# Motion Design for Autonomous Mobile Manipulator based on Programming Style "Action Primitive"

**Keiji Nagatani**

Systems Engineering Dept.
Okayama University
3-1-1 Tsushima-naka
Okayama 700-8530, JAPAN
keiji@sys.okayama-u.ac.jp

**Yutaka Tanaka**

Systems Engineering Dept.
Okayama University
3-1-1 Tsushima-naka
Okayama 700-8530, JAPAN
field@sys.okayama-u.ac.jp

## Abstract

Our research goal is to realize an intelligent autonomous motion for mobile manipulator. A model task for mobile manipulators, "Returning Books to Bookshelf" was chosen as our research task. To realize the task, we adopt a programming style named "action primitive" for programming of robot controller. Each action primitive is a basic motion control program, and the whole motion of the robot is generated by executing a sequence of action primitives step by step. This paper describes motion design for above research task based on action primitive concept, and reports simulation results of the designed motion.

## 1 Introduction

### 1.1 Research Objective

Recently, a demand of intelligent motion of mobile manipulator becomes popular because of both benefits, mobility and manipulatability. However, to control mobile manipulator has peculiar difficulties (e.g. positioning error of end effector caused by positioning error of mobile base etc.), so intelligent and complicate motion is far from practicable use.

On the other hand, a working area for robots has been expanding from structured environments (e.g. factory area) to unstructured environments (e.g. office buildings and home). In unstrucutured environments, intelligence and autonomy become more important to cope with unexpected situations.

On the basis of above background, we chose our research task as "Returning Books to Bookshelf" by autonomous mobile manipulators. Under the process of realizing the task, we may find what is the most important technology for controlling mobile manipulator, and also it may work in actual libraries. In first step, a target environment is our laboratory.

In this paper, we describe an analysis and design of robot's motion to realize the task, and report a programming method based on **action primitive**[1].

### 1.2 Related Works

Many task based researches for mobile manipulator had been reported recently.

"Door Opening Motion" was one of the model task for application research because of it's difficulty and demand. Prof. Schmidt [2], Dr. L.V. Litvintseva [3] had researched "Door Opening Motion" by their own approach. In 1997, The author also had realized the motion in a real environment[4].

On the other hand, Prof. Khatib researches cooperative motion of several mobile manipulators to carry one object[5]. Messenger robot in Prof. Nakano's group takes messages and document to another places [6].

Generally, most of all research of mobile manipulators was done in structured environment, and current robotics research had not realized intelligent and robust motions yet. In our research, we will set a specific task in unstructured environment, and we make clear a bottleneck of intelligent motion for mobile manipulator.

## 2 Action Primitive

To realize an intelligent motion of autonomous robots, suitable choices of actuators and sensors are necessary according to robot's situations. Therefore, we adopt a programming style "Action Primitive" for our research task to program our robot controller. In this section, we explain a general brief of this programming style.

## 2.1 Division of Motion

For benefit of implementation, a complicates motion should be divided into several simple motions along time axis. A guideline of the division is as follows.

1. switch of actuators
2. switch of sensor feedback

We call a program unit that realizes one primitive motion as "**action primitive**'. A sequence of action primitives enables an entire complicate motion.

## 2.2 Responsibilitiy of Motion

It is very difficult to keep a consistency of robot motion by executing several independent programs simultaneously. In the worst case, several programs generate a conflict of actuator control. Therefore, one responsible program unit that takes care of whole robot motion is necessary for an intelligent robot motion. Usually, one coordinate program mediates several actuation programs that are executed in parallel. In our case, a coordinate program just selects a suitable action primitive, and each "action primitive " corresponds to responsible whole robot motion.

## 2.3 Coping with Motion Errors

In unstructured environment, it is very difficult for robots to obtain a perfect model of the target environment, and usually robots' motion generates motion errors. To cope with the error, each action primitive should have sensor feedback systems. It receives motion parameters and environment model from coordination program, and modifies control input by sensor information in realtime. Once the motion is almost executed correctly, the error must be small. Therefore the modification is done by linear feedback.

## 2.4 Watching Unexpected Situation

Even if the robot adjusts its motion by linear feedback, unexpected structural differences may happen in unstructured environment (e.g. unexpected obstacles on robot's path). To cope with the problems, each action primitive has a mechanism to check robot's circumference. If it finds a big discrepancy between environment model and real, the action primitive terminates the robot's motion. Then, it restores the robot situation back to the beginning of the action primitive, and acknowledges "**fail**" to coordination program (Shown in next subsection).

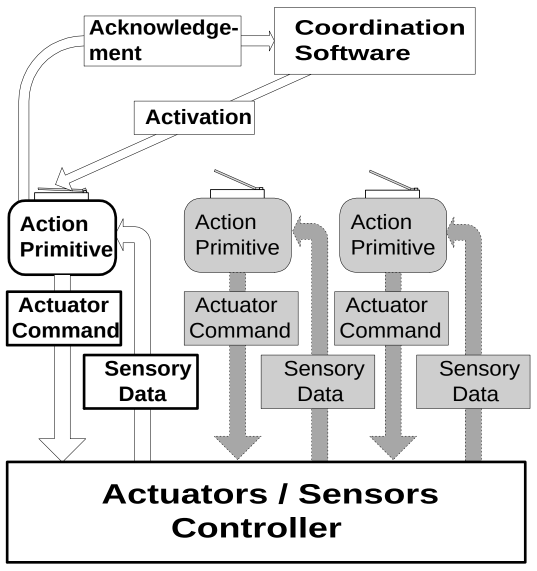

Figure 1: Execution of Action Primitives

## 2.5 Termination of Action Primitive

Three robots' situations is considerable at the end of execution of action primitive as follows.

| | |
|---|---|
| **Success** | : Target motion is completed |
| **Fail** | : Restoring situation at the beginning |
| **Give up** | : Impossible to continue the motion |

When one of above condition is satisfied, the action primitive returns robot's situation (Success, Fail or Give up) to the coordination program, and terminates its motion. Note that "**Give up**" is the situation that the robot can never solve the problem without human's help (e.g. the robot is stacked into undetected steps).

## 2.6 Coordination Program

Total motion is realized by a sequential execution of suitable action primitives. A role of coordination program is to plan a sequence of suitable action primitives, and executes them step by step (Figure 1).

To plan a sequence of action primitives, we prepared a big network of every possible robot's status that are transited by executions of action primitives. We call it **Action Network**. Each action primitive has a cost, and a sequence of action primitives is planned by a simple graph search on the action network. This graph search is executed before action primitives' execution.

When the action primitive detects a discrepancy between model and real environment by sensor information, it sends "**Fail**" to the coordination program. In this case, the coordination program re-plans a new sequence of action primitives from the current situation to the goal. It is a reasonable mechanism to cope with unexpected situations.

### 2.7 Feature of Action Primitive

In this programming style, a responsibility of coordination program is very small. The important roles, such as coping with motion errors, are done by each action primitive. If there is no acknowledgment of "**Fail**" from each action primitive, a motion is completed by a sequential execution of planned action primitives step by step. The other words, the feature of this programming style is to increase an importance of low level of programming and to restrict a role of coordination programs.

Of course, once there is an acknowledgment of "**Fail**" from an action primitive, the coordination program acts very important role to re-plan a sequence of action primitives.

## 3 Motion Design based on Action Primitives

We adopt the programming style "action primitive" to realize our research task "Returning Books to Bookshelf". In this section, we discuss a motion design of task.

### 3.1 Task Analysis

Referring human motion, our research task requires robot's mobility, manipulatablility and object handling. Therefore our target robot should be mobile manipulator with robot hand.

We also add one condition. Locations of bookshelves are known, but an exact location of the target book in the shelf is unknown. Therefore, the robot should search a returning location of the book.

### 3.2 Dividing Motion and Generating Action Network

First, we divided the target task into several action primitives. Usually, a human picks up a book, estimates where it should be returned by checking its tag, takes it to the bookshelf, finds the location based on a sequence of books' numbers in bookshelf and inserts the book into it.

On the basis of above human motion, an action network for the target task is described shown in Table 1. Required action primitives are "RECOGNIZE", "PICK_UP ", "MOVE_TO", "SEARCH", "APPROACH", "INSERT" and "RELEASE ". Detail designs are shown in following section. If there is no interrupt, the robot executes from (0) to (8) sequentially.

| | Name of Action Primitive | Parameter | Next (OK) | Next (Fail) |
|---|---|---|---|---|
| (0) | START | | 1 | |
| (1) | RECOGNIZE | BOOK | 2 | 9 |
| (2) | PICK_UP | BOOK | 3 | 9 |
| (3) | MOVE_TO | BOOKSHELF | 4 | 10 |
| (4) | SEARCH | TARGET_POS | 5 | 10 |
| (5) | APPROACH | TARGET_POS | 6 | 4 |
| (6) | INSERT | BOOK | 7 | 4 |
| (7) | MOVE_TO | GOAL | 8 | 12 |
| (8) | GOAL | | | |
| (9) | GIVEUP | | | |
| (10) | MODIFY | PATH | 3 | 11 |
| (11) | MOVE_TO | START_POS | 13 | 13 |
| (12) | MODIFY | PATH | 7 | 11 |

Table 1: Action Network of "Returning Books to Bookshelf" Motion

### 3.3 Design of Each Action Primitive

#### 3.3.1 PICK_UP

An objective of action primitive "PICK_UP" is to pick up the target book by mounted manipulator.

First step of our research, an initial book location is fixed on a table equipped with robot's body, and human put a book on it precisely. Then, "PICK_UP" motion is done by teaching-playback motion without sensor feedback.

#### 3.3.2 RECOGNIZE

An objective of action primitive "RECOGNIZE" is to get information of target book, and determine a position of returning place.

Usually, each book in libraries has its own number that indicates unique location of it. Therefore, the robot can find a returning location by recognizing a tag of the book using a vision sensor. Database of books' information is required in this action primitive.

#### 3.3.3 MOVE_TO

An objective of action primitive "MOVE_TO" is to navigate the mobile base to the target bookshelf or its goal.

Environment information and destination are required for path planning before execution of this action primitive. The program makes the mobile base follow the planned path by sending locomotion commands to locomotion function in realtime.

To localize the mobile base, odometry is used for our positioning system. However, positioning error is accumulated by wheels' slippage and uneven ground. To cope with the positioning error, the robot compares model of environment with actual information detected by sensors, and adjust its estimated position.

#### 3.3.4 SEARCH

An objective of action primitive "SEARCH" is to search a location of insertion place of the book in target bookshelf.

The robot has an information of rough returning place of the book. However it requires a precise location. To find it, the robot checks a continuation of serial numbers of books in the bookshelf by vision sensor. Information of books' back cover is required for this action primitive.

To cope with a narrow view angle of vision sensor, we mount a CCD camera on the top of manipulator (hand-eye system), and control the manipulator to cover a wide area for checking books' back covers. Template matching is used for recognizing insertion place.

#### 3.3.5 APPROACHING

An objective of action arimitive "APPROACHING" is to approach the robot hand to returning place.

A motion at the top of manipulator has a positioning error, particularly it is affected by an orientation error of the mobile base. To cope with the error, it adjusts a trajectory of the robot hand based on sensor information.

#### 3.3.6 INSERT

An objective of Action Primitive "INSERT" is to insert the target book into the exact location of the target bookshelf.

This motion is very difficult by simple two fingers' hand. Therefore, a special structure for returning action is necessary. The motion is generated by teaching playback motion without sensor feedback.

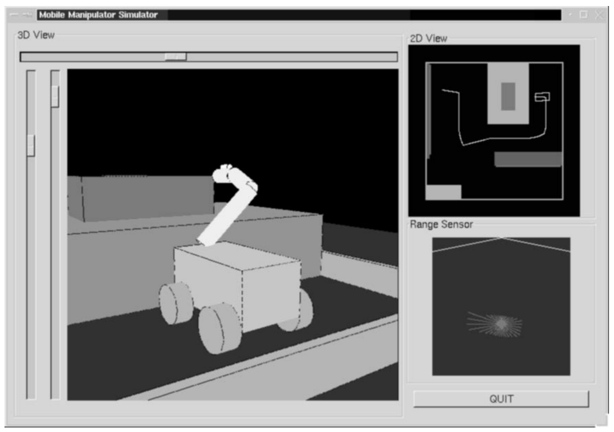

Figure 2: Mobile Manipulator Simulator

## 4 Simulation and Implementation

Currently, we are setting up a mobile manipulator to execute above designed motion in real environment. At the same time, we are developing a simulator for mobile manipulator to confirm its motion program. In this section, we introduce the simulator, and describe an implementation of action primitives that is working in the simulator.

### 4.1 Simulator of Mobile Manipulator

To confirm a behavior of action primitives and target robot, we developed a simulator of mobile manipulator. An appearance of it is shown in Figure 2.

The simulator receives (1) desired velocity and angular velocity of mobile base, and (2) desired joint angles for mounted manipulator, from a user program. Then it calculates a position and orientation of mobile base, and a pose of manipulator. At the same time, an ideal range sensor information is sent back to user program. 3-D Display of the simulator is constructed by gtk and open GL in Linux Operating System.

A benefit of the simulator is that an interface between user program and "simulator" is the same protocol as an interface between user program and "actual robot ". Therefore, the program developed for the simulator can work on actual robot. A structure of the simulator is shown in Figure 3.

The purpose of the simulator is not a physical simulation, but a confirmation of robot's motion by user programs. Therefore, it does not consider a force of the manipulator, a vision sensor and positioning error.

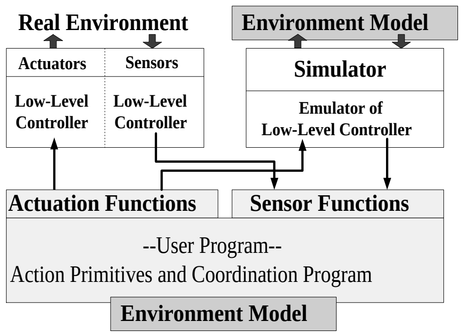

Figure 3: Structure of Simulator

## 4.2 Implementation of "MOVE_TO"

Action primitive "Move_to" requires a function of path planning from current robot position to destination. In our task, the robot has information of target environment. So we chose "Reduced Generalized Voronoi Graph (RGVG)" [7] for path planning.

In 2D case, Generalized Voronoi Graph(GVG) is expressed by

$$G(x) = \big[(d_1 - d_2)\big](x) = 0 \qquad (1)$$

where $d_1$ and $d_2$ are the distance to two closest convex objects. RGVG is a subset structure of GVG which unnecessary edges are eliminated. The unnecessary edge is the edge attaching to the objects. By using RGVG, we use following procedure for path planning.

1. Connect from start point to the closest point P1 on RGVG
2. Connect from goal point to the closest point P2 on RGVG
3. Plan a path between P1 and P2 on RGVG

An example of path planning result is shown in Fig.4.

The planned path is not a straight line, so it is approximated by serial short segments. The mobile base can follow the path by switching these small segments as reference lines.

Currently, action primitive "MOVE_TO" is working on developed simulator, and Figure 2 shows one scene of it. It shows that the mobile base has just arrived at the front of target bookshelf in 3-D view, and planned path is displayed in upper-right box.

## 4.3 Implementation of "SEARCH"

Action primitive "SEARCH" requires to search the target location to insert the book. To cover a wide

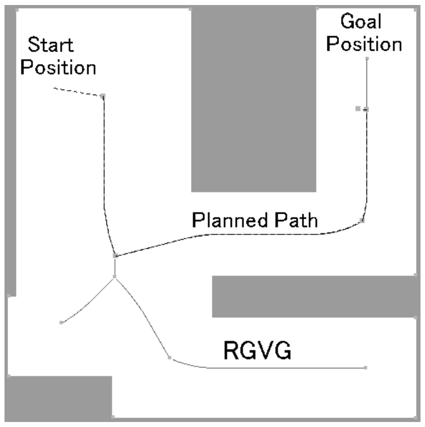

Figure 4: Reduced Generalized Voronoi Graph

area of the surface of the target bookshelf, manipulator with CCD-camera is controlled. To realize the motion, we are now constructing small-size manipulator for the robot and control board of it. At the same time, we are implementing approach motion in simulation world.

## 4.4 Implementation of "Coordination Program"

Currently, "MOVE_TO" and "SEARCH" are implemented in our simulator, and the other action primitives are implemented as dummy programs. Also, we have implemented "coordination program" to control above action primitives based on a concept of section 2.6.

## 5 Future Work

In this paper, We described a design of the intelligent motion "Returning Books to Bookshelf" for mobile manipulators, and reported an implementation of action primitives "Move_to" and the coordination software in simulator of mobile manipulator.

To realize such motion in real environment, we still have many problems (e.g. an error problem at the top of the manipulator). Our future work is to find and solve general mobile manipulators' problems by implementing a specific task on real robots.

## Acknowledgments

Prof.Yuta (Univ. of Tsukuba) helped me to construct a concept of Action Primitive. I would like to thank him.

## References

[1] K. Nagatani and S. Yuta, "Designing strategy and implementation of mobile manipulator control system for opening door," in *Proc. of IEEE International Conf. on Robotics and Automation*, pp. 2828–2834, 1996.

[2] K. Azarm and G. Schmidt, "Integrated mobile robot motion planning and execution in changing indoor environments," in *Proc. of IEEE/RSJ Int. Conference on Intelligent Robots and Systems*, pp. 298–303, 1994.

[3] L.V.Litvintseva, T.Tanaka, T.Hirabayashi, Y.Watanabe, K.Ogino, M.Hamuro, K.Saeki, K.Yasukawa, and S.V.Ulyanov, "Intelligent mobile robot for service use in office buildings developed through university-industry cooperation (uic)," in *Proc. of 3rd Robotics Symposia (In Japanese Conf.)*, pp. 197–202, 1998.

[4] K.Nagatani and S.Yuta, "Autonomous mobile robot navigation including door opening behavior (system integration of mobile manipulator to adapt real environment)," *Field and Service Robotics, Springer-Verlag*, pp. 195–202, 1998.

[5] O. Khatib, "Robot planning and control," in *Proc. of International Workshop on Some Critical Issues in Robotics*, pp. 65–80, 1995.

[6] K. Hanada, T. Takahasi, and E. Nakano, "Software architecture for constructing a mobile robot with highly independent modules," in *Proc. of The Fourth Int. Conf on Control, Automation, Robotics and Vision (ICARCV'96), Vol.2*, (Singapore), pp. 858–862, 1996.

[7] K. Nagatani and H. Choset, "Toward robust sensor based exploration by constructing reduced generalized voronoi graph," in *Proc. of IEEE/RSJ Int. Conference on Intelligent Robots and Systems*, pp. 1687–1692, 1999.