On the Use of Teamwork Software for Multi-Robot Formation Control

(Extended Abstract)

Gal A. Kaminka
The MAVERICK Group
Bar Ilan University, Israel
galk@cs.biu.ac.il

Meytal Traub
The MAVERICK Group
Bar Ilan University, Israel
traubm@cs.biu.ac.il

Yehuda Elmaliach
Cogniteam, Ltd. &
College of Management, Israel
yehuda@cogniteam.com

Dan Erusalimchik
Cogniteam, Ltd., Israel
dan@cogniteam.com

Alex Fridman
The MAVERICK Group
Bar Ilan University, Israel

Categories and Subject Descriptors

D.2.11 [**Software Architectures**]; I.2.9 [**Robotics**]: Autonomous Vehicles; I.2.11 [**Distributed Artificial Intelligence**]: Coherence and Coordination

General Terms

Experimentation,Reliability

Keywords

Multi-robot formations;Multi-robot Systems;Teamwork

1. INTRODUCTION

In recent years there has been a growing interest in multi-robots systems, where a group of N robots are working collaboratively in order to execute a given task. One specific example is *multi-robot formation maintenance*, where the goal is for a group of robots to move while maintaining relative positions with respect to each other (typically describing a specific geometric shape). Indeed, there exists vast literature on various techniques for maintaining formations in a variety of settings. Different controllers have advantages and disadvantages which can be complementary.

Little attention has been given to the possibility of integrating together multiple, complementary, formation controllers for greater robustness. For example, it is possible to have the robot team switch between controllers. Separation-bearing controllers (SBC) rely heavily on the robots sensors [4, 2, 7]. Communications-based controllers rely instead on reliable communications from one robot to the others [3]. If all members of the team dynamically switch between these controllers, together, it would allow the team to compensate for sensor faults using communications, and to compensate for communication faults, by relying on sensors.

Such integration is a formidable challenge. The execution of a multi-robot formation controller is often distributed, with each robot individually executing a program associated with its role. For an integrated formation system to be effective, it must have all the robots in the team switch together from one type of controller to another. Tight coordination is required not just in deciding on the time of the switch, but also in its contents—all robots must switch to the same controller. The complexity of the communication protocols and the decision-making procedures themselves can be significant.

In this abstract, we briefly discuss the use of a teamwork software architecture to automate joint selection of a multi-robot controller, and its execution. Teamwork software (such as BITE [6] or CogniTAO [1]) automates sharing of information and the use of social choice protocols. In particular, it allows synchronous joint selection of controllers for execution. Building on this automation, we describe a formation maintenance system, which integrates together several different formation-maintenance controllers for greater robustness. The robots jointly switch between different controllers, so as to address intermittent failures in sensing or communications. The use of teamwork software in robotics has been reported before [6], but its use for integration of different control schemes is novel.

2. RELATED WORK

In formation-maintenance tasks, the objective is to move a group of robots on a desired path, while they maintain their relative position with respect to their peers, according to a desired geometric shape. Many formation maintenance methods have been investigated. (e.g., [2, 8, 5, 7, 3]). These vary in their operating assumptions, their reliance on specific sensors, and their ability to handle different types of faults.

Some researchers have examined integration of multiple methods. Fierro et al. [4] analyzed the stability of different controllers, and proposed using manually-constructed control targets to allow up to three robots to switch between alternative schemes, without relying on communications. Fredslund and Matarić [5] describe a method that supplements SBC control by communications, for robustness, thus fusing SBC and communications-based control. Elmaliach and Kaminka [3] build on this to experiment with different ways of integrating communications and SBC controllers (switching between them and/or fusing their actions). However, neither investigation discusses integration of multiple controllers in general, and both ignore the requirement for a principled way for the robots to jointly select their control scheme.

In this paper, we describe how to use teamwork software to automate the joint online selection of controllers by the robots. Teamwork architectures facilitate the development of distributed multi-agent systems that collaborate towards a joint goal, via an agreed-upon plan of execution. *BITE* (Bar Ilan Teamwork Engine) is a behavior-based distributed teamwork control software, specifically targeting multi-robot teams [6]. While BITE has been used with

Appears in: *Proceedings of the 12th International Conference on Autonomous Agents and Multiagent Systems (AAMAS 2013)*, Ito, Jonker, Gini, and Shehory (eds.), May, 6–10, 2013, Saint Paul, Minnesota, USA.

multi-robot formations [6], it has only been previously used with a single type of controller (SBC). CogniTAO [1], used here, is a commercial teamwork software development kit, which is used to develop multi-robot applications.

3. TEAMWORK SOFTWARE

We utilize a straightforward behavior selection mechanism, in which behaviors (simple controllers, capable of carrying out specific subtasks) are proposed and selected based on their preconditions matching the current world state (as perceived by the sensors); once a behavior is selected, its execution is terminated when its termination conditions are satisfied. There are two components. A *control process* that carries out the selection, execution, and termination of behavior (as described above), and a *world modeling* process that processes sensor information, so that the preconditions and termination conditions of behaviors can be matched against the current state of the world as perceived by the robot.

Behavior-based teamwork software, executing in a distributed fashion on multiple robots, extends the control and world modeling processes. We describe the generic principles of these extensions, rather than focus on a specific implementation.

Each robot executes its own processes; these communicate with those of other robots as needed. In the control process, in addition to the conditions, we now also associate a *team flag* with each behavior. When set, the flag informs the architecture that the developer requires selection of this behavior to be synchronized with the other robots, i.e., that if the behavior is executed on the robot, its copies on the other robots must be executed as well. Behaviors thus flagged are called *team behaviors*. The control process is extended to facilitate *synchronized behavior selection*, in a distributed manner. The most general form of this extension works as follows. Whenever the extended control process of a robot reaches a decision point (e.g., to select a new behavior for execution, or to terminate an existing behavior), it first checks the team flag. If set, the process then takes actions (typically, by communications) to select/de-select the same behaviors in the other robots. These actions typically take the form of a robust interaction protocol. Finally, the world modeling process is extended to communicate (a subset of) its sensor processing results with other robots, and appropriately receive their own perceptions. Such communications can be carried out using a variety of *information fusion protocols*.

The result of these extension is that perception—information—is shared between robots, and so the basis for their decision making is some (partial) shared perception of the world. Team behaviors get selected and de-selected jointly by interaction protocols, based on this shared perception. Behaviors not marked as team behaviors get selected/de-selected independently from the choices of other robots. Some architectures distinguish between *choice protocols*, in which team-members come to an agreement (e.g., by applying voting rules) as to the team-behavior to be executed or terminated, and *allocation protocols*, in which team-members decide together on assignment of non-team behaviors that should be run at the same time, in service of an agreed upon team-behavior.

4. MULTIPLE FORMATION CONTROLLERS

The use of teamwork architecture allows the developer to easily integrate different controllers, for greater robustness. We chose to integrate two sophisticated robust formation-maintenance controllers. The first, switching separation-bearing control (SBC) is originally described in [7]. The second, a communication-based formation controller is originally described in [3]. Each of these was proposed to address a different set of potential failures. Their combination, made possible by the use of the teamwork architecture, provides robustness against a wide variety of failures.

Figure 1: Three shrimps robots in an outdoor environment.

The switching SBC controller [7] was designed in order to solve longer-duration sensing failures, e.g., caused by permanent loss of the identification pattern, or relatively long visual loss when a target lead took a sharp turn. In these cases the robots look for a new SBC control graph in order to keep the formation structure.

On the other hand, the communication-based controller [3] was designed to solve problems of short-duration intermittent failures to visually recognize a target robot, by allowing a follower robot to follow its target blindly. The target transmits information as to its movements; the follower translates these into its own target coordinates, and moves accordingly.

By integrating these controllers, we make the system robust to all the problem described above. If a robot loses its vision abilities temporarily, the use of the communication-based controller works to maintain the formation for a short duration using communications from the lead robot. If and when this fails, the switching SBC controller will be used in order to maintain the formation.

We fully implemented the integrated controller described above, using CogniTAO [1]. The system was deployed with three BlueBotics Shrimps III robots. The robots utilized IEEE 802.11g communications, Sony PTZ cameras Hokuyo lidars. Each robot was controlled by a VIA C7 CPU, utilizing 512MB ram (Figure 1).

Dozens of runs were executed in both environments. These demonstrate the efficacy of the approach. In addition, we have shown that from a software engineering perspective, the use of the teamwork architecture leads to very significant savings in software development efforts.

5. REFERENCES

[1] CogniTeam, Ltd. CogniTAO (Think As One).

[2] J. P. Desai. A graph theoretic approach for modeling mobile robot team formations. *Journal of Robotic Systems*, 19(11):511–525, 2002.

[3] Y. Elmaliach and G. A. Kaminka. Robust multi-robot formations under human supervision and control. *JoPHA*, 2(1):31–52, 2008.

[4] R. Fierro, A. K. Das, V. Kumar, and J. P. Ostrowski. Hybrid control of formations of robots. In *ICRA-01*, 2001.

[5] J. Fredslund and M. J. Mataric. A general algorithm for robot formations using local sensing and minimal communications. *IEEE Transactions on Robotics and Automation*, 18(5):837–846, 2002.

[6] G. A. Kaminka and I. Frenkel. Integration of coordination mechanisms in the BITE multi-robot architecture. In *ICRA-07*, 2007.

[7] G. A. Kaminka, R. Schechter-Glick, and V. Sadov. Using sensor morphology for multi-robot formations. *IEEE Transactions on Robotics*, pages 271–282, 2008.

[8] F. Michaud, D. Létourneau, M. Gilbert, and J.-M. Valin. Dynamic robot formations using directional visual perception. In *IROS*, 2002.