Robotics and Autonomous Systems 61 (2013) 1360–1378

Contents lists available at ScienceDirect

Robotics and Autonomous Systems

journal homepage: www.elsevier.com/locate/robot

A fault-tolerant approach to robot teams

Adrian Martin, M. Reza Emami *

University of Toronto, Institute for Aerospace Studies, 4925 Dufferin Street, Toronto, Ontario, Canada M3H 5T6

HIGHLIGHTS

- Adaptability and reusability are underexplored areas in robot control system design.
- We present the Control ad libitum philosophy for control system design.
- We developed the HAA (Host, Avatar, Agent) architecture for distributed robot teams.
- Experiments with a full HAA implementation for exploration, mapping, and foraging are presented.
- Performance under various failure scenarios is studied.

ARTICLE INFO

Article history:
Received 17 October 2012
Received in revised form
10 June 2013
Accepted 27 July 2013
Available online 2 August 2013

Keywords:
Robot teams
Control system architecture
Distributed systems
Fault tolerance
Hardware-in-the-loop simulation

ABSTRACT

As the applications of mobile robotics evolve it has become increasingly less practical for researchers to design custom hardware and control systems for each problem. This paper presents a new approach to control system design in order to look beyond end-of-lifecycle performance, and consider control system structure, flexibility, and extensibility. Towards these ends the Control *ad libitum* philosophy was proposed, stating that to make significant progress in the real-world application of mobile robot teams the control system must be structured such that teams can be formed in real-time from diverse components. The Control *ad libitum* philosophy was applied to the design of the HAA (Host, Avatar, Agent) architecture: a modular hierarchical framework built with provably correct distributed algorithms. A control system for mapping, exploration, and foraging was developed using the HAA architecture and evaluated in three experiments. First, the basic functionality of the HAA architecture was studied, specifically the ability to: (a) dynamically form the control system, (b) dynamically form the robot team, (c) dynamically form the processing network, and (d) handle heterogeneous teams and allocate robots between tasks based on their capabilities. Secondly, the control system was tested with different rates of software failure and was able to successfully complete its tasks even when each module was set to fail every 0.5–1.5 min. Thirdly, the control system was subjected to concurrent software and hardware failures, and was still able to complete a foraging task in a 216 m^2 environment.

1. Introduction

The field of mobile robotics has seen dramatic growth and technologies have reached the point where many real-world applications of robot teams are being realized, which makes it all the more important to step back and consider how these teams are being designed and built. Often a particular robot or control system is designed with a specific task in mind, and not much consideration is given to how that task is achieved as long as at the end of the day it works. Although this approach has yielded many successes it seems fundamentally limited in terms of its robustness and potential for far reaching application in the real world. The aptly-titled article *1,001 Robot Architectures for 1,001 Robots* [1] highlights this issue and asks the question "Is it really impossible to subject robot architectures and software systems to any objective performance evaluation?" The review of benchmarking and standardization conducted in [2] lists many initiatives, but by and large their focus appears to be on after-the-fact performance analysis rather than strategies to assist developers design better teams. Two exceptions are the Robotics Domain Task Force of the Object Management Group [3] who encourage designs using modular components, and the Joint Architecture for Unmanned systems (JAUS) which follows the five principles of vehicle platform independence, mission isolation, computer hardware independence, technology independence and operator use independence [4].

In an attempt to approach the issue of robot team design from a broader "big picture" perspective, the Control *ad libitum* philosophy is introduced in [5], and several tenets are proposed

* Corresponding author. Tel.: +1 416 946 3357; fax: +1 416 946 7109.
E-mail addresses: martin@utias.utoronto.ca (A. Martin), emami@utias.utoronto.ca (M.R. Emami).

0921-8890/$ – see front matter © 2013 Elsevier B.V. All rights reserved.
http://dx.doi.org/10.1016/j.robot.2013.07.015

that can help lead to the design of more adaptable, more efficient, and more robust teams. Following these tenets, the HAA (Host, Avatar, Agent) architecture was developed to provide a generic and dynamic foundation for distributed control systems. Using the HAA architecture virtually any type of control strategy can be implemented, including fully centralized or fully distributed components, while leveraging the inherent robustness of the load balancing and failure recovery features. After implementing the HAA architecture using provably correct distributed algorithms, a control system for mapping and foraging was developed. The control system was used in extensive Hardware-in-the-Loop simulation (HILS) experiments to verify the functionality of the architecture and explore the effects of failure.

Section 2 provides a background on the relevant aspects of robot teams, with particular focus on the concepts of distributed processing and behaviour migration. Section 3 introduces the underlying concept of Control *ad libitum* and the format of the HAA architecture. The experimental setup is described in Section 4, which touches on the details of the HAA implementation but focuses on the experimental scenarios and specifications of the experiments presented in this paper. A series of three experiments are discussed in Section 5. The first experiment analyses the basic functionality of the HAA implementation, the second studies the impact of various rates of agent failure, and the third tackles the issue of concurrent hardware and software failure. The conclusion in Section 6 summarizes the goals of the HAA architecture and its overall success as a fully realized implementation, but points out that further refinements can greatly improve the simplicity and usability of the architecture.

2. Background

Developing and implementing a team of cooperative mobile robots is a very challenging task, yet the reward is an efficient and effective solution for many application problems. These applications span a wide range of practical, real world, scenarios, and include: working in hazardous environments, surveillance, and mine field demolition [6]. Sometimes tasks can be carried out by a single robot with powerful sensors and high processing capability; however, often these tasks can be carried out faster, more efficiently, and more robustly by using a team of simpler and cheaper robots [7]. Huge amounts of research has been done on a multitude of aspects of team development, but many fundamental questions are far from answered. This section is concerned with reviewing the key elements which must be considered when designing a cooperative team, focusing on the areas of distributed processing and behaviour migration.

2.1. Teams vs. single robots

Some of the traditionally envisioned robotic tasks include environment mapping, surveillance, and search and rescue. Often these tasks demand a robot with powerful sensors and high processing capacity [7], which usually corresponds to a high cost. However, in some applications it is possible to use a team of simpler robots to accomplish the tasks faster and more efficiently [7]. Each individual may be significantly less capable than a single more expensive robot, but when properly controlled it is possible for them to work together and perform complex tasks. Depending on the structure of the team there may not be any inherent cost benefit; however, there are a number of other benefits that come with multi-robot systems: efficiency, cost per system, robustness through redundancy, parallel processing, and scalability [8,9]. Additionally, larger areas can be serviced and multiple tasks can be accomplished simultaneously by spreading out the team [10]. Furthermore, there is a potential for self-diagnosis and self-repair of failures in robot teams [11].

2.2. Homogeneous vs. heterogeneous teams

When building a team of robots there are a number of possibilities regarding the homogeneity of its members. Obviously, it is possible to select a number of different types of robots to form the team, which results in a physically heterogeneous population; but even in the case of physically identical robots it is possible to introduce heterogeneity in their controllers, either by design or through learning. Another factor to be considered is the degree of heterogeneity of the team, which may be a useful metric when comparing the relative performance of two systems. The concept of social entropy is introduced in [12], providing a way to quantitatively rate the diversity of a team based on factors relevant to the application.

A heterogeneous team of robots that explore and map their surroundings and then set up to monitor the presence of intruders is presented in [13]. Their team consisted of a few highly capable expensive robots equipped with powerful sensors and processors and a large number of simpler robots with weak sensors and processors. In this way the cost of the team was reduced by almost an order of magnitude while still providing coverage for large areas. They were able to take advantage of heterogeneity to find a balance within the cost/capability/number of robots trade space.

An example of a physically homogeneous team with heterogeneous controllers is discussed in [14]. Inspired by specialization in insect colonies, as a swarm of robots learns how to perform basic tasks related to finding and collecting objects they begin to develop proficiencies in different areas and the tasks are allocated throughout the team based on fitness.

Looking from the team design perspective, in the examples in [14,13] it can be seen that by appropriately selecting the type and degree of diversity it is possible to improve the efficiency of the team and find a balance between the cost, capability, and number of robots. Furthermore, as in nature, heterogeneity and diversity in a population can provide the much needed robustness. In experiments using robots emulating wolf-pack hunting strategies, [15] showed that heterogeneous teams composed of both peak and senescent "wolves" could outperform a team of purely peak wolves in certain scenarios. There is also the consideration of scalability and utilizing all available resources. When designing a team from the ground up, it is possible to specify the exact configuration of each of the members, but if at a later time the team is expanded or assigned new tasks it might not be feasible to supply exactly the same kind of robot. A system that is capable of handling heterogeneity can potentially make use of whatever robots are on hand which can reduce the cost of the changes [16].

2.3. Dynamically formed teams

One interesting area of research that has received little study is the concept of dynamic or "pickup" teams [16]. Rather than planning the team membership and methods of interaction in advance, it is more useful to have a system that can dynamically adapt based on the available resources and the environment. This area covers a number of issues, such as how to handle the formation of a new team, what to do when a robot fails or comes back online, or the formation of sub-teams from a larger group of robots to handle particular tasks. The work in [16] outlines some reasons why this capacity is needed: (1) it is impractical for a single group to develop large teams of expensive robots simultaneously, (2) engineering coordination strategies by hand is time consuming and may not be acceptable in emergency situations, and (3) when replacing robots hardware of the same type may not be available. A strategy for accomplishing this goal through communication between potential team members is outlined in [16], and a treasure hunting application using two types of robots is presented. There is significant potential for flexibility and cost reduction when