Experimental Evaluation of Teamwork in Many-Robot Systems (Demonstration)

Andrea D'Agostini
Department of System and Computer Sciences
"Sapienza" University of Rome, Italy
andreadago@gmail.com

Daniele Calisi
Department of System and Computer Sciences
"Sapienza" University of Rome, Italy
calisi@dis.uniroma1.it

Alberto Leo
Space Software Italia s.r.l.
Rome, Italy
alberto.leo@ssi.it

Francesco Fedi
Space Software Italia s.r.l.
Rome, Italy
francesco.fedi@ssi.it

Luca Iocchi
Department of System and Computer Sciences
"Sapienza" University of Rome, Italy
iocchi@dis.uniroma1.it

Daniele Nardi
Department of System and Computer Sciences
"Sapienza" University of Rome, Italy
nardi@dis.uniroma1.it

General Terms

Experimentation

Keywords

Multi-robot system, experimental setting

1. INTRODUCTION

The experimental evaluation of methods and techniques for teamwork in multi robot systems (MRS) is challenging. Experiments with multiple robots are very difficult to manage [4] and thus the proposed approaches are seldom evaluated on real multi robot systems composed by several robots.

Teamwork in MRS, especially when aiming at massive experiments, is often evaluated using abstract simulators, which typically focus on the communication model, but make very rough assumptions on the behavior of the robots in the operational environment. In these cases, it may happen that the simulation model is too abstract to provide convincing evidence that the results obtained in simulation, apply also to the real case. Obviously, the more complex each individual robot is, the larger the distance between the simulation and the real case. Indeed, we have experienced that the performance of teamwork in MRS is deeply influenced by the performance of the robotic platform in the operational environment. Consequently, in order to bridge the existing gap with real robots, we have focussed on simulators that are originally designed for robotic systems and provide a more accurate model of the performance of the robots. This approach is challenging for a number of reasons. First of all, simulation tools are sometimes embedded in a software development framework, like for example Microsoft Robotics Developer Studio[1], or come as commercial products (e.g Webots [5]). Moreover, even when the simulator is accessible through a dedicated interface, the design and implementation of the system and of the simulation scenario can be rather resource intensive.

Player [2] is a very widespread tool; it includes in its package both a 2D (Stage) simulator and a 3D one (Gazebo): the Stage simulator is particularly suited for large-scale simulation of teams of several robots, as reported in [6]. In addition, these simulators provide models of distance sensors, thus allowing for an accurate modeling of navigation and localization in the environment, that make them suitable for experimental evaluation of several robotic tasks. Moreover, Player is providing an interface to robotic platforms and sensors that is becoming a de-facto standard. However, experiments of complex teamwork capabilities, that include several robots with complex individual functionalities and make use of a realistic robot simulator such as Stage have not been deeply investigated.

In this paper, we present an experimental set-up, based on our robotic software, that allows to make performance evaluation of systems including tenths of robots, simulated as complete applications, using Player/Stage. The key feature of our implementation is that each robot is simulated using the whole robotic software, by simply replacing the interface to the real robot with the Player interface. By switching interface we can run the real robot, thus allowing, for example, experiments simultaneously including real and simulated robots.

The expected benefits of our proposed setting are mainly in reducing the gap between the behavior of the simulation as compared with experiments with real robots. To this end, in addition to the usually implemented variants of the communication model, we run experiments which analyze the behavior of the system with respect to different robotic platforms, different sensor settings, different navigation algorithms, different localization algorithms, etc..

In the next section, we describe the implemented system and then we provide some examples of experimental evalu-

Cite as: Experimental Evaluation of Teamwork in Many-Robot Systems (Demonstration), A. D'Agostini, D. Calisi, A. Leo, F. Fedi, L. Iocchi, D. Nardi, *Proc. of 10th Int. Conf. on Autonomous Agents and Multiagent Systems (AAMAS 2011)*, Yolum, Tumer, Stone and Sonenberg (eds.), May, 2–6, 2011, Taipei, Taiwan, pp. 1327-1328.

[1] `www.microsoft.com/robotics`

ations.

2. SYSTEM DESCRIPTION

The software of each simulated robot runs inside a virtual machine: in this way, the deployment to real robots is straightforward. Details of the system functionalities and capabilities can be found in [1].

Different tasks are performed by the components included in each virtual machine: behaviors (e.g., exploration, take a picture, obstacle avoidance, etc.), sensor processing, etc. In particular, we focus on the coordination module, which has been designed to implement task assignment [3], with several degrees of flexibility. The coordination algorithm manages entities called tasks, that are distributed over the network together with other information in order to assign each task to one robot.

For task-assignment purposes, we use the two-phase approach described in the following. First, tasks are dynamically discovered by some robots (depending on sensor reading and situation assessment) or injected into the system by an external agent (e.g., a user GUI); the robots that receive the task use a utility function to decide whether to candidate themselves to execute the task or not. The candidature is the second phase of the algorithm: robots send their candidature (i.e., their expected utility) to the subgroup of robots that participate to the candidature of this task. The robot with the highest candidature is assigned the task.

In addition to the above outlined schema, a number of features have been added to ensure the generality of the approach:

- *duplicate task removal*: the system is able to detect similar tasks (e.g., the same task that has been discovered by two different robots) and drop all but one;
- *task persistence*: if no robot decides to candidate to the execution of a task, this is re-submitted;
- *task priority*: a priority is assigned to each task class, and each task instance can further refine this priority: while a robot is performing a task, it always candidates for other tasks with a higher priority, if it gets the assignment of the task, it interrupts the previous one and re-submits it into the system;
- *sub-teams formation*: in order to execute tasks that require more than one robot, the system is able to build sub-groups of team-mates, each of which with a specified role in the task execution;
- *open teams*: since the sub-teams of robots that are interested in a task are dynamically built during the mission, the robots do not need to know the exact number of their team-mates: this results in the possibility for robots to lately join or leave the team.

In the next section we describe a set of experiments that we performed in order to evaluate the behavior and the robustness of the system. In these experiments, we have been able to run up to 20 robots using 20 virtual machines distributed over a network of 4 multi-core hosts, with an additional server that runs the Stage simulator.

3. SOME PRELIMINARY EXPERIMENTS

In this section we present a first set of experiments that aim at addressing types of analyses that are not typically taken into account in experimental evaluation of coordination and cooperation in multi robot systems. The work reported here is not meant to be exhaustive; a detailed analysis of the influence of various aspects of robotic performance on the effectiveness of teamwork is on-going work.

First of all, we focus on an exploration task, that is inspired to a de-mining application. Thus, robots operate outdoor and their common goal is to check the area for the presence of a (simulated) target (e.g., a heat source); the robots are provided with a set of short-range sensors to detect the target (e.g. measure the temperature). Once the target is found, the robots are required to coordinate in order to dynamically build small groups that should act upon the target (e.g., a robot marks the zone, another takes a picture, etc.). The area to be explored is discretized according to a grid of cells (size 4x4 meters). Each target can be identified only from the cell where it is located.

As already mentioned, the goal of our system is to allow for the analysis of the performance of different approaches and features of MRS teamwork, when varying both the environment and the robot capabilities. In order to evaluate the performance of the system, we consider the following measures: time to finish the mission (i.e., to explore the whole area), number of heat sources found (wrt their total number), percentage of total area to explore.

We present three sets of experiments. In the first, we vary the number of robots (2-12), operating on different-sized areas. The results of the experiment show that the proposed approach does not degrade the performance, when the explored area and the number of robots are increased consistently.

The second set of experiments shows the behavior of the system with respect to different localization errors. In this case, we observed three different behaviors when the localization error is increased: the robots explored cells that were outside the assigned area; sometimes they were not able to detect duplicated tasks and thus explored the same area more than once; finally, some cells have been skipped in the exploration.

In the third set of experiments, we change the maximum navigation speed that is allowed for each robot. The performance evaluation of these tests shows that, as expected, there is an optimal speed limit, and if the speed overcomes this limit, the performances degrades, because the navigation algorithm is not able to steer the robot.

4. REFERENCES

[1] D. Calisi, F. Fedi, A. Leo, and D. Nardi. Software development for networked robot systems. In *Proc. of the 7th IFAC Symposium on Intelligent Autonomous Vehicles (IAV)*, 2010.

[2] T. Collet, B. MacDonald, and B. Gerkey. Player 2.0: Toward a practical robot programming framework. In *Proc. of the Australasian Conf. on Robotics and Automation (ACRA 2005)*, Dec. 2005.

[3] B. Gerkey and M. J. Matarić. A formal analysis and taxonomy of task allocation in multi-robot systems. *Int. journal of robotic research*, 23(9):939–954, Sept. 2004.

[4] K. Konolige, C. Ortiz, R. Vincent, A. Agno, B. Limketkai, M. Lewis, L. Briesemeister, D. Fox, J. Ko, B. Stewart, and L. Guibas. CentiBOTS: large scale robot teams. In *Proceedings of the International Conference on Autonomous Agents and Multi Agent Systems (AAMAS)*, 2003.

[5] O. Michel. Webots: professional mobile robot simulation. *International Journal of Advanced Robotic Systems*, 1(1):39–42, 2004.

[6] R. T. Vaughan. Massively multi-robot simulations in stage. *Swarm Intelligence*, 2(2-4):189–208, 2008.