Subsumption-based Architecture for Autonomous Movement Planning for Planetary Rovers

Shinichi Nakasuka and Seikoh Shirasaka
Department of Aeronautics and Astronautics, University of Tokyo
Hongo 7, Bunkyo-ku, Tokyo 113, JAPAN
TEL: +81-3-3812-2111 ext.6590 FAX: +81-3-3818-7493
e-mail: nakasuka@space.t.u-tokyo.ac.jp

KEY WORDS AND PHRASES

Autonomy, machine learning, planetary rover, subsumption architecture

ABSTRACT

The paper proposes a new architecture for autonomously generating and managing movement plans of planetary rovers. The system utilizes the uniform representation of the instantaneous subgoals in the form of virtual sensor states and the autonomous generation of the subsumption type plan network, which are expected to lead to the capability to persue the overall goal while efficiently managing various unpredicted anomalies in a partially unknown, ill-structured environment such as a planetary surface.

INTRODUCTION

Among the autonomous functions required for future unmanned planetary rovers, the one especially required for such rovers will be the capabiliy to generate and manage various movement plans under partially unknown, ill-structured environments. For example, the path planning will be made based on the maps of the planet which will have been obtained beforehand by the observation from the planetary orbit, but these maps will not be so accurate and there will be in many cases lots of obstacles (such as small rocks or gaps) not represented on the maps. The path planning system, therefore, must be flexible enough to compensate for the inaccuracy of the maps, quickly respond to the unpredicted events such as collisions with the unknown obstacles, gather geographical information, and re-plan the path to the goal. This kind of flexibility will be needed in many other planning activies of the planetary rovers as well.

The paper proposes a novel architecture for autonomously generating and managing such movement plans of planetary rovers. The architecture is, basically, similar to the well-known subsumption architecture (Fig.1)[1] in the sense that the finally obtained movement plans are represented in the form of a hierachical supression/promotion network of primitive reflex actions such as "moving towards a prescribed point", "wandering about", "moving towards the reverse direction when a certain touch sensor senses an obstacle", and so on. This representation of plans is, as has been discussed in many literatures, superior in 1) robustness in the actual world because no "symbolic world model" is utilized, 2) real-timeness because no complicated symbolic manipulation is required, and 3) easiness in system integration and ex-

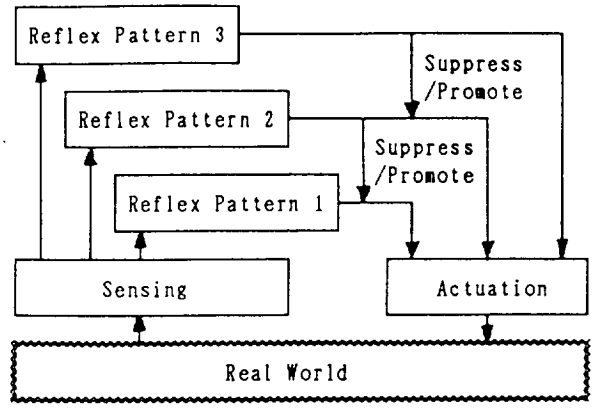

Figure 1.Subsumption Architecture

tension because a bottom-up-type system construction is quite easy. For this reason, this architecture is quite suit with the plan representation schema for rovers which move in an unstructured world. Its most significant departure from the conventional system concept is that the goal of the plan is not represented explicitly, but is achieved during the course of the interactions between the reflex actions' network (called "RAN" hereafter) and the environment. This feature is called "emergent functionality".

This architecture, however, has some difficult problems to be solved before the actual use, such as; 1) the RAN must be sophisticatedly designed by human designers so that the emergent functionality achieves the given goal, which is far more difficult task than to build a system which deals with the goal explicitly, and 2) once coded, the network is fixed during the actual operations, and the change of the environment or system itself cannot be dealt with. From these shortcomings, it can be said that the subsumption architecture cannot be employed in its original form for our objectives.

We modified and enhanced the subsumption architecture in the following three points: 1) uniform representation of the instantaneous subgoals is introduced in the form of virtual sensors so that the goal can be more explicitly persued, 2) the RAN is automatically generated by compiling the database of the actions' behavior networks obtained by machine learning, and 3) the RAN is modified during the actual operations to cope with the changes of the system and environment. The resultant system is expected to have the capability to persuit the overall goal while efficiently and more flexibly managing various unpredicted anomalies in a partially unknown, ill-structured environment such as a planetary surface.

In the following explanation, it is assumed an example task to fetch a certain object which is placed at a certain position (not at the rover position) and to carry it to a prescribed goal

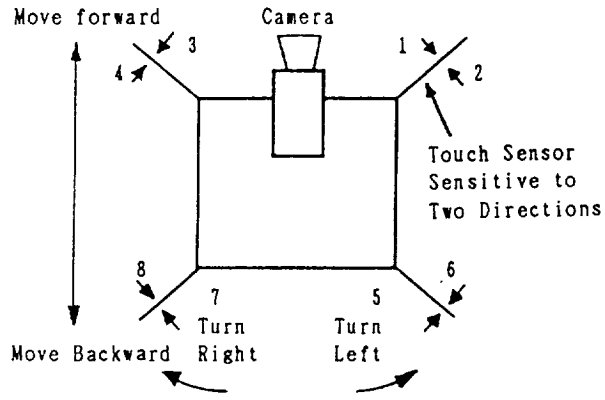

Figure 2.Schematic View of the Example Rover

position. The rover is assumed to have four touch sensors (each is sensitive to two direction forth) and one camera, and be able to turn right/left and move forward/backward as illustrated in Fig.2. It is assumed the rover knows its current position and orientation.

NEW ARCHITECTURE

Virtual Sensor States

Various actions are uniformly represented in the form of change of sensor outputs. In order for the high-level tasks such as "Plan Path" or "Write Obstacle Position to Map" to be represented in the same way, the state such as "whether the map is updated or not" or "whether there are no obstacles between the current target and the rover position" has also been represented as one "virtual" sensor state. For the example task, the eight sensor states (including three virtual sensor states) such as

Example	< Sensor State >
35	X_1. Head Angle from the Goal Direction (0 - 360°)
10	X_2. Distance from the Goal
0	X_3. Head Angle from the Object Direction (0 - 360°)
0	X_4. Distance from the Object
3	X_5. Touch Sensor Output (0 - 8) (2 directions x 4 sensors: 0 for no touch)
1	X_6. Object Carried ? (0 for Yes and 1 for No)
1	X_7. No Obstacles between Target and Current Position ? (0 for No and 1 for Yes)
1	X_8. Map Updated ? (0 for Yes and 1 for No)

Figure 3.Content of Sensor States

in Fig.3 are employed (called $X_1 \sim X_8$.) The goal state for the example problem can be represented as $(*0*0*0**)^T$.

Learning of Behavior Network

The plan management system learns when a certain action can be applied and how the action changes the sensor state. During the learning phase, the rover chooses actions randomly, which is continued until at least one of the sensor state changes. The change of the sensor state is defined as follows; for the discrete-value type states (such as $X_5 \sim X_8$), any changes of the value, and for the continuous-value type states (the other states), transitions of the value between positive, negative and zero. Examples are described in the leftmost state transitions of Fig.4. These transitions are translated into the more abstract form of state transitions (the middle forms of Fig.4) and stored in the database. In this figure, the "* (wild card) " means an arbitrary value, ">" means a positive value and "**" means that the value has not been changed from the one before the action is taken.

After accumulating large amount of such data for each action, the conventional inductive learning algorithm is applied to yield generalized form of state transition of the action (such as the rightmost form of Fig.4.) The

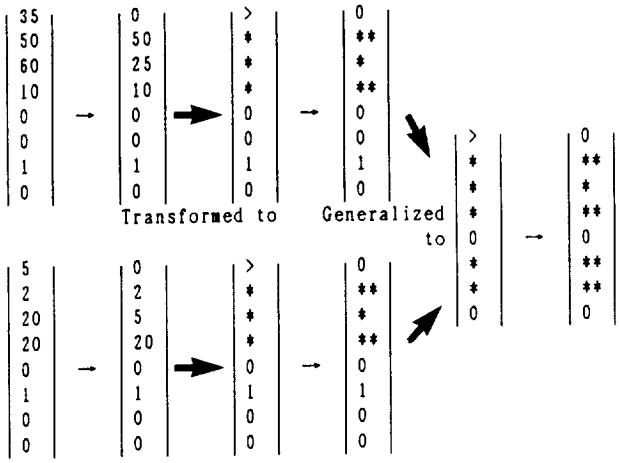

Figure 4. Acquisition and Generalization of Behavior Network

employed generalization rules include "turning a constant into a variable" rule, and "constraint deletion" rule. If the generalization between the current representation and the new instance would result in a trivial state transition (such as that all the states are represented as *), a disjunctive generalization is also introduced. Finally, several disjunctive representations are obtained for each action. These state transitions are called "Behavior Networks" in this paper.

Higher level actions such as path planning also have the behavior networks. As these networks are hard to learn and can be easily defined beforehand, they are specified by the system designer. The anomalous events during the actual movements such as collisions with obstacles are also defined as state transitions.

Compilation of Behavior Networks

After behavior networks of all the actions become mature, they are compiled into a subsumption type plan network. The major tasks of this compilation are the identifications of sensor stimuli for each action to be fired and the extraction of priority relationships between the actions. The following rules are observed in constructing the plan network.

(1) Actions are defined in the form of "continue action A1 until X_k becomes a certain constant c." Therefore, for the "turn right" action, several variations of actions are generated such as "turn right until the head angle from the goal direction becomes zero" or "turn right until touch sensors sense no forth", and so on.

(2) The actions whose consequences match the goal state are considered as candidates of the lowest level of the plan network.

(3) If taking a certain action (say A1) requires that a certain state be a certain value (0 or other integers), then the action (say A2) whose consequences satisfy this precondition is categorized as a candidate of action

which must be performed before A1 (in other word, whose firing suppresses the activation of A1.) The preconditions of A1 which are not explicitly satisfied by A2 are registered as the stimuli for firing A1. Then all the consequence states of A2 are matched with the precondition states of A1, and the precondition states of A2 are replaced with the values obtained by this matching.

(4) Many hierarchical relationships will be acquired in the above processes. From these, the best plan network is obtained by searching the space of all the combinations, based on the following criteria;

- The network does not have any loops.
- The network can lead the system to the goal state from arbitrary states.

Figure 5 describes the obtained plan network for the example problem. In this figure, the wave line shows the "Supression Signal." For example, the action "MF(X_4=0)" (stands for "move forward until X_4=0") must be performed preferentially if X_4=0 is not satisfied when trying to start action "TR(X_1=0)" or "TL(X_1=0)". When trying to start "MF(X_4=0) ", if X_3=0 is not satisfied, then the action "TR(X_3=0)" or "TL(X_3=0)" is performed according to the sign of X_3. In this way, the plan network takes into account the priority relationships between actions and the anomaly handling (such as separating from a obstacle when a touch sensor finds it) as well. For example, if the rover, during a certain action (say A1), collides with an obstacle (X_7 and X_8 become 1), which first triggers the action "write obstacle position to the map (WO)" to change X_8 to 0, and then triggers "plan path (PP)" to change X_7 to 0. Then the system resumes A1, and if another action with higher priority is not triggered, action A1 is continued. Please note that as a side effect of the WO and PP actions, the states $X_1 \sim X_4$ will be changed.

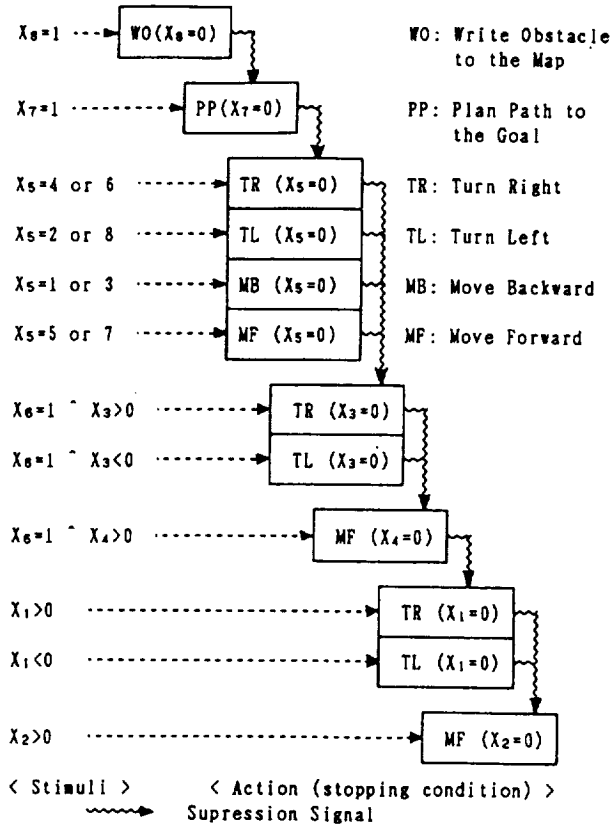

Figure 5. Obtained Plan Network for the Example Problem

If the consequence of a certain action is found inconsistent with the learned behavior network, then the learning of the correct behavior network is re-initiated for the specific action, which also triggers the recompilation of the behavior networks into the plan network. With this technique, the system has the flexibility to adapt itself to the change of the environment or the system itself.

CONCLUSIONS

An architecture to manage the rover movement plans under ill-structured, partially unknown environments has been proposed. Simulation studies have indicated the effectiveness of the architecture, and experiments using an actual rover-type vehicle is now being performed.

REFERENCES

[1] Brooks, R.A.,1986. A Robust Layered Control System for a Movile Robot, *IEEE J. Robotics and Automation*, Vol.RA-2, April: 14-23.