Integration of Operator Learning, Planning and Execution in a Heterogeneous Multi-Robot System

YAMADA Seiji and IWASAKI Ryouhei
CISS, IGSSE, Tokyo Institute of Technology
4259 Nagatsuta, Midori, Yokohama, 226-8502, JAPAN
yamada@ymd.dis.titech.ac.jp

Abstract

This paper describes a practical framework in which generation of operators, planning and an execution of a plan are integrated for a heterogeneous multi-robot system. In our framework, first we design literals to describe an environment. Next a teacher directly manipulates real mobile robots to achieve a task in the environment, and a sequence of the pairs of sensed data and an executed action is obtained. By analyzing the sequence, a system is able to automatically generate an operator by constructing a precondition-list, an add-list and a delete-list from them. Once the operators are acquired, a system can apply a complete and sound planner GraphPlan to generate a plan. The generated plan is executed by instance-based learning with a Nearest Neighbor method. This execution of a plan utilizes a sequence of the pairs of sensed data and an executed action as a set of instances. Thus a system integrates operator generation, planning and the execution of a plan. The feasibility of our framework is verified through experiments in which two heterogeneous mobile robots cooperatively achieve a task.

1. Introduction

One of important technologies for bridging between AI(Artificial Intelligence) and robotics is *planning*. In the AI planning, operators describing robot actions are given in advance, and the automatic generation of an operator sequence which can transform an initial state to a goal state[3, 6]. An obtained sequence of operators is called a *plan*. An environment in which a robot works is described with predicate calculus, and a problem consisting of an initial state and a goal state is given. In this context of AI planning, planning is just to search for a plan, and most of the studies in AI planning have been done on purpose of developing an efficient search algorithm. However, planning is essentially a sub-system of a whole framework in which a robot recognizes an environment, describes the environment and operators, generates a sequence of actions and executes it. Though searching for a plan is significant, automatic generation of operators recently has been recognized as very important and difficult for a practical application of planning to robotics. Nevertheless few attempt has been done for generating operators in robotics and AI[8, 7].

A multi-robot system is also an important issue in the view of engineering because many practical robotics systems are consisting multiple robots. However most of the investigated multi-robot systems are homogeneous in the sense that all the robots have identical sensors and actuators. Though such systems are robust to breakdown of some robots, they have no adaptation that individual robot plays a different role depending on its ability. In a real environment, a multi-robot system inevitably becomes heterogeneous since no physical robots are completely identical and they are often different in the function. Thus we need a framework for a heterogeneous multi-robot system.

Hence we propose a practical framework in which generation of operators, planning and the execution of a plan are integrated for a heterogeneous multi-robot system. In our framework, first we design literals to describe a environment. Next we actually manipulate a real mobile robot to achieve a task in the environment, and a sequence of the pairs of sensed data and an executed action is obtained. By analyzing the sequence, a system is able to automatically generate an operator, and apply an efficient planner to generate a plan. The generated plan is executed by instance-based fashion. As a result, a system integrates operator generation, planning and the execution of plans. We verify the feasibility of our framework through experiments in which two different types of mobile robots cooperatively achieve a task.

Wang proposed a method to generate and refine operators using a machine learning approach[8]. A version space method and a large number of training examples were utilized for operator generation. The experimental results supported that the approach is promising. Unfortunately the environment was symbolic, and no real environment in which a physical robot actually works was dealt with. In contrast with the study, we apply our framework to a real and physical environment using real mobile robots.

Schmill, at. el [7] studied learning for classifying sensed data into action classes using an efficient inductive learning algorithm C4.5. They made experiments in a real environment using a physical mobile robot. However they did not propose a concrete method to generate an operator in their research. Our framework not only deals with a real environment but develops a method to generate operators.

2. System overview

We propose a framework in which a mobile robot is able to learn necessary operators by user's simple and direct teaching to do planning and execute a plan. Fig. 1 shows the overview of our system. It has three modes: a *learning mode*, a *test*

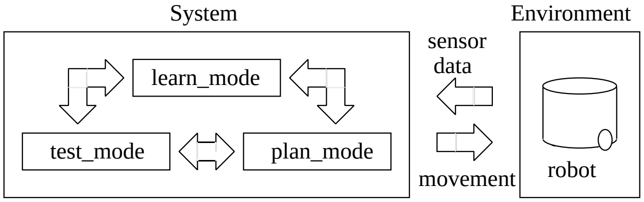
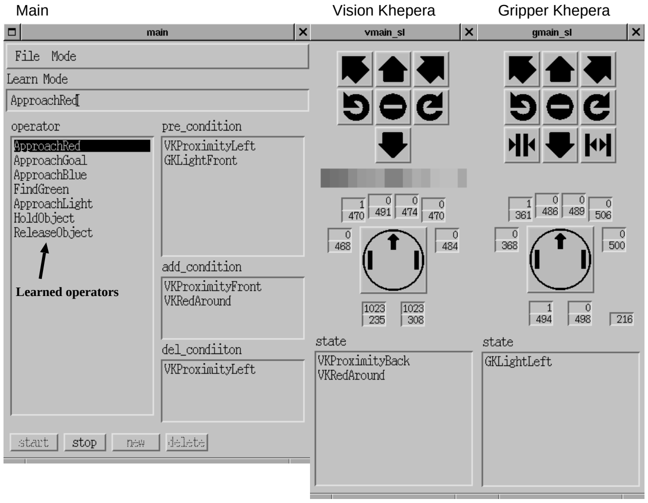


Figure 1 System overview



ApproachRed

operator
ApproachRed
ApproachGoal
ApproachBlue
FindGreen
ApproachLight
HoldObject
ReleaseObject

pre_condition
VKProximityLeft
GKLightFront

add_condition
VKProximityFront
VKRedAround

del_condiiton
VKProximityLeft

state
VKProximityBack
VKRedAround

state
GKLightLeft

Figure 2 GUI

mode and a *planning mode*, and a user(teacher) can change the modes anytime. Thus a user can teach a robot in a learning mode as monitoring its execution in a test mode. Also a user can teach a robot immediately when a robot fails planning in a planning mode. A user and a robot can cooperate interactively to generate and execute a plan. User-frendly GUI(Fig. 2) is provided for a user to change modes and monitor various information on an operator and sensed data.

- *A learning mode*: A user directly teaches a robot, and a robot can learn executable operators by the teaching. In the teaching, a user directly manipulates a mobile robot through GUI to perform a given task like remote control. Furthermore a system is able to delete and teach again an operator which failed to be executed in a test mode.
- *A test mode*: A user checks whether a generated operator in a learning mode can be executable or not.
- *A planning mode*: Using generated operators, a given initial state and goal state, a system generates a plan.

3. Environment model, operator and planning

3.1. Environment model

In our system, an environment is described a conjunction of literals obtained from sensed data. Thus this set of literals is called an *environment model*. For simplicity, we use no variable for describing the literals. These primitive literals are designed in advance, and an operator is automatically acquired through teaching. For example, $ProximityFront$, $ProximityRight$, $LightBack$ are literals for an environment model. A snapshot of an environment model is called a *state*. A planning problem is described with a pair of an initial state and a goal state. An *initial state* means an initial environment model when a robot starts to work, and a *goal state* means a final environment model which a robot should approach.

3.2. Operator

An operator is basically described with a *precondition-list*, an *add-list* and a *delete-list*. The precondition-list means a set of literals which are necessary to execute the operator. The add-list means a set of literals which become true after the execution of the operator, and the delete-list stands for literals are deleted from the state after the operator execution. Thus an operator application to an environment model transforms a state. This is a popular description in traditional AI planning like STRIPS[3, 6].

Additionally we introduce an *instance set* for executing an operator using a robot in a real environment. The instance set consists of pairs of sensed data and an executed action by teaching. As mentioned later, a Nearest Neighbor method is utilized to execute an operator, thus an instance set is necessary. Examples of an operator and an instance set are shown in Fig. 3.

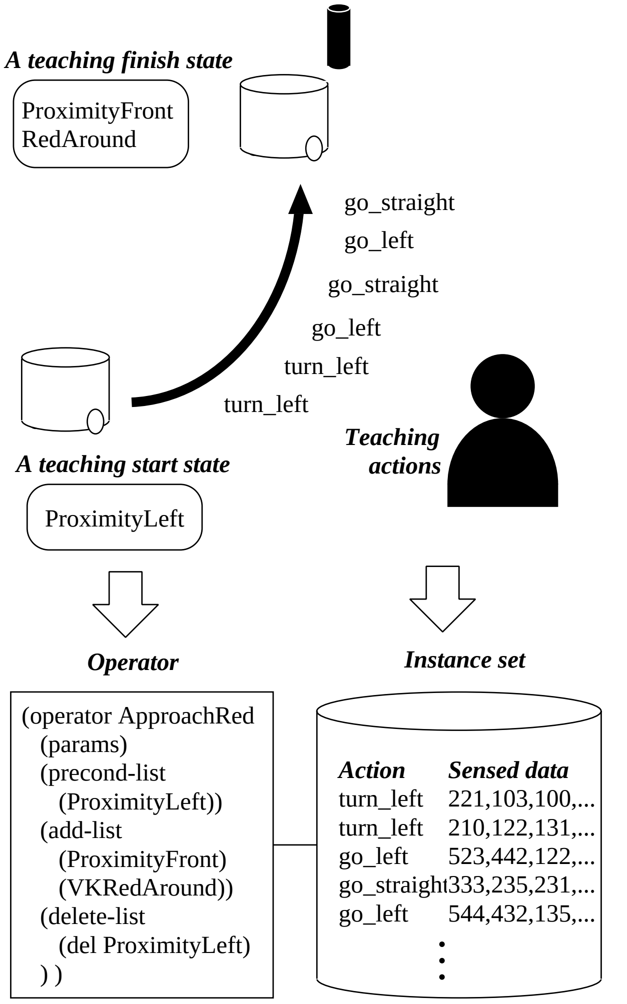
3.3. Planning

As mentioned before, planning is generation of an operator sequence to transform an initial state to a goal state. Since development of an efficient planning algorithm is not a purpose of this study, we employ Graphplan[1], one of complete and the most efficient planners. Graphplan describes a plan as a directed graph having causality among operators, and the planning is dealt with as manipulation of the graph. Using Graphplan, a system is able to generate an adequate plan efficiently.

We explained the overview of our framework above. In the following sections, we describe how to generate and execute an operator in more detail.

4. Automatic generation of operators

Fig. 3 shows learning of operators by direct teaching by a human teacher. He/she teaches a mobile robot by directly manipulating it like remote control. Using GUI(Fig. 2), a teacher can easily control a mobile robot, and monitors the behavior, sensed data and literals.



```
(operator ApproachRed
  (params)
  (precond-list
    (ProximityLeft))
  (add-list
    (ProximityFront)
    (VKRedAround))
  (delete-list
    (del ProximityLeft)
  ) )
```

Action	*Sensed data*
turn_left	221,103,100,...
turn_left	210,122,131,...
go_left	523,442,122,...
go_straight	333,235,231,...
go_left	544,432,135,...
⋮	

Figure 3 An operator and the learning

4.1. Checking literals

Since GUI shows a current state, a teacher can recognize literals which are true in an environment through it. However, checking whether a literal is true or false from sensed data is difficult because we hardly design a stable and accurate function to classify the literal's value with sensed data.

Thus we obtain instances consisting of sensed data and literals' values, and utilize a 1-Nearest Neighbor method[2] for determining literal value. Also CNN(Condensed Nearest Neighbor rule)[5] and RNN(Reduced Nearest Neighbor rule)[4] are applied to reduce instances. Furthermore significant attributes are given to a system as background knowledge. This instance-based approach is experimentally found better than designing a classification function.

4.2. Generating an operator

At the beginning of teaching, the state is stored as a *teaching start state*. A teacher selects next action on a current state, tell it to a robot, and the robot executes it in a real environment. This direct teaching is repeated until a teacher stops it. A teacher monitors literals and stops teaching when an execution of a single operator finishes. When a teacher stops teaching, the state is stored as a *teaching finish state*. The teaching start state and the teaching finish state are inputs for an operator generation procedure.

During teaching, a robot constantly stores the pairs of sensed data and an executed action as an instance set for instance-based operator execution. These pairs are utilized to execute the learned operator when a plan including it is executed.

An operator is automatically generated using the following procedure in a learning mode. The inputs are a teaching start state and a teaching finish state, and the output is an operator.

1. A teacher describes a name of an operator.
2. A system generates a precondition-list including all the literals in a teaching start state.
3. A user teaches a robot by directly operating it step by step through GUI. During this teaching, a system stores the pairs of sensed data and an executed action as an instance set for later instance-base operator execution. At last, a user stops teaching.
4. By comparing a teaching finish state S_f with a precondition-list P, the add-list and the delete-list are generated. The add-list is obtained by $\overline{P} \cap S_f$ and the delete-list is generated from $P \cap \overline{S_f}$.

5. An execution of an operator

As well as operator generation has not been studied actively, a method to execute a plan using a physical robot has not been developed so much. In our framework, we propose an *instance-based operator execution* of a plan. The procedure is described like the following.

1. By observing an environment, an add-list and a delete-list are checked. If all the literals in the add-list are true and no literal in the delete-list is true, the execution of the operator is finish.
2. A robot selects an action using a 1-Nearest Neighbor method[2] with the operator's instance set. It compares current sensed data with data in the instances, and the most similar instance is determined. The action of the selected instance is executed by a robot.
3. Go to 1.

A k-Nearest Neighbor method outperforms a 1-Nearest Neighbor method in the case that a large number of instances are available. In this instance-based operator execution, a few instances are obtained in a learning mode because an operator has a relative small number of instances. Thus, due to the cost of a k-Nearest Neighbor method for selecting the best k instances, we employed a 1-Nearest Neighbor method in our research.

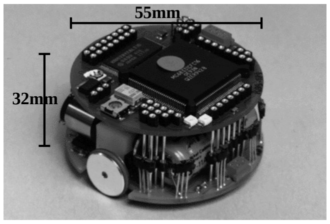


Figure 4 Khepera.

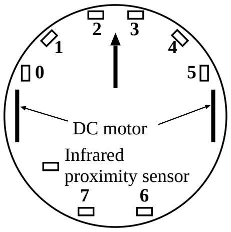
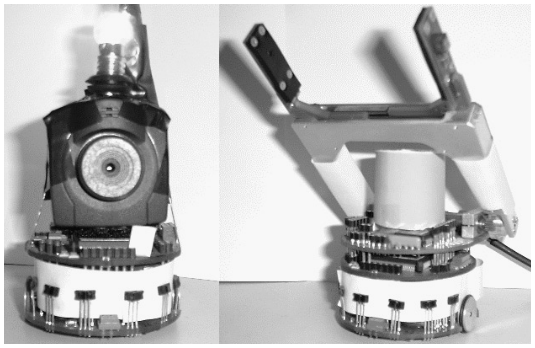


Figure 5 Sensor positions.



(a) VisionKhepera

(b) GripperKhepera

Figure 6 Heterogeneous multiple robots.

6. Experiments

In this section, we describe experiments for investigating the feasibility of our framework in a real environment where heterogeneous mobile robots work.

6.1. Heterogeneous multiple robots

We employ a small mobile robot Khepera(Fig. 4) which is very popular in AI and robotics researches. Khepera has eight IR proximity sensors and light sensors at the same positions as shown in Fig. 5. Khepera can recognize an object only within 20mm around using an IR proximity sensor, and a far light using a light sensor.

Two types of Khepera mobile robots(Fig. 6) are used for experiments. The VK(VisionKhepera) has a color CCD camera (270,000 pixels) and a light on the top like Fig. 6(a). VK recognizes the direction of a far colored object, however it can not grasp the object because of no gripper. In contrast with VK, the GK(GripperKhepera)(Fig. 6(b)) can grip an object with its gripper, and can not recognize a far colored object because of no CCD camera. GK has a green cylinder on the top which can be recognized by VK. Also since VK has a light, GK can recognize its direction with the light sensors. In the sense that these mobile robots have different sensors and actuators, a multi-robot system consisting of them is heterogeneous.

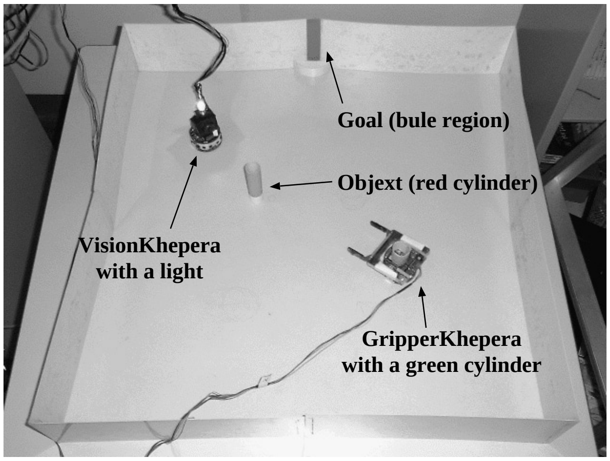

Goal (bule region)
Object (red cylinder)
VisionKhepera with a light
GripperKhepera with a green cylinder

Figure 7 Experimental environment.

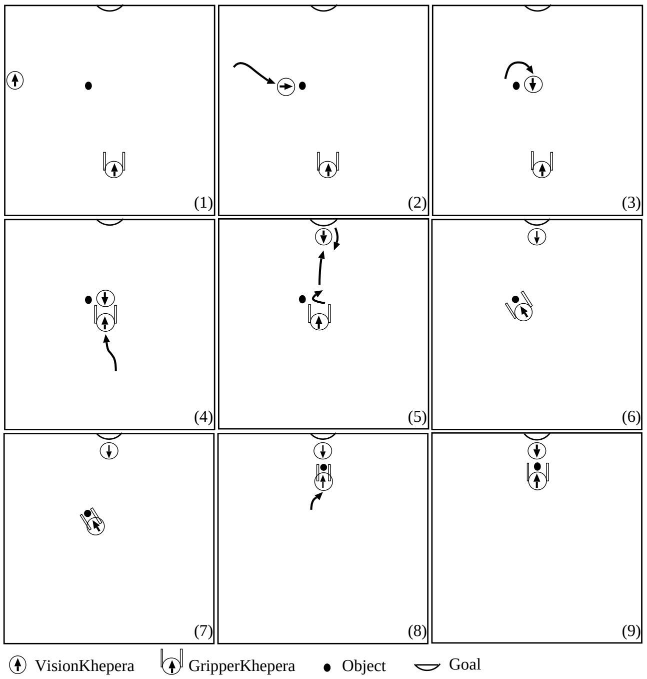


Figure 8 Cooperative object gathering.

6.2. Experimental environment

The two robots work in an environment shown in Fig. 7. The environment is a 72cm×72cm square area surrounded with white walls. There is a red cylindrical object and a blue goal region on the wall, which VK can recognize their directions and GK can not do so. Since the object, the goal region and GK have different colors, VK is able to distinguish them. In this environment, even a simple task like moving a red object to a goal needs cooperation between the two mobile robots. Hence this environment is valid for investigating our framework.

6.3. Cooperative task

We verify the validity of our framework with a cooperative task in a heterogeneous multi-robot system. The task is that two mobile robot move a red object to a blue goal region. A single robot VK or GK can not achieve this task because VK can not grasp the object and GK can not find it. Thus one of solutions to this task is shown in Fig. 8. We define the following operators for the solution, however the precondition-list, the add-lists and the delete-lists are learned automatically. In the following operators, (i)–(j) means the operator corresponds an action from (i) to (j) in Fig. 8.

- *ApproachRed*: VK rotates and approaches a red object. (1)–(2)
- *FindGripperKhepera*: VK does wall-following around the red object until GK becomes visible. (2)–(3)
- *ApprochVisionKhepera*: GK approaches VK by recognizing its light. (3)–(4)
- *ApprochBlue*: VK approaches the blue goal and rotates there until it finds the GK. (4)–(5)
- *GripObject*: GK approaches the red object and picks it up. (5)–(7)
- *ApproachGoal*: GK approaches the blue goal by recognizing the light of VK. (7)–(8)
- *ReleaseObject*: GK puts down the red object in front of VK. (8)–(9)

The literals are carefully designed by a human designer. Table 1 shows the name and interpretation of the literals. We consider the literals are sufficient, even not necessary. Also actions which the robots can execute are defined. Actions go_left, go_straight, go_right, turn_left, wait, turn_right, back} are defined for VK. For GK, actions {grip_object, release_object are defined in addition to the VK's actions.

To perform the task, we give our system a problem: an initial state {VKProximityLeft, GKLightFront} and a goal state {VKProximityBack, VKRedAround, GKLightFront, GKProximityFront}. This problem means that robots move an red object to a blue goal region.

6.4. Experimental results

In order to achieve this task, we directly manipulated VK and GK in a learning mode for generating the operators mentioned above, and a system acquired the operators successfully. Next a system generated a plan to solve the problem using the learned operators and Graphplan in a planning mode. Finally the two mobile robots executed the plan in a real environment, and the goal state was satisfied.

Fig. 9 shows a generated plan. In this figure, the descriptions of learned operators are indicated. Fig. 10 shows the trajectories of VK and GK in a real environment. We verified that they worked based on the plan and successfully achieved the goal state.

Literal name	Interpretation
VKProximityFront	An object in the front of VK
GKProximityFront	An object in the front of GK
VKProximityBack	An object at the back of VK
GKProximityBack	An object at the back of GK
VKProximityLeft	An object on the left of VK
GKProximityLeft	An object on the left of GK
VKProximityRight	An object on the right of VK
GKProximityRight	An object on the right of GK
VKLightFront	A light in the front of VK
GKLightFront	A light in the front of GK
VKLightBack	A light at the back of VK
GKLightBack	A light at the back of GK
VKLightLeft	A light on the left of VK
GKLightLeft	A light on the left of GK
VKLightRight	A light on the right of VK
GKLightRight	A light on the right of GK
VKRedAround	A red object in the front of VK
GKGrippedObject	GK grasps an object.
VKGreenAround	A green object in the front of VK
VKBlueAround	A blue object in the front of VK

Table 1 Literals used in the experiments.

7. Discussions

7.1. Design and check of literals

In a current system, we need to carefully design sufficient literals for achieving a given task as mentioned in 6.3. This design of literals becomes very hard for a human designer as a task gets more complex. Though an full-automatic method for designing literals is difficult, we need to develop a semi-automatic approach in which a system and a designer defines sufficient literals interactively. We consider unsupervised learning is applicable to generate candidates of literals.

7.2. Segmentation of operators

The segmentation of operators from a sequence of actions is done by a human teacher. He/she explicitly divides the sequence into operators by pointing out a teaching teaching start and a teaching finish state. However a system should segment a whole action sequence consisting of necessary operators to achieve a task into adequate sub-sequences. When tasks to be performed are given, the segmentation is realized so that the obtained operators can be necessary and sufficient to achieve all the tasks. Since this segmentation needs combinational search, we have a plan to employ an evolutionary computation approach.

8. Conclusion

We proposed a practical framework in which the generation of operators, planning and the execution of a plan were integrated for a heterogeneous multi-robot system. In our framework, first we designed literals to describe a environment, and directly manipulated real mobile robots to achieve

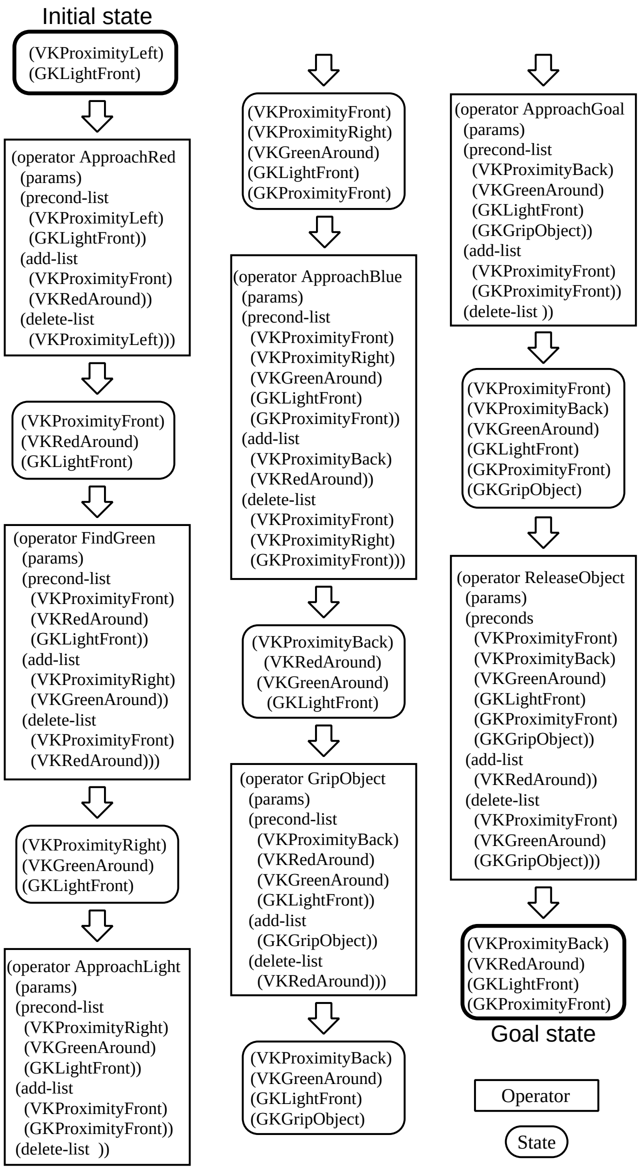


Figure 9 Generated plan.

a task in an environment. As a result, the initial state, the finish state and a sequence of the pairs of sensed data and an executed action were obtained. Using the two states, a system can generate an operator by constructing a precondition-list, an add-list and a delete-list. Once the operators were acquired, a system can apply a complete and sound planner to generate a plan. The generated plan was executed by instance-based operator execution with a 1-Nearest Neighbor method using the pair of sensed data and an executed action. Thus operator generation, planning and the execution of a plan were integrated in our framework. The feasibility of our framework was verified through experiments in which heterogeneous mobile robots cooperatively achieved a task.

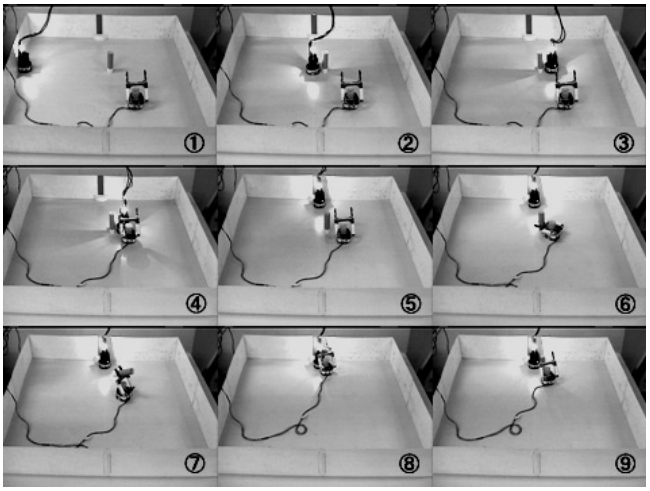


Figure 10 Execution of a plan in a real environment.

References

[1] A. Blum and M. Furst. Fast planning through planning graph analysis. *Artificial Intelligence*, 90:281–300, 1997.

[2] B. V. Dasarathy. *Nearest Neighbor (NN) Norms: NN Pattern Classification Techniques*. IEEE Computer Society Press, 1991.

[3] R. E. Fikes and N. J. Nilsson. STRIPS: A new approach to the application of theorem proving to problem solving. *Artificial Intelligence*, 2:189–208, 1971.

[4] G. W. Gates. The reduced nearest neighbor rule. *IEEE Transactions on Information Theory*, IT-18(3):431–433, 1972.

[5] P. E. Hart. The condensed nearest neighbor rule. *IEEE Transactions on Information Theory*, IT-14(3):515–516, 1968.

[6] S. Russell and P. Norvig. *Artificial Intelligence –A Modern Approach–*. Prentice-Hall, 1995.

[7] M. D. Schmill, M. T. Rosenstein, P. R. Cohen, and P. Utgoff. Learning what is relevant to the effects of actions for a mobile robot. In *Proceedings of the Second International Conference on Autonomous Agent*, pages 247–253, 1998.

[8] X. Wang. Learning by observation and practice: An incremental approach for planning operator acquisition. In *Proceedings of the Twelfth International Conference on Machine Learning*, pages 549–557, 1995.