Available online at www.sciencedirect.com

ScienceDirect

Procedia Computer Science 56 (2015) 538 – 543

Procedia Computer Science

International Workshop on Communication for Humans, Agents, Robots, Machines and Sensors (HARMS 2015)

Self-Management Technique for Adaptive Robot Software based on Task Environment Similarity

Yunsik Son[a] and Jin-Woo Jung[a,] *

[a]*Department of Computer Science and Engineering, Dongguk University, 3-26 Pil-dong, Jung-gu, Seoul 100-715, Republic of Korea*

Abstract

In this paper, we propose a novel method by which the robot can choose the most suitable software module for the given task based on the evaluation of task environment, and the resulting relationship between the environmental information and robot software module is managed by the robot itself. In addition, the performance of the robot can be improved through the update process of the E-S(Environment-S/W module) relationship information when the new environmental information or new robot software modules are given. The effectiveness of the proposed self-management technique is shown by the experiments with 70 random maps, which shows the improvement of performance as the task continues to proceed.

Peer-review under responsibility of the Conference Program Chairs

Keywords: Self-Update of Software Modules; Environment-S/W Module Relationship Information; Similarity of Task Environments; Adaptive Robot Software Framework.

1. Introduction

Dynamically changing behaviors, situations, and environments are difficult to be completely covered specially in the complex software for multi-robot cooperation. Therefore, the methodology to cope with dynamically changing elements for the given task has been studied in many different ways[1,2,3] and some of them focus on the modularity and the adaptation of robot software to the changing status of robot by replacing robot S/W modules.

If robots are able to adapt various changes in software aspect, we can reduce the complexity of software design for the robot that can perform in an open environment with a variety of purposes. And the success ratio of the given

* Corresponding author. Tel.: +82-2-2260-3812; fax: +86-2-2265-8742.
E-mail address: jwjung@dongguk.edu

Peer-review under responsibility of the Conference Program Chairs
doi:10.1016/j.procs.2015.07.248

task will be also increased by the use of adaptation concept in the robot software. Especially, if the robot is able to improve its own software component by self-evaluation, more efficient results could be expected for the given task. However, most of previous studies focus on dynamically reconfigurable framework with pre-known relationship between robot S/W modules[4,5,6]. And it is very hard to find any research on the dynamic reconfiguration of robot S/W modules with unknown relationship. Therefore, newly developed modules are hard to be applied to the robot, which is already under working, since there is no previously known information on this module.

In this paper, we propose a novel method that can overcome these drawbacks of previous researches by using self-evaluation on insufficient information.

2. Related Studies

2.1. R-Object Model

R-Object(Robot Object)[2] is a conceptual robot component model, by which a robot can be abstracted with the union of S/W and H/W function set. By dynamic reconfiguration of function set in R-Object, heterogeneous multi-robot cooperation can be easily realized. It makes an advantage that only one R-Object model can replace a set of R-Object models related with several robots. Every R-Object model has three basic functions as follows: inference ability for extracting the required union of functions for the given task, self–deduction ability of its own possible combination of basic functions, and communication ability with other R-Objects to use additional functions belonging to other robots. By using these three basic functions, a robot with R-Object can also diagnose whether a newly given task is executable or not. And, R-Object model could be implemented using the TBPPC structure (Tasks - required Behaviors - Platform Independent Function elements - Platform Specific Function elements - executable Components) with the attributes and action algorithms of the robot and the pre-known mapping information between TBPPC elements. Using the R-Object model, a hardware robot could be abstractly portrayed. As a result, the robot can be described with consistency from functional perspective independent from its own hardware element. R-Object can offer a TBPPC structure-based hardware-independent interface so that dynamic aggregation and separation of software modules is possible and it can be enlarged into the dynamic reconfiguration of heterogeneous robots. This allows us to apply a unified model for robot hardware and software.

2.2. SHAGE

SHAGE(Self-Healing, Adaptive, and Growing SoftwarE)[3] is a framework that S/W module can be dynamically reconfigured to reduce the complexity of system. The robot based on SHAGE is able to adapt for various tasks or environmental changes. It is similar with R-Object model in terms of re-configurability of robot S/W function modules. But, the framework requires detailed and complete knowledge on the module which will be changed.

3. Proposed Self-Management Method

Methodologies, which mentioned in chapter 2, are based on the technologies to enable the task to continue even in the dynamically changing environment by the modularization of robot software. In this paper, we propose a novel method based on aforementioned methodologies. Proposed method can change the current software module, which is already known as suitable for the current task environment, with the better module for the same task still while running for the current task. The method is comprised of two algorithms. The first one is a selection algorithm to find a suitable robot software module in the given environment based on E-S (Environment-S/W Module) relationship information. The other is an algorithm for updating E-S relationship information when new software modules are added.

3.1. E-S (Environment–Software Module) Relationship Information

A software module of robot, in this paper, is a unit module for performing a specific task. In case of multiple modules or multiple combined modules having the same functionality for the given task, each performance of the

module or combined modules is generally determined by its implementation or combination algorithm. However, in the robot system, the environment also affects each performance of the module or combined modules for the given task. Therefore, to find a suitable module or suitable combined modules in the dynamically changing environment, not only the module information but also the additional information on relationship between environment and modules is required. E-S relationship information describes the relationship between previously experienced environment set and available module set to perform the same task, such as different maps and its suitable path planning algorithms. In E-S relationship information, each environment is connected with a robot software module whose performance is the highest recorded. In this paper, we define the E-S relationship information as follows: <E: Environment, M: Software Module, P: Task Performance>.

3.2. Software Module Selection Algorithm

To adapt various environments, software module selection algorithm decides a suitable robot software module on the given environmental information like Fig. 1.

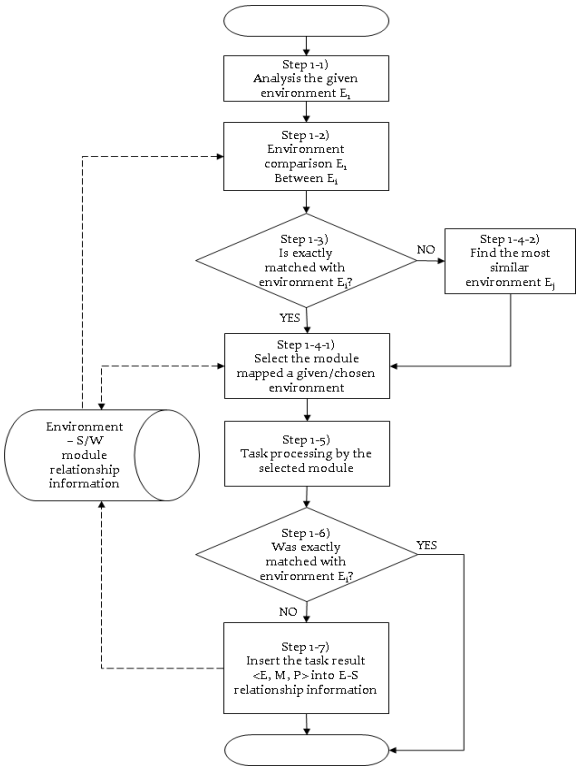

Fig. 1. Diagram for software module selection algorithm.

First, the robot analyzes the given environment and compare with the existing environments in E-S relationship information. If the given environment is exactly matched with the existing environment, a software module related with the existing environment is selected through E-S relationship information. Otherwise, the environment, which is included in the E-S relationship information and is the most similar with the given environment, is found and its related software module is also selected. At this time, the similarity between the newly given environment and the existing environment should be considered. Next, the given task is tried to be done with the selected software module. In case of new environment, additional E-S relationship information are measured based on some performance measure and is updated into the form of E-S relationship information as follows: <E: current environment, M: selected module, P: measured performance>.

3.3. E-S Relationship Information Update Algorithm

E-S relationship information update algorithm reflects the changed status on all the existing E-S relationship information when a new or updated software module is given from other robots (in the case of distributed multiple robot cooperation) or the main control system (in the case of centrally controlled robot) even when the robot is still running for the given task. The algorithm is shown in Fig. 2. In case of Step 2-3), E-S relationship information is updated as follows: <E: target environment, M: newly added module, P: measured performance>.

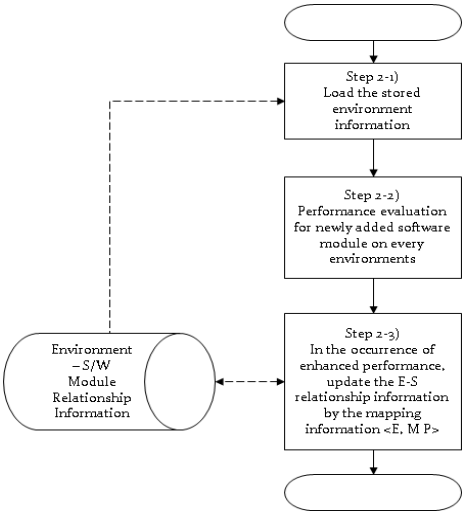

Fig. 2. Diagram for E-S relationship information update algorithm.

4. Case Study

As a case study, the environmental information is defined as 2-D binary maps and the 9 path planning algorithms are used as robot software modules. Detailed assumptions and the experimental scenario are addressed in Sect. 4.1.

4.1. Experimental Environments

We assume an experimental environment such as Table 1 to verify the proposed method. Table 2 shows the experimental scenario in this case study.

Table 1. Assumptions for the case study experiment.

1.		The target platform for the experiment is a mobile robot.
2.		Use 2-D binary map for the environment (0:free space, 1:obstacle)
3.		Random map is generated under the following conditions
	A.	Map size: 30 cell * 30 cell
	B.	Cell size: 30cm * 30cm (it covered actual robot platform size)
	C.	Start posture: randomly determined within 3 cells on the left side of the map
	D.	Goal posture: randomly determined within 3 cells on the right side of the map
	E.	Obstacles: 3~10 obstacles for each maps
4.		Assigned task to the robot is finding a collision-free path and move the robot from the start posture to the goal posture.
5.		Robot has the initial E-S relationship information as

	<E: Empty map environment, M: Algorithm ①, P: +infinity>.
6.	No. of randomly generated map is 70.
7.	Software modules for experiments are 6 path planning algorithms.

Table 2. Scenario for the case study experiment.

1.	Experimental space for robot platform is $81m^2$ (30cell * 30cell * 30cm * 30cm), and the maps that include logical obstacles are given for path planning.
2.	The robot received new random generated map and performs the task using the module selection algorithm.
3.	For every 10th task, new software module is given as ascending order.
4.	Given the new software module, the robot applies the E-S relationship information update algorithm first and then performs the given task on the next map.
5.	Performance is measured with the travelled path length by odometry estimation and shorter path is better one.

Our experiment uses 6 path planning algorithms as robot software modules. The software modules consist of 2 different path planning algorithms (cell decomposition and advanced cell decomposition[7]) and 3 different map analysis methods (DFS, BFS, A*). By each combination, totally 6 different algorithms are made as followings:
① Cell Decomposition with DFS, ② Cell Decomposition with BFS, ③ Cell Decomposition with A*, ④ Advanced Cell Decomposition with DFS, ⑤ Advanced Cell Decomposition with BFS, ⑥ Advanced Cell Decomposition with A*

4.2. Similarity Estimation for Different Environments

In this case study, we assume the environmental information as a 2D binary map. And, the similarity between two maps is evaluated based on the size of shared area after proper alignment process to match the start and goal posture of two different maps.

This estimation process is composed of two phases. The first phase is a preprocessing phase for comparing two maps, and the second phase is a similarity estimation phase on the common area with the preprocessed map. In the preprocessing phase, the newly given 2D binary map image IB is transformed to match with the existing map image I_A, where I_A, I_B: $(x, y) \in \mathbb{Z}^2 \rightarrow [0,1]$, as follows:

$I_{B'} = (S_{ToEnd} \circ R_{Start \rightarrow End} \circ T_{ToStart}) I_B$

i) $T_{ToStart}$ is a translation matrix to match with the start posture of I_A.
ii) $R_{Start \rightarrow End}$ is a rotational matrix to match with the direction from the start to the end of I_A.
iii) S_{ToEnd} is a scaling factor matrix to match with the end posture of I_A by scaling after translation and rotation.

In the similarity estimation phase, the similarity between different environmental maps is defined as follows:

$$Similarity = 1 - \sum_{(x,y) \in D} \frac{|I_A(x,y) - I_{B'}(x,y)|}{n(D)} \quad (1)$$

where D is the common area of I_A and transformed $I_{B'}$, and the n(D) is the number of elements in the set D.

At this time, the most similar map among the existing maps is decided based on the highest similarity value with Eq. (1) in the common area.

4.3. Experimental Results

Fig. 5 shows the performance ratio of each used algorithm, which has been changed based on the transferred and selected software modules in the non-uniform time scale. In Fig. 5, the path length ratio is the rate between the results of robot and the estimated worst case results as maps. And, the optimal and worst algorithm results were calculated as follows. For each map, all 6 algorithms were used to be evaluated and the best performed algorithm's

result was selected for each target map. In contrast, the results of worst case were selected by the lowest performance in the results of 6 algorithms for each map.

As shown in Fig. 5, the path length ratio is gradually reduced and closer to the optimal result as time goes on while E-S relationship information is continually updated. Accordingly, the robot can adapt new software modules and variant task environments flexibly.

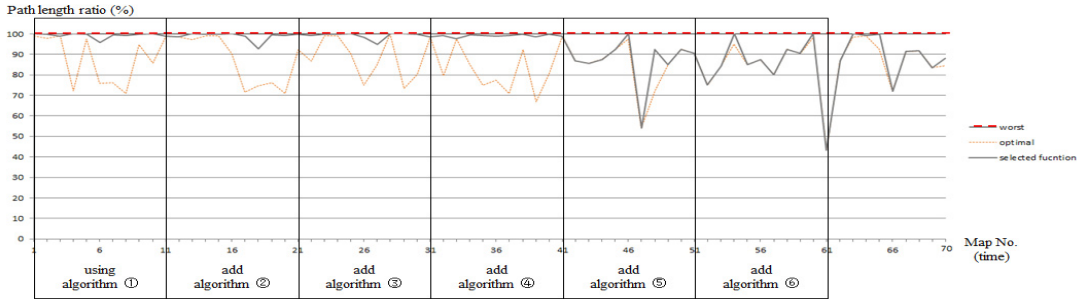

Fig. 5. Experimental results as the transition of maps.

5. Conclusions

The researches on the modularity of robot software and platform for dynamically changing environments are much in progress. But, it is hard to find the research on the methodology to deal with reconfigurable robot software modules with unknown relationships. In this paper, we proposed a method that robot can decide a suitable software module by itself under the given situation to improve the task performance. Through the proposed method, robot can evaluate and select, by itself, what is the most proper module for the dynamically changing environment according to the given task. And newly added or updated modules could be also applied efficiently to the robot as the updated E-S relationship information, even when the robot is already under working. The proposed method could be more effective in some application areas where the long-lasting cooperation of multi-robots is highly required.

Acknowledgements

This work was partially supported by the Basic Science Research Program (2010-0025247) through the National Research Foundation of Korea funded by the Korean government (MEST). And this work was started from the IT R&D program of MKE/KEIT (2008-F-041-01).

References

1. Lundh R, Karlsson L, Saffiotti A. Automatic Configuration of Multi-Robot Systems: Planning for Multiple Steps. *Proceeding of the 2008 conference on ECAI 2008*; 2008. pp.616-620.
2. Park JW, Son Y, Jung JW, Oh SM. R-Object Model for Evolutionary Robots using Multi-robot Cooperation. *The 2nd IFAC International Conference on Intelligent Control Systems and Signal Processing*; 2009. pp.438-443.
3. Kim D, Park S, Jin Y, Chang H, Park YS, Ko IY, Lee K, Lee J, Park YC, Lee S. SHAGE: a framework for self-managed robot software. *Proceedings of the 2006 international workshop on Self-adaptation and self-managing systems;* 2006. pp.79-85.
4. Kim D, Park S. Alchemistj: A Framework for Self-Adaptive Software. *The 2005 IFIP International Conference on Embedded And Ubiquitous Computing*; 2005. pp. 98-109.
5. Koo HM, Ko IY. A Repository Framework for Self-Growing Robot Software. *Proceedings of 12th Asia-Pacific Software Engineering Conference*; 2005. pp.515-524.
6. Lee H, Choi HJ, Ko IY. A semantically-based software component selection mechanism for intelligent service robots . *4th Mexican International Conference on Artificial Intelligence*; 2005. pp.1042-1051.
7. Jung JW, So BC. An Idea to Reduce Number of Cells in the 2D Exact Cell Decomposition-based Mobile Robot Path Planning. *The 7th International Conference on Ubiquitous Robots and Ambient Intelligence*; 2010. pp.417-420
8. Park JW, Son Y, Jung JW, Oh SM. Software Interface for Hardware-independent Robot Platforms. *International Journal of Assistive Robotics and Mechatronics* 2008;9:110-119.