environments," in *Proc. of IEEE International Conference on Robotics and Automation*, 2009, pp. 1416–1422.