MARS 2007

Peter Sapaty and
Joaquim Filipe (Eds.)

Multi-Agent Robotic Systems

Proceedings of the
3rd International Workshop on
Multi-Agent Robotic Systems - MARS 2007

In conjunction with ICINCO 2007
Angers - France, May 2007

INSTICC PRESS

Peter Sapaty and
Joaquim Filipe (Eds.)

Multi-Agent Robotic Systems

Proceedings of the
3rd International Workshop on
Multi-Agent Robotic Systems
MARS 2007

In conjunction with ICINCO 2007
Angers, France, May 2007

INSTICC PRESS
Portugal

Volume Editors

Peter Sapaty
The University of Aizu
Aizu-Wakamatsu, Japan

and

Joaquim Filipe
Polytechnic Institute of Setúbal / INSTICC
Setúbal, Portugal

3rd International Workshop on
Multi-Agent Robotic Systems
Angers, France, May 2007
Peter Sapaty and
Joaquim Filipe (Eds.)

Printed in Portugal

ISBN: 978-972-8865-86-6
Depósito Legal: 257880/07

Foreword

This volume contains all the papers presented at MARS-2007 (*3rd International Workshop on Multi-Agent Robotic Systems*), held in Angers/France, in May 2007, as a satellite workshop of the ICINCO-2007 conference (*International Conference on Informatics in Control, Automation and Robotics*) which was co-organized by INSTICC (Institute for Systems and Technologies of Information, Control and Communication) and the University of Angers.

The *3rd International Workshop on Multi-Agent Robotic Systems* intends to be a forum for the debate of issues concerning all kinds of theoretical and applied studies combining the interdisciplinary areas of multi-agent systems and robotics, with communicating agents, flexibly materialized in either software or hardware - the latter as mobile robots. The workshop papers included in these proceedings spread across a number of topic areas including the following: autonomous and emergent formations, swarm robotics, cooperative robotics, human-robot communication and coordination, mobile robot teams, maps and navigation, robotic soccer and other types of multi-agent system aspects related to robotics, like agent-oriented design, evolution and self-recovery, behavior planning and task allocation, distributed administration and maintenance, mission reliability and simulation, artificial intelligence, targets search, sensing, convergence, and others. After a double blind review process a total of 9 papers out of 17 submitted were selected for oral presentations, arranged in 3 sessions of a single track program; 3 were selected as poster presentations.

We would like to express our sincere gratitude to all the authors, who provided the rich material discussed at the workshop, and the members of the Program Committee who have reviewed and assessed the scientific merit of each submitted paper, thus ensuring high quality standards. Last but not least, thanks to Marina Carvalho for her effort in the secretariat support, and to Bruno Encarnação and Vitor Pedrosa for producing the proceedings.

Joaquim Filipe
Polytechnic Institute of Setúbal / INSTICC, Portugal

Peter Sapaty
Institute of Mathematical Machines and Systems, NAS, Ukraine

Workshop Chairs
Peter Sapaty
The University of Aizu
Aizu-Wakamatsu, Japan

and

Joaquim Filipe
Polytechnic Institute of Setúbal / INSTICC
Setúbal, Portugal

Program Committee

Ronald Arkin, Georgia Tech, U.S.A.
Nicolas Bredeche, LRI, France
Alfred Bruckstein, Technion - IIT, Israel
Luca Cernuzzi, Universidad Católica, Paraguay
Anibal Castilho Coimbra de Matos, Faculdade de Engenharia da Universidade do Porto (FEUP), Portugal
Urbano Nunes, University of Coimbra, Portugal
Michel Occello, Université Pierre-Mendès-France, France
Mohan Sridharan, University of Texas at Austin, U.S.A.
Edward Tunstel, NASA Jet Propulsion Laboratory, U.S.A.
Paul Vogt, University of Edinburgh, U.K.

Table of Contents

Full Papers

Posters

FULL PAPERS

A Hybrid Dynamic Task Allocation Approach for a Heterogeneous Multi-Robot System

Yan Meng and Kashyap Shah

Department of Electrical and Computer Engineering
Stevens Institute of Technology, Hoboken, NJ 07030, USA
yan.meng@stevens.edu, kshah18@stevens.edu

Abstract. In this paper, we propose a communication-efficient hybrid task scheduling algorithm for a heterogeneous multi-robot system under dynamic unknown environment, where each robot makes its own decision through communicating with others as well as checking a global task status queue. The proposed hybrid algorithm takes advantage of centralized approaches to improve the overall system efficiency and distributed approaches to reduce the communication overhead, which automatically leads to a reasonable reduction of power consumption. This algorithm avoids unnecessary communication by broadcasting global information which is in everybody's interest and meanwhile limits specific information which is in interest of some specific robots only. In addition, each robot would dynamically allocate the task to robots which are capable and most available. This feature makes the system robust against communication failures and robot failures. Simulation results demonstrate the efficiency and robustness of the proposed approach.

1 Introduction

With growing need of building reliable real-time applications coupled with advancement of high-speed networks and high-performance computers, in the past decade heterogeneous multi-robot systems have been increasingly used for many real-time applications, like urban search and rescue, surveillance, hazardous materials detection, and reconnaissance, in which the correctness of the systems depend not only on the results of a computation, but also on the time which these results are produced [1]. To achieve real-time performance of such a complex system, an efficient task allocation and coordination among the team members is required. Vali Veloso stated in [2] that team performance can be drastically increased if the team coordinates well and the information is being shared by all teammates in a multi-robot environment.

Dynamic task allocation for multi-robot systems under dynamic environment is a challenging problem, which aims to efficiently finish all of unknown tasks as fast as possible while keep the cost as low as possible. Although some algorithms have been proposed to tackle this problem, such as auction-based algorithm like MURDOCH [3][4], behavior-based algorithm like ALLIANCE [5][6], and instantaneous greedy scheduler based approaches, all of these available methods have a great deal of broadcast communication overhead to share information with all of team members.

Some of available algorithms are only good for a homogeneous robot team with one global task like mapping or exploration of an area, and some of them don't take consideration of system robustness in the case of communication failure or robot malfunctions.

In this paper, we aim at investigating a task scheduling algorithm for a heterogeneous multi-robot system under dynamic unknown environment. As we know, a centralized approach consists of making all decisions in one place, where all the tasks to be performed are collected by a central scheduler. This centralized scheduler decomposes tasks into programs of actions, order actions when necessary and assigns them to robots with respect to their capabilities, work loads, and locations. The centralized approach is efficient with small number of agents, but its performance would be degraded significantly in a large-scale team. Furthermore, centralized approaches are not appropriate for coordinating the action of multiple robots in a dynamic unknown environment where unforeseeable events may occur.

On the other hand, in a decentralized approach, each robot makes its own decisions for a particular set of tasks. No central unit is needed. Some initial decomposition of the global scheduling may be imposed and robots can negotiate with others to make the best of coordination and solve conflicts dynamically. Furthermore, to improve the system robustness, error handling and system recovery are critical issues. According to Dias and Zink [7], in a multi-robot environment, system failure can occur in three different ways: (1) communication failure; (2) partial robot malfunctioning; and (3) robot death. The scheduling algorithm should take these situations into consideration.

Based on the above observation, we propose a hybrid task scheduling algorithm, where each robot makes its own decision through communicating with others as well as checking a global status queue to improve the coordination efficiency. This algorithm avoids unnecessary communication by broadcasting global information which is in everybody's interest and meanwhile limits specific information which is in interest of some specific robots only. Therefore, the proposed algorithm takes advantage of centralized approaches to improve the overall efficiency and distributed approaches to reduce the communication overhead. To improve the system robustness under dynamic environment, instead of making each robot to adapt to some unexpected tasks which may be beyond its own capability due to changed environment, the robot would send help signals to those who can handle the tasks. In addition, by tracking the communication signal that it has sent and expected to receive, each robot would dynamically allocate the tasks to robots which are capable and most available. This feature makes the system robust against communication failures and robot failures.

The paper is organized as follows. Section II introduces background and related work in the field of task allocation algorithms for multi-robot systems. Section III describes the problem statement. Section IV proposes a real-time dynamic task allocation algorithm for heterogeneous multi-robot systems. Extensive simulation results are discussed in Section V. The paper is concluded by Section VI.

2 Related Work

Increasing amounts of research have been conducted in the area of dynamic task scheduling for multi-robot systems. One of the easiest approaches to work in dynamic task assignment is trial and error method, where all robots will try the same task one by one until the perfect match is found. This method is very inefficient and time consuming. James McLurkin [8] proposed three different methods of task assignment in robot swarms: random-choice algorithm, which is extremely communication extensive and inefficient, card-dealer's algorithm, which assigns tasks to individual robots sequentially, using minimal communications but a great deal of time, and tree-recolor algorithm, which is a compromise between extreme-communication and card-dealer's, balancing communications use and running time.

Ashely and Ramprasad [9] proposed a behavior-based planning algorithm for multi-robot systems, where each robot predicts the behavior of its companion and proceeds for further steps. The main idea of this method is that the robot should not try to adapt to the situation but instead should directly transfer that task to an associated robot who can handle that situation. Brumitt and Stentz [10] proposed a dynamic mission planning for multi-robot systems in a dynamic environment, where the planning system dynamically reassign robots to goals in order to continually minimize the time to complete the mission. Trade-offs between robot's traveling cost and running cost of mission planner has to be balanced. Smith and Davis [11] proposed a contract net protocol, where the collection of nodes (robot) can be represented as a contract net. There are many auction based methods available for handling dynamic task allocation and MURDOCH [3][4] is a popular one among them, which uses contract net protocol as its communication protocol. Generally, distributed systems rely on fitness based actions and negotiation protocols. MURDOCH uses publish/subscribe messaging for distributed control of multi-robot systems.

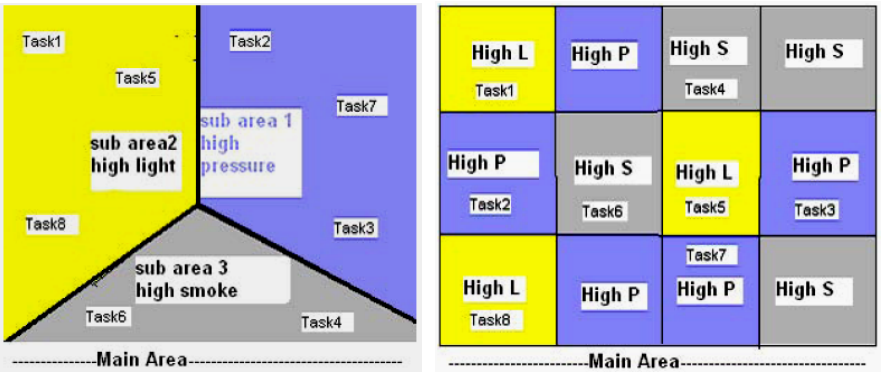
A task-assignment architecture was proposed in [12] for cooperative transport by multiple mobile robots in an unknown static environment, which consists of two real-time planners: a priority-based task-assignment planner and motion planners based on short-time estimate. This method is also compared with Stillwell's algorithm in [13], where homogenous robots are ant like objects who try to move a big piece of food from one place to their nest. Each of them tries to contribute in the most efficient way. The scheduler proposed in [14] is one example of greedy decentralized schedulers. Generally, these kinds of co-operative search approaches are efficient and robust in applications like military scouting and automatic trash collection. A novel emotion-based recruitment approach was proposed in [15] for a multi-robot task allocation problem. Affective recruitment is tolerant of unreliable communication channels, and can find better solutions than simple greedy schedulers.

3 Problem Statement

A simplified proof-of-concept task environment, as shown in Fig. 1, is divided into 3 different sub-areas: high-pressure sub-area, intensive-light sub-area, and smoking sub-area. Different types of robots are defined based on their capabilities. For exam-

ple, *P type robot*, which is only capable of working in high-pressure subarea, but not the other two subareas. Here *P* stands for pressure. Similar definitions are applied to *L type robot* and *S type robot*. Some robots may have capabilities suitable for multiple sub-areas, such as *PS type* or *LS type*. In order to implement the assigned missions in a more efficient manner, task preemption is not allowed among the robots in the proposed task allocation approach. It is assumed that robots are autonomous, are able to localize themselves within the environment, can avoid obstacles and plan path to a destination.

Consider that we have *N* heterogeneous robots and *M* different tasks randomly distributed in different sub-areas. Here task is a conceptual terminology, which can be defined as various physical jobs, such as trash can collection, de-mining, transportation, construction, or assembling. The robots are expected to move to the position where the tasks are located and process the task. The environment can be as simple as Fig, 1(a) 1 or as complex as Fig. 1(b), which is unknown to robots. However, it is assumed that each robot has on-board sensors capable of detecting different subareas. Initially, if some predefined tasks have been stored in a robot, it will move to those tasks. If no predefined task exists, robots would randomly move around to search for the tasks in the environment. The requirement of these tasks may be changed due to dynamic environment. The objective of this project is to develop an efficient task scheduling algorithm among heterogeneous robots under dynamic environment so that all of the tasks would be completed as soon as possible meanwhile cost (i.e. power consumption) can be reduced as low as possible.



(a) (b)

Fig. 1. Possible task environments.

4 A Hybrid Approach

To tackle this scheduling problem, a hybrid centralized and decentralized method is proposed, where each robot makes its own decision, communicates with others to share task information, as well as to check a global task status queue to improve the coordination efficiency. The architecture of the approach is shown in Fig. 2.

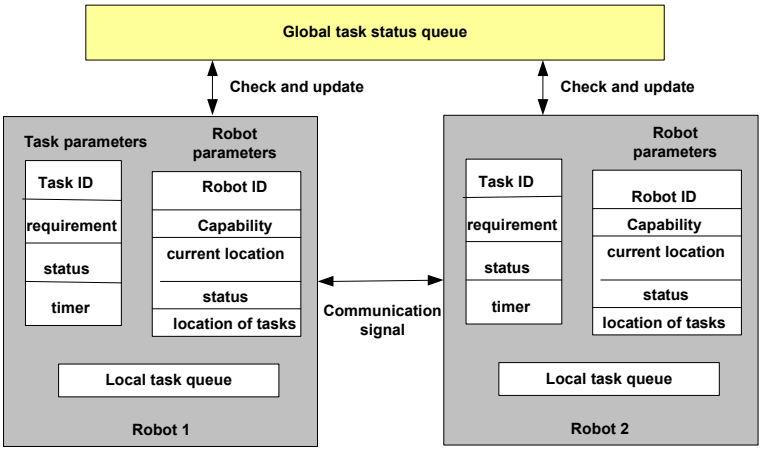


Fig. 2. The architecture of the hybrid approach.

Each robot has a local database to keep all the information it required to make decisions. This database structure, as shown in Fig.2, includes three major parts: robot parameters, task parameters, and local task queues. Robot parameters consist of robot ID, capability vectors of all robots, current locations of all robots relative to a reference coordinate system, status of all robots, locations of all tasks. Task parameters consist of task ID, task requirement, task status, and task timer.

Robot ID is a unique identification number for each robot. Capability is robot's ability to perform a task, which is a combination of different sub-areas represented by a capability vector. Robot status consists of free (its local task queue is empty), busy (some tasks are in its local task queue), or failed. Task ID is represented by its physical location in a reference coordinate. Task requirements depend on the sub-area where the task located. Task status shows whether the task is in progress, completed, in-trouble, or time-out. Task timer is used to track how long the task has been processed. To prevent the system to be hanged by one task forever, if the processing time is greater than a predefined threshold, time-out status would be labeled on the task.

Local task queue keeps a list of tasks a robot will perform sequentially. These tasks may be some predefined tasks before the system starts, or tasks detected or reassigned on the fly. Once the robot finishes its first task in its task queue, it would remove the finished task and go to next one until the last task in the queue. Once this queue becomes empty, robot will start moving randomly to search for a new task. Since it is difficult to predict all tasks in advance, some predefined tasks may not be appropriate for a robot anymore under dynamic environment. If a robot finds out that it is difficult for it to process a task in its local queue during execution, it would send help request to those robots whose capabilities match the task requirement. If help responses are received, the robot will assign the task to the responded robot, and delete it from its local queue. Meanwhile the responded robot would add that task in its local queue.

Global task status queue is a queue in which robot keeps the information about all tasks being processed or completed. The main purpose of this global task status queue is to prevent any unnecessary redundancy among robots to process the same task.

This global queue will be updated by all robots whenever task status has been changed.

Initially, all robots move around randomly in the environment searching for tasks. When a robot detects a task, it checks the requirements of the task first. If the requirements of the task match with its own capability, then the robot would perform the task and update the global task status queue. When a robot completes a task, it would update the global task status queue about its current task state. The states of a task include beginning, completion, in-trouble, and time-out. This global status of tasks is stored in the global task status queue for all robots. Whenever a robot needs help, it only broadcasts helps to those capable robots instead of everyone. Here a trade-off between memory capacity of robots and communication overhead among robots has to be made. The system stops when the global task status queue is filled up with all tasks with status of completion.

If multiple robots respond to the help requesting signal, the helper needs to make decision which robot to pick. On the other hand, if a robot was selected by more than one helper, it also need to make decision which task to select (if more than one responded robots) or which task to put into its local task queue first (if only one robot can do these tasks). A fitness function is required for this decision making. Here, a auction-based method is applied, which is defined as follows:

$$F(c_i, s_i, d_i) = k \frac{f(c_i \mid t_j) f(n_i)}{d_i}, i = 1,2,..., N, j = 1,2,..., M. \quad (1)$$

Where c_i, d_i and n_i represent the capability, distance from the current task location, and number of tasks in local queue of robot *i*, respectively. t_j represents the task types, and *k* is a scale factor. $f(c_i \mid t_j)$ is a matching function of the capability of robot *i* related to the type of task *j*, which is defined as follows:

$$f(c_i \mid t_j) = \begin{cases} 1, & \text{if the robot capability matches the task type} \\ 0, & \text{otherwise} \end{cases}.$$

And $f(n_i)$ is defined as follows:

$$f(n_i) = \begin{cases} 1, & \text{if } n = 0 \\ 1/n, & \text{otherwise} \end{cases}$$

In other words, if all of the responding robots are busy, the numbers of tasks in their local task queues are compared. The smaller the task number in queue, the higher the possibility of the corresponding robot would be selected as the helper.

Since robots need share task information with others, a specific communicate protocol is required for this application. Basically, four types of signal frames are defined in the communication protocol, (1) help requesting signal frame; (2) help responding signal frame; (3) help accepting signal frame; and (4) global task updating signal frame. The detailed frame definitions are shown in Fig. 3. When a help seeking robot receives help responding signal, it would send help accepting signal back to the selected robot. If responder robot is busy at that time, it would add that task in its local task queue and continue working on its current task. Global task updating signal is used to update the global task queue.

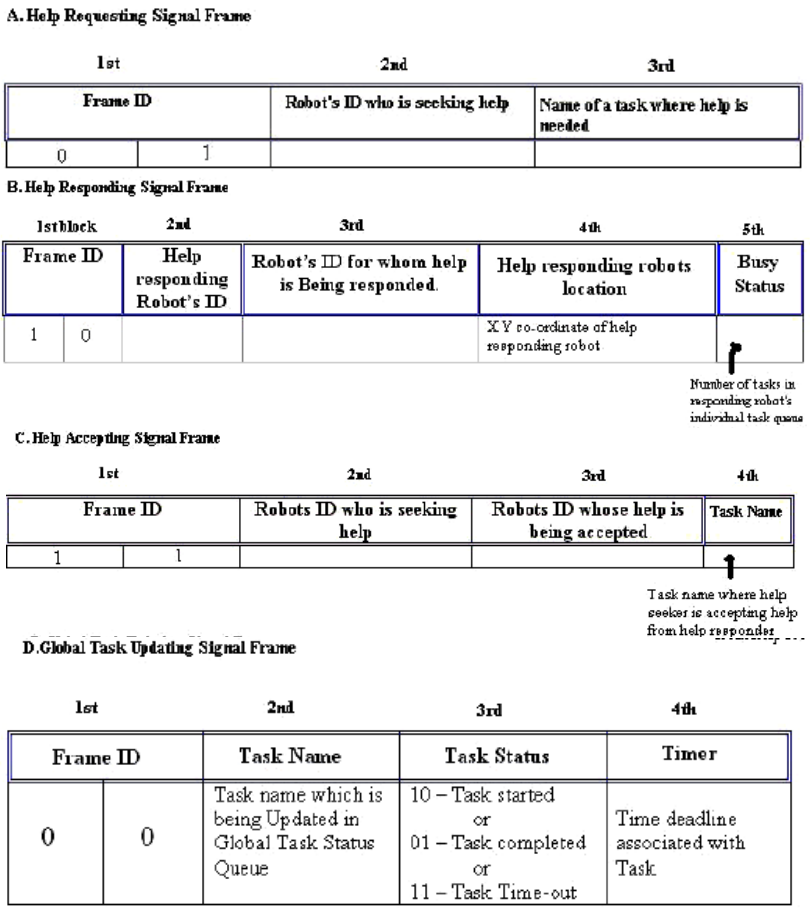


Fig. 3. Communication protocols among robots.

5 Simulation Results

To evaluate the proposed algorithm, a simulator is developed using C/C++ language under Windows environment, and a snapshot of the simulation screen is shown in Fig. 4. Six robots are employed in the simulation including 2 P-type (represented as Rp1 and Rp2), 2 L-type (RL1 and RL2), and 2 S-Type (Rs1 and Rs2). Eight tasks are generated, which are represented by the location coordinate within a reference frame. T1(X1,Y1), T2(X2,Y2) and T3(X3,Y3) are in high-pressure sub-area, T4(X4,Y4), T5(X5,Y5) and T6(X6,Y6) in smoking sub-area, and T7(X7,Y7) and T8(X8,Y8) in intensive-light sub-area. The local task queue is located on top-left, and the global task status queue is listed on top-right. The bottom-left part indicates the communication signals sent or received by robots. Various geometric shapes in the task environment represent static obstacles.

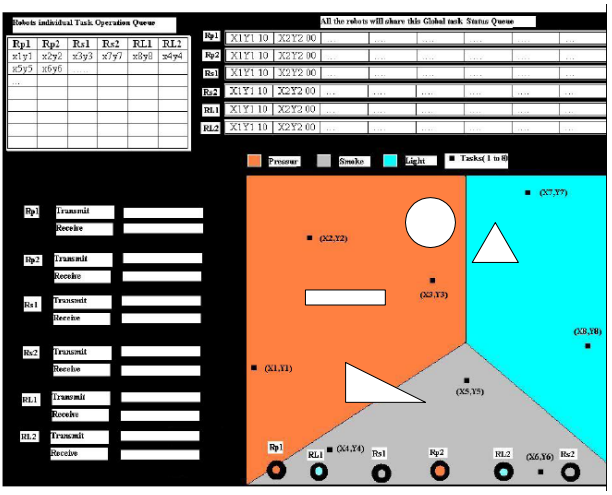


Fig. 4. A snapshot of the simulation screen.

Most task allocation problems among robots have applied group-wide broadcast communication to share the information and negotiate with team members. This kind of broadcast communication usually requires more communication overhead, power, time and cost, especially for a heterogeneous team where only some specific members can conduct specific types of tasks, not all of them. In our algorithm, instead of broadcast to everyone, the help information is only broadcasted to those which have the capability for the task. The communication cost comparison results are shown in Fig. 5, where the communication cost of our approach (i.e. 23 times) has been significantly reduced compared with the group-wide broadcasting method (i.e., 35 times). Communication overhead is directly proportional to task processing time and power consumption. In other words, our approach would be more power efficient and spend less time to finish the tasks than the group-wide broadcasting method. For this simple example with 6 robots and 8 tasks, the time required to finish all tasks are shown in Fig. 5(c).

A simple auction-based method using broadcasting is applied for comparison, where robots randomly search for tasks and broadcast the task information to all team members. If one robot needs help and more than one response received, the robot which is closest to the task will be selected. Four cases of different task distributions are designed in the simulation, where 8 task locations are re-distributed in different cases. The time required to finish all tasks under different task configuration cases are recorded and shown in Fig. 6. The proposed algorithm obviously outperforms the random searching one. The proposed method selects the helper robot not only depends on its current distance to the task, but also its current status. If there is a long list of tasks in its local task queue of a robot, even if it is closest one to the task, it may end up selecting other robot with a much shorter list of tasks in local queue.

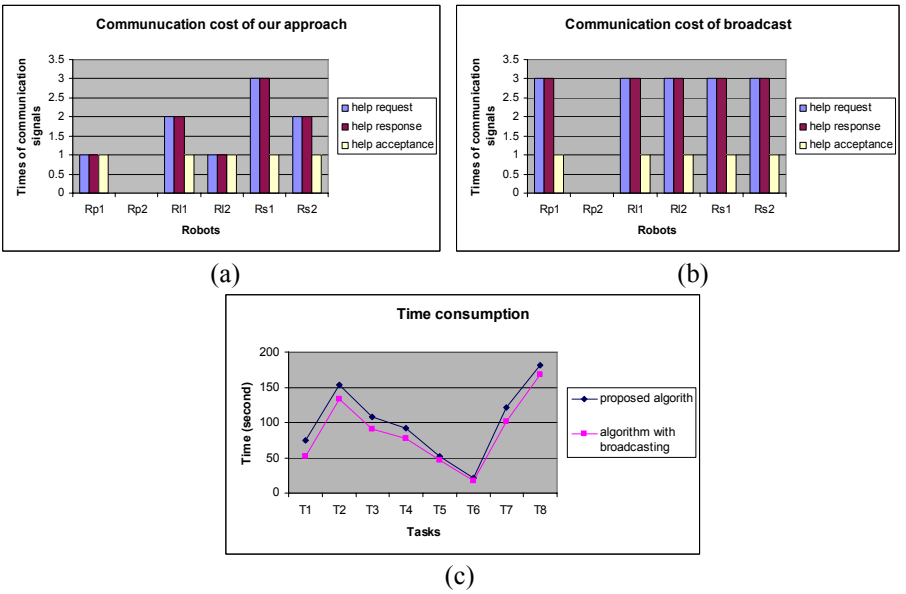


Fig. 5. Comparison of communication overhead.

To evaluate the robustness of the proposed algorithm under the failure situations, such as communication failure and robot failure, another set of simulation results with the same four task distributions as in previous experiments are shown in Fig. 7. It is assumed that the communication failure happens once for a while due to the environment or temporary traffic jam. It can be seen that the communication failure didn't affect the system performance extensively. This is because once a robot detects a communication failure, it would send signals again in next cycle until the acknowledgement is received, which prevents the signal loss due to the communication failure. In this simulation, it is assumed that Rs2 is dead, which means that all of tasks in smoking sub-area have to be conducted by Rs1. The simulation results show that the system performance degraded at some level with a failure robot instead of being stuck forever.

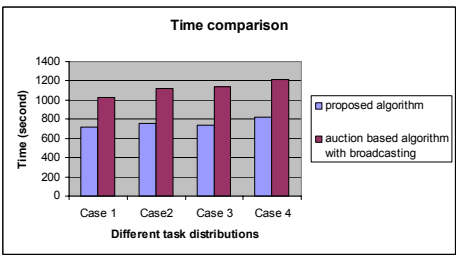


Fig. 6. Simulation comparison.

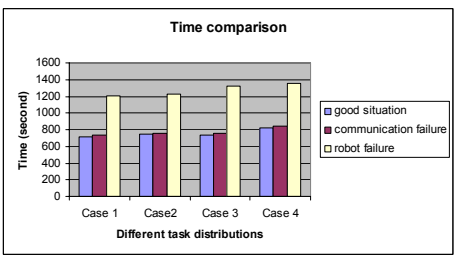


Fig. 7. Simulation with system failures.

6 Conclusions and Future Work

In this paper, a hybrid task scheduling approach has been proposed, which significantly reduces communication overhead while improving the overall system performance through dynamic task allocation. This algorithm avoids unnecessary communication by broadcasting global information which is in everybody's interest and meanwhile limits specific information which is in interest of some specific robots only. Each robot would dynamically allocate a task which is difficult for itself to other capable and most available robots, and keeps tracking the help requests, which makes the system robust against communication failures and robot failures. Simulation results show robot communication overhead can be significantly reduced, which automatically leads to reduction of power consumption and time consumption. In our future work, more dynamic situations will be considered, such as malicious agents, dynamically adding to or removing agents from the current team, global update failures. Furthermore, the method will be implemented to a real-world multi-robot system, where robot dynamics, kinematics, robot-robot interaction and sensors would have to be considered.

References

1. W. A. Halang, R. Gumzej, M. Colnaric, and M. Druzovec, "Measuring the performance of real-time systems", Journal of Real-Time Systems, 2000, 18(1):59-68.
2. Vail and Veloso, "Dynamic multi-robot coordination," *Multi- Robot Systems: From Swarms to Intelligent Automata, vol*. II, pp. 87–100, 2003.
3. B. Gerkey and M. J. Mataric, "Murdoch: Publish/subscribe task allocation for heterogeneous agents," *Proceedings of Autonomous Agents,* Barcelona, Spain, June 3-7, pp. 203–204, 2000.
4. D. Rus, S. Singh, S.-V. B. Heidelberg, B. P. Gerkey., and M. J. Matarik, "Principal communication for multi robot task allocation," *Experimental Robotics VII,* LNCIS, pp. 353–362, 2001.
5. L. E. Parker, "Alliance: An architecture for fault tolerant multirobot cooperation," *IEEE Trans. Robot. Automat, vol. 14,* p. 220240, Apr 1997.
6. L. E. Parker, "Task-oriented multi-robot learning in behavior-based systems," *Proceedings of the IEEE/RSJ International Conference on Intelligent Robots and Systems,* 1996.
7. M. B. Dias, M. Zinck, R. Zlot, and A. Stentz, "Robust multirobot coordination in dynamic environments," *Proceeding on DIAS,.*2004.
8. J. McLurkin and D. Yamins, "Dynamic task assignment in robot swarms,"*IEEE Proceedings of Robotics: Science and Systems, June 2005*. MIT Computer Science and Artificial Intelligence Lab.
9. A. W. Stroupe, R. Ravichandran, and T. Balch, "Value-based action selection for exploration and dynamic target observation with robot teams," *Proceedings of the IEEE Conference on Robotics and Automation,* 2004.
10. B. Brumitt and A. Stentz, "Dynamic mission planning for multiple mobile robots," *Proceedings of the IEEE International Conference on Robotics and Automation,* 1996.
11. R. Davis and R. G. Smith, "Negotiation as a metaphor for distributedo," *Conf. Artificial Intelligence Laboratory, Massachusetts Institute, of Technology, Cambridge, MA* 02139, ETATS-UNIS, vol. 20, no. 1, p. 63109, 1993.

12. N. Miyata, J. Ota, T. Arai, and H. Asama, “Cooperative transport by multiple mobile robots in unknown static environments associated with real-time task assignment,” *IEEE transactions on robotics and automation,* vol. 18, no. 5, 2002.
13. D. J. Stilwell and J. S. Bay, “Toward the development of a material transport system using swarms of ant-like robots,” *Proc. IEEE Int. Conf. Robotics and Automation*, pp. 766-771, 1993.
14. C. Hsieh and R. Murray, “Experimental implementation of an algorithm for cooperative searching of target sites,” *Proceedings of American Control Conference,* 2005.
15. A. Gage, D. R. Murphy, D. K. Valavanis, and M. Long, “Affective task allocation for distributed multi-robot teams,” Center for Robot-Assisted Search and Rescue (CRASAR), TR2006-26.

A Bio-inspired Multi-Robot Coordination Approach

Yan Meng, Ọlọrundamilọla Kazeem and Jing Gan

Department of Electrical and Computer Engineering
Stevens Institute of Technology, Hoboken, NJ 07030
yan.meng@stevens.edu, okazeem@stevens.edu, jgan@stevens.edu

Abstract. In this paper, a bio-inspired stigmergy-based coordination approach is proposed for a distributed multi-robot system. This approach is inspired from the behavior of social insect swarming, where social insect colonies are able to build sophisticated structures and regulate the activities of millions of individuals by endowing each individual with simple rules based on local perception. A virtual pheromone mechanism is proposed as the message passing coordination scheme among the robots. The proposed algorithm has been implemented on embodied robot simulator Player/Stage, and the simulation results show the feasibility, robustness, and scalability of the methods under different dynamic environments with real-world constraints.

1 Introduction

The main challenges for swarm robots are to create intelligent agents that adapt their behaviors based on interaction with the environment and other robots, to become more proficient in their tasks over time, and to adapt to new situations as they occur. Typical problem domains for the study of swarm-based robotic systems include foraging [1], box-pushing [2], aggregation and segregation [3], formation forming [4], cooperative mapping [5], soccer tournaments [6], site preparation [7], sorting [8], and collective construction [9]. All of these systems consist of multiple robots or embodied simulated agents acting autonomously based on their own individual decisions. However, not all of these control architectures are scalable to a large number of robots. For instance, most approaches rely on extensive global communication for cooperation of swarm robots, which may yield stressing communication bottlenecks. Furthermore, the global communication requires high-power onboard transceivers in a large scale environment. However, most swarm robots are only equipped very limited sensing and communication capability.

An alternative paradigm to tackle the scalability issue for swarm robots while maintaining robustness and individual simplicity is through Swarm Intelligence (SI), which is an innovative computational and behavioral metaphor for solving distributed problems by taking its inspiration from the behavior of social insects swarming, flocking, herding, and shoaling phenomena in vertebrates, where social insect colonies are able to build sophisticated structures and regulate the activities of millions of individuals by endowing each individual with simple rules based on local perception.

The abilities of such natural systems appear to transcend the abilities of the con-

stituent individual agents. In most biological cases studies so far, robust and coordinated group behavior has been found to be mediated by nothing more than a small set of simple local interactions between individuals, and between individuals and the environment.

Reynold [10] built a computer simulation to model the motion of a flock of birds, called *boids*. He believes the motion of the boids, as a whole, is the result of the actions of each individual member that follow some simple rules. Ward et al. [11] evolved *e-boids*, groups of artificial fish capable of displaying schooling behavior. Spector et al. [12] used a genetic programming to evolve group behaviors for flying agents in a simulated environment. The above mentioned works suggest that artificial evolution can be successfully applied to synthesize effective collective behaviors. And the swarm-bot [13] developed a new robotic system consisting of a swarm of s-bots, mobile robots with the ability to connect to and to disconnect from each other depends on different environments and applications, which is based on behaviors of ant systems. Another swarm intelligence based algorithm, Particle Swarm Optimization (PSO), was proposed by Kennedy and Eberhart [14]. The PSO is a biologically-inspired algorithm motivated by a social analogy, such as flocking, herding, and schooling behavior in animal populations.

Payton et al. [15] proposed pheromone robotics, which was modeled after the chemical insects, such as ants, use to communicate. Instead of spreading a chemical landmark in the environment, they used a virtual pheromone to spread information and create gradients in the information space. By using these virtual pheromones, the robots can send and receive directional communications to each other.

In this paper, we propose a bio-inspired coordination paradigm to achieve an optimal group behavior for multi-agent systems. Each agent adjusts its movement behavior based on a target utility function, which is defined as the fitness value of moving to different areas using the onboard sensing inputs and shared information through local communication. Similar to [15], inspired by the pheromone drip trail of biological ants, a unique virtual agent-to-agent and agent-to-environment interaction mechanism, i.e. *virtual pheromones,* was proposed as the message passing coordination scheme for the swarm robots. Instead of using infrared signals for transceivers in [15], which requires line of sight to transmit and receive, we use wireless ad hoc network to transmit information and the virtual pheromone structure is designed to be more robust and efficient.

This new meta-heuristic draws on the strengths of two popular SI-based algorithms: Ant Colony Optimization (ACO)'s autocatalytic mechanism and Particle Swarm Optimization (PSO)'s cognitive capabilities through interplay. Basically, two coordination processes among the agents are established in the proposed architecture. One is a modified stigmergy-based ACO algorithm using the distributed virtual pheromones to guide the agents' movements, where each agent has its own virtual pheromone matrix, which can be created, enhanced, evaporated over time, and propagated to its neighboring agents. The other one is interaction-based algorithm, which aims to achieve an optimal global behavior through the interactions among the agents using the PSO-based algorithm.

In our previous work [16], this hybrid algorithm was implemented in a proof-of-concept simulator, where each agent has been simplified to one dot without any sensors installed. The target detection was based on the distance between the agent and

the target. Once the distance is within the detection range, it is assumed that the target is detected. To apply this algorithm to the real-world robotic systems, more realistic issues need to be solved. In this paper, we will modify the utility function of the previous proposed hybrid ACO/PSO algorithm to adapt to a more realistic robotic simulation environment. Player/Stage is utilized as our embodied robot simulator. Each robot in Player/Stage is installed with a camera system, a laser range finder, sonar sensor, and wireless communication sensor. The strength of this ACO/PSO coordination architecture lies in the fact that it is truly distributed, self-organized, self-adaptive, and inherently scalable since global control or communication is not required. Each agent makes decisions only based on its local view, and is designed to be simple and sometimes interchangeable, and may be dynamically added or removed without explicit reorganization, making the collective system highly flexible and fault tolerant.

The paper is organized as follows: Section II describes the problem statement. Section III presents the proposed stigmergy-based architecture for distributed swarm robots. Section IV presents the simulation environment and simulation results. To conclude the paper, section V outlines the research conclusion and the future work.

2 Problem Statement

The objective of this study is to design a bio-inspired coordination algorithm for distributed multi-robot systems. To evaluate this coordination algorithm, a multi-target searching task in an unknown dynamic environment is implemented in Player/Stage simulator. The targets can be defined as some predefined tasks need to be processed by the agents in real-world applications, for example, collective construction, resource/garbage detection and collection, people search and rescue, etc.. The goal is to find and process all of the targets as soon as possible. Assume that the agents are simple, and homogeneous, and can be dynamically added or removed without explicit reorganization. Each agent can only communicate with its neighbors. Two agents are defined as neighbors if the distance between them is less than a pre-specified communication range. The agent can only detect the targets within its local sensing range.

3 A Stigmergy-Based Coordination Approach

3.1 Virtual Pheromone as Inter-Agent Communication Mechanism

The ACO algorithm, proposed by Dorigo et al. [13], is essentially a system that simulates the natural behavior of ants, including mechanisms of cooperation and adaptation. The involved agents are steered toward local and global optimization through a mechanism of feedback of simulated pheromones and pheromone intensity processing. It is based on the following ideas. First, each path followed by an ant is associated with a candidate solution for a given problem. Second, when an ant follows a path, the amount of pheromone deposit on that path is proportional to the quality of

the corresponding candidate solution for the target problem. Third, when an ant has to choose between two or more paths, the path(s) with a larger amount of pheromone are more attractive to the ant. After some iterations, eventually, the ants will converge to a short path, which is expected to be the optimum or a near-optimum solution for the target problem.

In the classical ACO algorithm, the autocatalytic mechanism, i.e. pheromone dropped by agents, is designed as an environmental marker external to agents, which is an indirect agent-to-agent interaction design in nature. In the real world applications using swarm agents, a special pheromone and pheromone detectors need to be designed, and sometimes such physical pheromone is unreliable and easily to be modified under some hazardous environments, such as urban search and rescue. A redefinition of this auto catalyst is necessary.

A *Virtual Pheromone* mechanism is proposed here as a message passing coordination scheme between the agents and the environment and amongst the agents. An agent will build a virtual pheromone data structure whenever it detects a target, and then broadcast this target information to its neighbors through a visual pheromone package. Let $p_{a^k}(t) = \{p_{ij}^k(t)\}$ represents a set of pheromones received by agent k at time t, where (i, j) denotes the 2D global coordinate of the detected target. Each p_{ij}^k has a cache of associated attributes updated per computational iteration. The data structure for the virtual pheromone is defined as follows:

```
  Pheromone structure
  {  Target position;
     Number of target detected;
     The ID of source robot who detects the targets;
     The robot IDs that pheromone has been propagated before
passed to this robot;
     Agent intensity;
     Pheromone interaction intensity;
     Time stamp;}
```

3.2 Target Utility

Basically, each target is associated with different pheromone. Each agent makes its own movement decision based on the parameters of a list of pheromone matrix. Here, let's define target utility and target visibility to explain the decision making procedure of each agent.

First, let $\mu_k(t) = \{\mu_{ij}^k(t)\}$ represents a set of target utilities at time t, where $\mu_{ij}^k(t)$ denotes the *target utility* of agent k , which is defined as follows:

$$\mu_{ij}^k(t) = (k_1\omega_{ij}^k(t) - k_2\tau_{ij}^k(t))/R \tag{1}$$

where $\omega_{ij}^k(t)$ and $\tau_{ij}^k(t)$ represent target weight and agent intensity, respectively. Let the target weight measures potential target resources available for agent k at time t. The agent intensity is an indication of the number of agents who will potentially process the corresponding target at location (i,j). When we say "potentially", we mean all of the agents who have received the same pheromone information may end

up moving to the same target. However, they may also go to other targets with stronger pheromone intensity based on their local decisions.

We can use target intensity to emulate the pheromone enhancement and elimination procedure in natural world, which can be updated by the following equation:

$$\tau_{ij}^{k}(t+1) = \rho * (\tau_{ij}^{k}(t) + T_{ij}^{k}) - (1-\rho) * e * \tau_{ij}^{k}(t) \tag{2}$$

where $0<\rho<1$ is the enhancement factor of pheromone intensity. T_{ij}^{k} is the pheromone interaction intensity received from the neighboring agents for a target at (i,j), which is defined as

$$T_{ij} = \begin{cases} \alpha, \text{if source pheromone} \\ \beta, \text{otherwise} \end{cases} \tag{3}$$

where $0 \le \beta < \alpha \le 1$. If an agent discovers a target by itself instead of receiving the information from its neighbors, it is defined as the *source agent*. The source agent then propagates the *source pheromone*, to its neighbors. A propagation agent is a non-source agent, and simply propagates pheromones it received to its neighbors. Basically, T_{ij}^{k} is used for pheromone enhancement. e represents the elimination factor. In the ants system, the pheromone will be eliminated over time if it is not being enhanced by the ants, and the elimination procedure usually is slower than the enhancement. When the pheromone trail is totally eliminated, it means that no resource is available through this pheromone trail. To slow down the elimination relative to enhancement, we set $e < 1$.

R denotes local target redundancy, which is defined as the number of the local neighbors who have sent the pheromones referring to the same target at (i, j) to agent k. k_1 and k_2 are constant factors which are used to adjust the weights of target weight and agent intensity parameters.

Generally speaking, the higher the target utility is, the more attractive the corresponding target is to the agent. More specifically, when the target weight is greater than the agent intensity, it means that there are more tasks need to be processed (or there are more resources left) in this target. Therefore, the benefit of moving to this target would be higher in terms of the global optimization. If the agent intensity is greater than the target weight, it means that there will be more potential agents (globally) moving to this target, which may lead to the less available tasks (or resources) left in the future. Therefore, the benefit of moving to this target would be less in terms of the global optimization. With the local redundancy, we are trying to prevent the scenarios that all of the agents within a local neighbor move to the same target instead of exploring new targets elsewhere.

3.3 Target Visibility

Initially, the agents are randomly distributed in the searching environment, where multiple targets with different sizes and some static obstacles are randomly dispersed within the environment. At each iteration, if each agent adjusts its behavior based only on the target utility, it may lead the agent to be very greedy in terms of the

agents' behaviors, since the agents would rather move to the target with higher utility than explore new areas. This greedy behavior of the agents may easily lead to local optima.

To prevent the local optima scenarios in utility-based approach mainly based on ACO, we have to take into consideration of target visibility. Let $\eta_k(t) = \{\eta_{ij}^k(t)\}$ represents a set of visibilities at time *t*, where $\eta_{ij}^k(t)$ denotes the target visibility for agent *k* in terms of target at location (*i, j*), which is defined by the following equation:

$$\eta_{ij}^k(t) = r^k / d_{ij}^k(t) \tag{4}$$

where r^k represents the local detection range of agent *k*, and the $d_{ij}^k(t)$ represents the distance between the agent *k* and the target at location (*i, j*). If $\eta_{ij}^k > 1$, we set $\eta_{ij}^k = 1$. When the target visibility is higher, it means the distance between the target and the agent is smaller, it would be more benefit to move to this target due to its less cost compared to moving to the far-away target under the same environmental condition.

3.4 Agent Behavior Control

Now the question is how to integrate the target utility and target visibility into an efficient fitness function to guide the movement behaviors of each agent. To tackle this issue, we turned our attention to another collective intelligence - Particle Swarm Optimization (PSO). The PSO algorithm is population-based: a set of potential solutions evolves to approach a convenient solution (or set of solutions) for a problem. The social metaphor that led to this algorithm can be summarized as follows: the individuals that are part of a society hold an opinion that is part of a "belief space" (the search space) shared by every possible individual. Individuals may modify this "opinion state" based on three factors: (1) The knowledge of the environment (explorative factor); (2) The individual's previous history of states (cognitive factor); (3) The previous history of states of the individual's neighborhood (social factor).

A direct PSO adoption to swarm agents would be difficult, because swarm agents may be blinded over in reference to global concerns without any feedback. However, the PSO algorithm is a decision processor for annealing premature convergence of particles in swarm situations. Thus, a new optimization technique specifically tailored to the application of swarm agents is proposed in this paper. This new meta-heuristics draws on the strengths of both systems: ACO's autocatalytic mechanism through environment and PSO's cognitive capabilities through interplay among agents. In this hybrid method, the agents make their movement decisions not only based on the target utility defined in (3), but also on their movement inertia and their own past experiences, which would provide more opportunities to explore new areas.

Basically, the PSO algorithm can be represented as in (5), which is derived from the classical PSO algorithm [14] with minor redefinitions of formula variables as follows:

$$v_{ij} = explorative + cognitive + social \tag{5}$$

where v_{ij} is the velocity of a agent. To determine which behavior is adopted by agent *k* of the swarm, the velocity, $v_{ij}^k(t)$ has to be decided first. If the received pheromone intensity is high, the agent would increase the weight of social factor, and

decrease the weight of cognitive factor. On the other hand, if the local visibility is of significant to the agent, then the velocity of the agent would prefer the cognitive factor to the social factor. Furthermore, at any given time, the velocity of the agent would leave some spaces for the exploration of new areas no matter what. Therefore, the basic idea is to propel towards a probabilistic median, where explorative factor, cognitive factor (local agent respective views), and social factor (global swarm wide views) are considered simultaneously and try to merge these three factors into consistent behaviors for each agent. The exploration factor can be easily emulated by random movement.

The challenge part is how to define local best (cognitive factor) and global best (social factor). One straight forward method is to select the highest target visibility from a list of available targets as the local best. If only one target is on the list, then this target would be the local best. The easy way to select global best is to select the highest target utility from a list of available targets. If only one target is on the list, then this target would be the global best.

Instead of defining a fitness function, for a robot system, the robot velocity vector including both magnitude and direction would be a better representation to control the movement behavior. Based on the above discussion and PSO algorithm, each agent would control its movement behaviors by following this equation:

$$v_{ij}^{k}(t+1) = \psi_e * rand_e() * v_{ij}^{k}(t) + \psi_c * rand_c() * (p_c - x_{ij}^{k}(t)) + \psi_s * rand_s() * (p_s - x_{ij}^{k}(t)) \quad (6)$$

where, $\psi_e, \psi_c, and \psi_s$ represent the propensity constraint factors for explosive, cognitive, and social behaviors, respectively, $0 \le rand_\Theta() < 1$ where $\Theta = e, c,$ or $s,$ and $x_{ij}^{k}(t)$ represents the position of agent k at time t. $p_s = \max(\mu_{ij}^{k}(t))$ represents the global best from the neighbors, and $p_c = \max(\eta_{ij}^{k}(t))$ represents the local cognitive best. The position of each agent k at time $t+1$ can be updated by

$$x_{ij}^{k}(t+1) = x_{ij}^{k}(t) + v_{ij}^{k}(t+1). \quad (7)$$

4 Simulation Results

To evaluate the performance of the proposed stigmergy-based algorithm in a distributed swarm agent system, we implement this algorithm on a Player/Stage robot simulator. As shown in Fig. 1, the environment is an open space with 20 homogeneous mobile robots. Each robot is equipped with a camera system to detect and track targets, a laser range finder to measure the distance between the target and itself, a sonar sensor to avoid obstacles (i.e. both static obstacles and mobile obstacles, such as other agents), and a wireless communication card to communicate with other agents.

As shown in Fig.1, the searching environment is a rectangle area, where several targets with different colors and sizes are randomly distributed in the environment, and grey dots represent the robots. The arc shape in front of each robot represents the field of view of the vision system on each robot. The communication range is set up as the same range of the vision but using a circle instead of an arc. Whenever the

robots are within other agent's communication range, they would exchange the information between them.

Initially, the agents are randomly searching for targets, as shown in Fig. 1(a) at t =1. Once a robot detects a target, it would propagate the pheromone of this target to its neighbors, as shown in Fig. 1(b), where a small rectangle indicates that the vision system on the associated robot has detected the targets. After receiving a pheromone message, robots make their own decisions where to move on next time step based on the proposed algorithm, as shown in Fig. 1(c) and Fig. 1(d). Sometimes, a robot may get trapped in a corner or boundary line trying to avoid obstacles. It may take long time before it can get out, as shown in Fig. 1(e). The simulation stops when all of the targets being found and processed, as shown in Fig. 1(f).

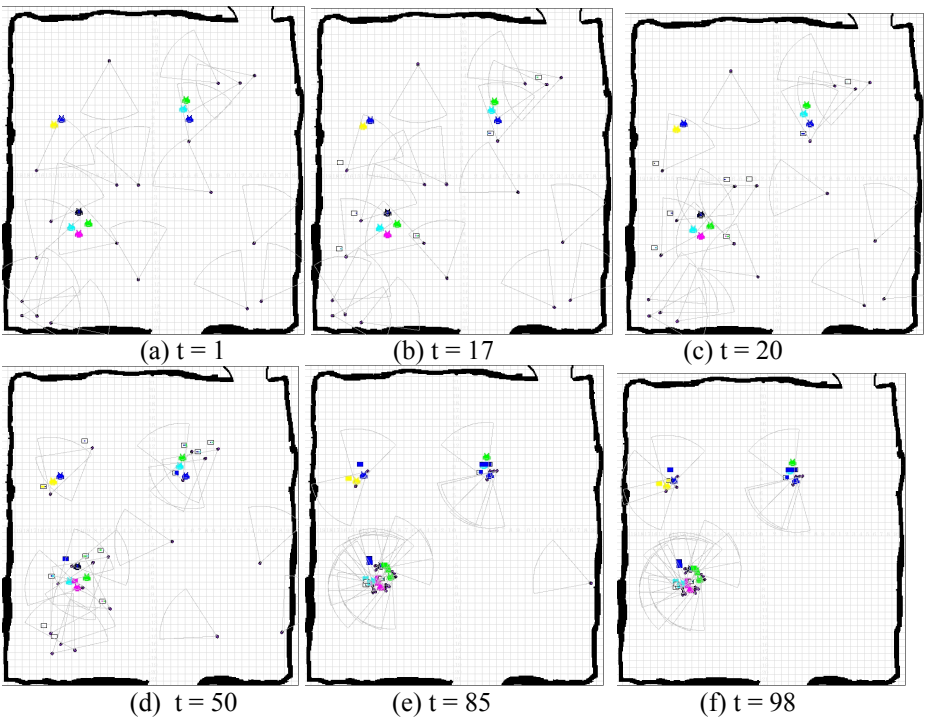


Fig.1. 20 robots searching for randomly distributed targets in an open space on a player/stage simulator at t = 1, 17. 20, 50, 65, and 82 time steps.

Generally, global path planning is very time consuming, especially for swarm robots where each agent may have to replan its global path very frequently due to the constant agent-to-agent collision. Dynamic mobile agent avoidance is another challenging task, which is not our focus in this paper. Therefore, to speed up the searching procedure in simulation, a simple path planning method is conducted. Once an agent makes its decision according to the proposed algorithms, it will set the selected target as its destination point, and move toward the target. Since there may have static obstacles and mobile obstacles (i.e. other agents) on its way to the destination, an

obstacle avoidance algorithm is necessary. Here, an adaptive vector algorithm is applied for obstacle avoidance.

To evaluate the performance of the proposed method, utility-based method is also implemented, where each agent makes its own movement decision only based on the target utility defined in (1). Three cases of different target distributions are conducted, where in each case 10 targets are distributed in the environment with fixed positions. Then, we start running the simulations with the swarm size of 20 using both methods, each method runs 35 times to obtain the mean and variance values in time steps. One time step represents the time that all of the agents need to make their movement decisions once sequentially. Some statistics results of comparison of these two methods are shown in Table 1.

Table 1. Simulation comparison of two methods.

	Mean/Variance (case 1)	Mean/variance (case 2)	Mean/Variance (case 3)
Utility-based	174/30	121/21	183/16
Hybrid	102/17	94/10	98/12

It is observed from Table 1 that the hybrid method outperforms utility-based method. The reason behind this observation is because the agents using utility-based method are extremely greedy and would always try to achieve the best utility. Therefore, they would rather move to detected targets with highest utilities than exploring areas for new targets. On the other hand, the hybrid method not only considers the target utility, but also consider the exploration (i.e. inertia factor), and its own past experiences. This exploration tendency would lead the agents using the hybrid method to be more dispersed for different targets, which may result in efficient searching results. When the agent receives the pheromone information of multiple targets, it would make decision whether to pick the target or explore to a new area, or if multiple targets are available, which one to pick so that the global optimization performance can be achieved. Furthermore, the hybrid method is more stable and consistent than the utility-based method from the variance values in Table 1. It is also observed that the proposed method is very robust to different target distributions in the environment.

5 Conclusion and Future Work

A bio-inspired stigmergy-based algorithm is proposed for a distributed multi-agent system. By using natural metaphors, inherent parallelism, stochastic nature, adaptivity, and positive feedback, the proposed method is truly distributed, self-organized, self-adaptive, and inherently scalable since there is no global control or communication, and be able to address the complex problems under dynamic environments.

However, there are still some unsolved issues remained. For example, the communication overhead among agents is extensive, which will consume too much power of limited on-board battery, especially for a large scale swarm agent system. Furthermore, it is difficult to predict the swarm performance according to a particular metric

or analyze further possible optimization margins and intrinsic limitations of these approaches from an engineering point of view. Our future work will tackle these issues and mainly focus on developing a dynamic swarm model to allow the swarm agents to achieve the target global goal and expected performance.

Reference

1. M.J.B. Krieger, J. B. Billeter, and L. Keller, "Ant-like Task Allocation and Recruitment in Cooperative Robots," *Nature*, Vol. 406, 2000, pp. 992-995.
2. M. J. Mataric, M. Nilsson, and K. T. Simsarian, "Cooperative Multi-robot Box-Pushing", *IEEE/RSJ International Conference on Intelligent Robots and Systems*, 1995.
3. A. Martinoli, A. J. Ijspeert, and F. Mondada, "Understanding Collective Aggregation Mechanisms: From Probabilistic Modeling to Experiments with Real Robots", *Robotics and Autonomous Systems*, Vol. 29, pp. 51-63, 1999.
4. T. Balch and R. C. Arkin, "Behavior-based Formation Control for Multi-robot Teams", IEEE Trans. on Robotics and Automation, 1999.
5. B. Yamauchi, "Decentralized Coordination for Multi-robot Exploration", *Robotics and Autonomous Systems*, Vol. 29, No. 1, pp. 111-118, 1999.
6. T. Weigel, J. S. Gutmann, m. Dietl, A. Kleiner, and B. Nebel, "CS Freiburg: Coordinating Robots for Successful Soccer Playing", Special Issue on Advances in Multi-Robot Systems, T. Arai, E. Pagello, and L. E. Parker, Editors, *IEEE Trans. on Robotics and Automation*, Vol. 18, No.5, pp. 685-699, 2002.
7. C.A.C. Parker and H. Zhang, "Collective Robotic Site Preparation". *Adaptive Behavior*. Vol.14, No. 1, 2006, pp. 5-19.
8. O.E. Holland and C. Melhuish, "Stigmergy, Self-Organization, and Sorting in Collective Robotics", *Artificial Life*, Vol. 5, pp. 173-202, 1999.
9. R. L. Stewart and R. A. Russell. "A Distributed Feedback Mechanism to Regulate Wall Construction by a Robotic Swarm". *Adaptive Behavior*. 14(1):21-51, 2006.
10. C. W. Reynolds, "Flocks, Herds, and Schools: A Distributed Behavioral Model", *Computer Graphics*, 21(4), July 1987, pp. 25-34.
11. C. R. Ward, F. Gobet, and G. Kendall. "Evolving collective behavior in an artificial ecology". *Artificial Life*, Vol. 7, No. 2, 2001, pp.191-209.
12. L. Spector, J. Klein, C. Perry, and M.D. Feinstein. "Emergence of collective behavior in evolving populations of flying agents". In *E. Cantu-Paz et. al. editor. Proceedings of the Genetic and Evolutionary Computation Conference (GECCO-2003)*, Berlin, Germany, 2003, pp.61-73.
13. M. Dorigo, V. Maniezzo, and A.Colorni, "Ant System: Optimization by a Colony of Cooperating Agents," *IEEE Transactions on Systems, Man, and Cybernetics-Part B: Cybernetics*, Vol. 26, No. 1, February 1996.
14. J. Kennedy and R. Eberhart, "Particle Swarm Optimization," *IEEE Conference on Neural Networks*, Proceedings 1995.
15. D. Payton, M. Daily, R. Estowski, M. Howard, and C. Lee, "Pheromone Robotics", *Autonomous Robots*, Vol. 11, No. 3, November, 2001.
16. Y. Meng, O. Kazeem, and J. Muller, "A Hybrid ACO/PSO Control Algorithm for Distributed Swarm Robots", *2007 IEEE Swarm Intelligence Symposium*, April 2007, Hawaii, USA.

To Add with Caution — Decreasing a Swarm Robotics' Efficiency by Imprudently Enhancing the Robots' Capabilities *

Yaniv Altshuler[1], Israel A. Wagner[1,2] and Alfred M. Bruckstein[1]

[1] Computer Science Department, Technion, Haifa 32000 Israel
{yanival, wagner, freddy}@cs.technion.ac.il
[2] IBM Haifa Labs, MATAM, Haifa 31905 Israel
wagner@il.ibm.com

Abstract. This work discusses the common opinion among robotics systems' designer, assuming that for a given assignment and robotics system, enhancing the robots by increasing their physical capabilities, may only result in an improvement in the overall performance of the system (albeit small). Therefore, a designer may rely on existing designs prepared in the past, and by continuously adding resources to the robots, finally achieve the overall system's performance he is interested in. As it can be shown, this assumption is wrong, as it may not only lead to a zero increase in the performance, but even to a new system, comprising far more advance (and expensive) robots, which achieve much worse results than the original system. The work presents an example concerning the problem of multi-robots exploration of a graph, in which adding communication features to the robots causes the entire system's performance to drop significantly.

1 Introduction

In recent years significant research efforts have been invested in design and simulation of multi-agent robotics and intelligent swarms systems — see e.g. [1–3] or [4–6] for biology inspired designs (behavior based control models, flocking and dispersing models and predator-prey approaches, respectively), [7–10] for economics applications and [11] for a physics inspired approach).

Tasks that have been of particular interest to researchers in recent years include synergetic mission planning [12], fault tolerance [13], swarm control [14], human design of mission plans [15], role assignment [16], multi-robot path planning [17], traffic control [18], formation generation [19], formation keeping [20], exploration and mapping [21], cleaning [22] and dynamic cleaning [23] and target tracking [24].

Hitherto, in the design of robotics systems, and specifically, in the design and implementation of multi-robotics systems, there exists an implicit yet common assumption concerning the monotonicity of the relation between the strength of the robots' capabilities (in terms of memory, sensors' accuracy, communication capabilities, etc'), and the overall performance this system achieves given a specific goal and an algorithm

* This research supported in part by the Ministry of Science Infrastructural Grant No. 3-942 and the Devorah fund.

for achieving it. In other words, is it widely assumed that given a multi-robotic system comprising robots of certain features, designed for accomplishing a specific goal, enhancing the robots' features, or alternatively, supplying those robots with additional capabilities, may only improve the performance these robots achieve when facing the same problem.

Although appealing, this approach for performance improvement as a result of tweaking existing multi-robotic designs by merely enhancing the robots' capabilities should be avoided, as such endeavors may result not only in spending expensive resources on futile attempts to increase the system's performance, but even in dramatic decrease in the overall performance of the system. Although strange at first, this phenomenon can be examined by systematically increasing some of the features of agents designed for a given task, for example — the physical exploration of a graph, while observing the changes in the performance of this group of agents.

One of the most interesting challenges for a robotics swarm system is the design and analysis of a multi-robotics system for searching and exploration (in either known or unknown areas). For example, works discussing cooperative searching tasks for static or dynamic targets can be found in [25–31] whereas examples for cooperative coverage of given regions are presented in [32–35].

This work presents a multi-agents system designed for exploring an unknown graph, by physically moving along its vertices. The problem and its model is described in Section 2. Once a system following the basic exploration algorithm was implemented and its performance measured, a change in its robots' features was made, namely — their technical specification was upgraded. The first upgrade was adding communication equipment to the robots, allowing them to share the information they acquire by traveling the graph. The second change was increasing the robots' sensors' range, in an effort to increase the accuracy of the information the robots use in order to plan their future actions, and as a result, to increase the system's efficiency. After these changes in the robots' specification were implemented, the performance of the new group was tested and analyzed. Note that the exploration algorithm itself, which was found to be achieve the best results in the original group of robots, was not changed during this process.

Surprisingly, the analyzed results of this experiment showed that not only that the upgraded group of robots did not achieve superior results compared to the original group of robots, but in fact, the exploration time required by this group was much longer compared to the exploration time of the original group of robots. This was true both for the robots with increased communication capabilities, as well as for the robots with increased sensors' range. The results and their analysis appears in Section 3.

2 Physical Graph Exploration

2.1 Physical Graphs

A *physical graph* denotes a graph $G(V, E)$ in which information regarding its vertices and edges is extracted using *I/O heads*, or *mobile agents*, instead of the "random access extraction" which is usually assumed in graph theory. These agents can physically move

between the vertices of V along the edges of E, according to a predefined, or an on-line algorithm or algorithm.

Moving along an edge e, however, require a certain *travel effort* (which might be a constant time, or alternatively, consumes a constant amount of *fuel*). Thus, the complexity of algorithms which work on physical graphs is measured by the total travel efforts required, which equals the number of edges traveled by the agents. We assume that each edge requires exactly one unit of travel effort.

Physical graphs are conveniently used in order to represent many "real world problems", in which the most efficient algorithm is not necessarily the one whose computational complexity is the minimal, but rather one in which the agents travel along the minimal number edges. Notice that while an algorithm which assumes a random access data extraction (from now on be referred to as *random access algorithm*) may read and write to the vertices of G at any order, an algorithm which assumes a physical data extraction (referred to as a *physical algorithm*) must take into account the distance between two sequential operations. The reason for this is that the use of a random access algorithm is performed using a processing unit and random access memory, whereas the use of a physical algorithm is actually done in the physical environment (or a simulated physical environment, which maintain the information access paradigm). Thus, a random access algorithm can access any vertex of the graph in $O(1)$, while a physical algorithm is confined to the distances imposed by the physical metric.

For example, for $u, v \in V$, let us assume that the distance between v and u in G is 5. Then if after a 'read' request from u, the algorithm orders a 'write' request to v, this process will take at least 5 time steps, and will consume at least 5 effort units. Furthermore, depending on the model assumed for the mobile agents knowledge base, this operation may take even longer, if, for example, the agents are not familiar with the shortest path from u to v, but rather know of a much longer path connecting the two.

2.2 Problem Description

For a given graph G, let each vertex $v \in V$ contain some small data storage unit v_s, capable of storing information saved by agents traveling through v. In time $t = 0$, let $v_s = \emptyset$ for every $v \in V$.

Let us assume that whenever a robot a goes through a vertex v, is saves at least its id number and the time of the visit in v_s.

While in vertex v, a robot a can detect the number of other robots located in v or in its immediate surroundings, and the number of edges going out from v. In addition, every edge has a unique id number, written on it (very similar to a web of roads, while each road has a unique name or a number, and that for finding out where this road leads, one must travel along it). In addition, the robot has access to all the data stored in v_s.

Given a group of k robots (or agents), capable of physically traveling the graph, according to the model described in Section 2.1, while each robot can move along a single edge per time-step, we are interested in the *goal state* G_{goal} in which $v_s \neq \emptyset$ for every $v \in V$, meaning — that every vertex was visited at least once by some robot. We are interested that the time in which G_{goal} is achieved will be minimal (namely, a short exploration as possible).

This abstract problem may ne used for simulating many common problems in the field of multi robotics, for example — a search and rescue mission of unknown number of survivors in a pre-defined (or alternatively — unknown) area, distributed autonomous mining, a de-centralized anti-virus mechanism scanning and cleaning a computer network, and so on.

2.3 Exploration Algorithm

Every robot a is equipped with a data structure a_s, capable of storing lists of vertices, edges and locations of other robots. At $t = 0$ all data structures are initialized as empty. At each time step, a robot located in vertex v follows the exploration algorithm, which controls the vertex u this robot will move to (notice that v must be a neighbor of u). Once a robot a reaches a certain vertex v, it integrates the information stored both in v_s and in a_s, so that at the end of this process, both contains the same information. Whenever an inconsistency is found regarding the status of a certain vertex, edge or robot, it is solved according to the most recent entry concerning this item.

It can be seen that throughout the movement along the graph generated by the exploration algorithm, combined with the information proliferation process executed by using the robots as a tool for transferring the information between the vertices, a more and more accurate image of G is generated in the vertices storage components, as well as in the robots' data structures. This accuracy in turn, is supposed to contribute to the efficiency of the robots, by accelerating the exploration process.

The exploration algorithm selected for this mission can generally be described as the following pseudo-code, executed by each robot independently :

1. For every v in V', when V' is the list of vertices currently known to the robot, perform the following :
   (a) Let *unvisited(v)* denote the number of edges of v, currently known to the robot, whose destination from v is currently unknown.
   (b) Let *distance(v)* denote the length of the shortest path between v and the current location of the robot, comprising only vertices and edges currently known to the robot.
   (c) Let *robots(v)* denote the probability that other robots are located at v. This is calculated based on the knowledge the robot has of the structure of G in the vicinity of v and of the knowledge the robot has concerning the whereabouts of the other robots.
   (d) Let *robots-neighborhood(v)* denote the probability that other robots are located at the close vicinity of v. This is calculated similarly to *robots(v)*.
   (e) Calculate the combined score of v, as a weighted average of *unvisited(v)*,*distance(v)*, *robots(v)*,*robots-neighborhood(v)*. Note that the selection of the averaging vector is an extremely important feature of the exploration algorithm.
2. Let v_{best} be the vertex whose combined score is the highest.
3. Start walking towards v_{best} (at the pace of a single edge per time-step).
4. When reaching v_{best}, randomly select one of the edges going out from v_{best} with an unknown destination, and move towards it.

For choosing the best averaging vector, many simulation were executed, testing a variety of weights values. Finally, several vectors were found, which were both robust (in terms of a relatively high score for the scenarios in which they function at their worst) and potent (in terms of the ability to score extremely high in scenarios in which they were at the best). A detailed discussion concerning the specific vectors and the process of selecting them will appear in an extended version of this work, currently under preparation.

2.4 Upgrades

Once the performance of a group of k robots implementing the exploration algorithm with the chosen averaging vectors were available, the robots' technical specification was enhanced by two major aspects.

First, a component simulating a full-range broadcasting equipment was added to each robots, allowing it to instantly update and receive information from the other robots of the group. The result of this upgrade if essentially the ability of a robot which calculates the heuristic score of the vertices of the graph, trying to decide its destination, to use the most accurate information, as it is known to *any of the robots*. This upgrade was expected to boost the performance of the robots, since often, a robot becomes isolated in the graph, traveling among previously visited vertices, while valuable information concerning this area of the graph was already gathered by the rest of the robots, and is unavailable for this robot.

The second upgrade was the addition of a full-range sensor, capable of scanning the entire graph G. Notice that this component transform each robot to an omniscient unit, making both communication equipment and data storage components along the vertices unnecessary (as at any given time, each robot can access any information it requires, with complete accuracy). This upgrade was expected to increase even further the robots' efficiency, and as a result — to decrease their exploration time.

3 Results

A simulation of the three types of robots was built. The exploration algorithm was tested on *Erdös-Renyi* random graphs $G \sim G(n,p)$ where G has n vertices, and each pair of vertices form an edge in G with probability p independently of each other.

Surprisingly, once examining the exploration times of the upgraded robots, and comparing them to those of the original groups of robots, the exploration times of the original groups were significantly lower than those of the upgraded robots. An example of this phenomenon appears in Figure 1.

It can easily be seen that although there is almost no difference between the performance of the broadcasting robots and the omniscient robots, both had much longer exploration times than the the original group of the "simple robots", which lacked either communication or extreme sensing capabilities. It is interesting to mention that this phenomenon became increasingly more intense as the graphs became more and more dense, that is — as p, the edge probability, was increased. Furthermore, as the group

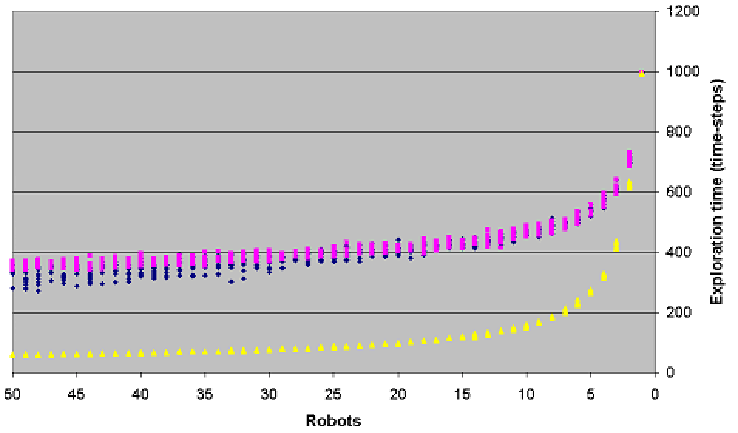


Fig. 1. This chart depicts the range of exploration times of three groups of robots, tested in a variety of random graphs. The lower yellow curve represents the exploration time of the original group, comprising "basic robots", to whom the exploration algorithm used was originally designed. The blue and purple curves represent the exploration times of the two groups of "upgraded robots", whose communication and sensing capabilities were enhanced, respectively.

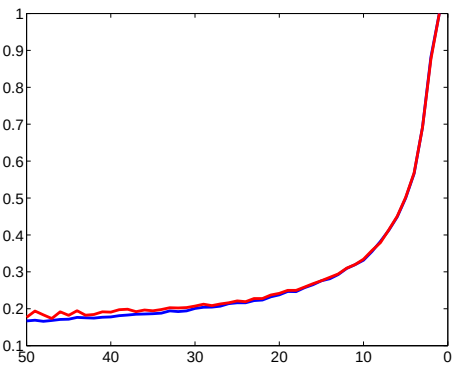


Fig. 2. The graph represents the ratio between exploration times of the "basic robots" and averaged results of the two groups of "upgraded robots" (the red curve represents the robots which were assigned a full-range broadcasting capability, while the blue curve represents the robots whose sensors' range was increased). As the number of robots (represented as the X axes) increases, the ratios discussed decreases. For groups of over 30 robots, the upgraded robots achieve an efficiency of approximately 20% than this of the simple robots (namely, 5 times larger an exploration time).

of robots became larger, the inefficiency of the upgraded robots became significantly clearer, as can be seen in Figure 2

After analyzing the reasons for these unexpected results, by reconstructing the internal decisions' considerations made by each robot in the various scenarios, it was discovered that the improved accuracy of the robots caused an undesired synchronicity effect, grouping the robots into a small and tightly packed group. As a result, the robots were not able to efficiently explore many parts of the graphs, as they moved heavily, delaying each other from exposing unrevealed valuable information (such as shorter paths between vertices).

As it turned out, the reason for this phenomenon was that the averaging vector, found to be best for the original group of robots, contained a positive weight for the *robots(v)* element. The positive contribution of this element to the overall score of some vertex v intended to assist the scattered robots to remain loosely tied, in order to sustain the proliferation of valuable information. As the accuracy of the robots' knowledge increased (first by providing them a accurate information concerning the other robots' whereabouts at any given time, and later by providing them even the shortest ways of reaching each other), the robots no longer needed such a strong attraction factor in their decision making process. However, as the robots utilizes the same exploration algorithm as originally was used by the simple robots, this attractor stopped being an assisting element, but rather generated the delaying effect described above.

After further investigating this phenomenon, as assumption was made, that by slightly changing the exploration algorithm, the upgraded robots will easily be able to achieve superior performance, as originally expected. For example, by simulating noise when it comes to the locations of the other robots, by deciding randomly whether to take the mentioned attracting factor into consideration, or by merely changing the averaging vector, decreasing the effect of the *robots(v)* component on the overall score of a vertex. However, while the first two methods require the robots to be enhanced once again (as a random generator was not currently included in the robots' specification), the last cannot easily be analytically shown to improve the performance. Nevertheless, it is very easy to show that there exist some alternative exploration algorithm which will enable the upgraded robots to produce far faster exploration than the simple robots (for example, having a complete knowledge of the graph, each robot can calculate locally the fastest way in which the entire group can scan the graph, and then simply act its role in this plan). However, as this was already known prior to this experiment, it does not contradict the experiment's result, namely — that enhancing the capabilities of robots which act according to an algorithm who did not take into consideration this enhancement, may result in an overall decrease of the system's performance.

4 Conclusions

This work discussed a multi-robotic system designed for the task of physically exploring an unknown graph. The problem and the solution model were presented, as well as the initial results of a selected exploration algorithm. Then, two changes in the robots' technical specification, intended to increase the robots' efficiency and performance were presented, and the results obtained by a group comprising the new robots

were presented and analyzed. These results hinted that counterintuitively, increasing the robots' physical capabilities caused a decrease in the system's overall performance, due to the appearance of a strong synchronicity between the robots. An estimation was made concerning a possible solution to this problem, which in turn would have required both changing the robots' exploration algorithm and possibly, enhancing even more the robots' specification. An observation concerning the results of this experiment was made, stating that when "improving" existing robots, one should take extra care to verify that this improvement does not result in such malicious impacts on the entire robots group. In conclusion, it is important to state that the results of the experiment discussed in this work do not intend to speak against the enhancements of existing robots', or multi-robotic systems' capabilities per-se, but rather — to remind designers of such systems that although innocent, any change in original designs should be done with care and systematic examination (both theoretical and empirical) of the possible results of such a change.

References

1. S. Hettiarachchi, W. Spears: "Moving swarm formations through obstacle fields", in International Conference on Artificial Intelligence, (2005).
2. I.A. Wagner, A.M. Bruckstein: "From Ants to A(ge)nts: A Special Issue on Ant—Robotics", Annals of Mathematics and Artificial Intelligence, Special Issue on Ant Robotics, Kluer Academic Publishers, vol. 31, Nos. 1–4, pp. 1–6, (2001)
3. L.Steels: "Cooperation Between Distributed Agents Through Self-Organization", Decentralized A.I - Proc. first European Workshop on Modeling Autonomous Agents in Multi-Agents world, Y.DeMazeau, J.P.Muller (Eds.), pp. 175–196, Elsevier, (1990)
4. R.C.Arkin: "Integrating Behavioral, Perceptual, and World Knowledge in Reactive Navigation", Robotics and Autonomous Systems, 6:pp.105-122, (1990).
5. M.J.Mataric: "Designing Emergent Behaviors: From Local Interactions to Collective Intelligence", In J.Meyer, H.Roitblat, and S.Wilson, editors, Proceedings of the Second International Conference on Simulation of Adaptive Behavior, pp.432-441, Honolulu, Hawaii, MIT Press, (1992).
6. T.Haynes, S.Sen: "Evolving Behavioral Strategies in Predators and Prey", In Gerard Weiss and Sandip Sen, editors, Adaptation and Learning in Multi-Agent Systems, pp.113-126. Springer, (1986).
7. B.P.Gerkey, M.J.Mataric: "Sold! Market Methods for Multi-Robot Control", IEEE Transactions on Robotics and Automation, Special Issue on Multi-robot Systems, (2002).
8. G.Rabideau, T.Estlin, T.Chien, A.Barrett: "A Comparison of Coordinated Planning Methods for Cooperating Rovers", Proceedings of the American Institute of Aeronautics and Astronautics (AIAA) Space Technology Conference, (1999).
9. S.M.Thayer, M.B.Dias, B.L.Digney, A.Stentz, B.Nabbe, M.Hebert: "Distributed Robotic Mapping of Extreme Environments", Proceedings of SPIE, Vol. 4195, Mobile Robots XV and Telemanipulator and Telepresence Technologies VII, (2000).
10. M.P.Wellman, P.R.Wurman: "Market-Aware Agents for a Multiagent World", Robotics and Autonomous Systems, Vol. 24, pp.115–125, (1998).
11. D.Chevallier, S.Payandeh: "On Kinematic Geometry of Multi-Agent Manipulating System Based on the Contact Force Information", The 6^{th} International Conference on Intelligent Autonomous Systems (IAS-6), pp.188–195, (2000).

12. R.Alami, S.Fleury, M.Herrb, F.Ingrand, F.Robert: “Multi-Robot Cooperation in the Martha Project”, IEEE Robotics and Automation Magazine, (1997).
13. L.E.Parker: “ALLIANCE: An Architecture for Fault-Tolerant Multi-Robot Cooperation”, IEEE Transactions on Robotics and Automation, 14(2), pp. 220-240, (1998).
14. M.J.Mataric: “Interaction and Intelligent Behavior”, PhD Thesis, Massachusetts Institute of Technology, (1994).
15. D.MacKenzie, R.Arkin, J.Cameron: “Multiagent Mission Specification and Execution”, Autonomous Robots, 4(1), pp. 29-52, (1997).
16. C.Candea, H.Hu, L.Iocchi, D.Nardi, M.Piaggio: “Coordinating in Multi-Agent RoboCup Teams”, Robotics and Autonomous Systems, 36(2- 3):67-86, August (2001).
17. A.Yamashita, M.Fukuchi, J.Ota, T.Arai, H.Asama: “Motion Planning for Cooperative Transportation of a Large Object by Multiple Mobile Robots in a 3D Environment”, In Proceedings of IEEE International Conference on Robotics and Automation, pp. 3144-3151, (2000).
18. S.Premvuti, S.Yuta: “Consideration on the Cooperation of Multiple Autonomous Mobile Robots”, In Proceedings of the IEEE International Workshop of Intelligent Robots and Systems, pp. 59-63, Tsuchiura, Japan, (1990).
19. N.Gordon, I.A.Wagner, A.M.Bruckstein: “Discrete Bee Dance Algorithms for Pattern Formation on a Grid”, In the proceedings of IEEE International Conference on Intelligent Agent Technology (IAT03), pp. 545–549, October, (2003).
20. T.Balch, R.Arkin: “Behavior-Based Formation Control for Multi-Robot Teams”, IEEE Transactions on Robotics and Automation, December (1998).
21. I.M. Rekleitis, G. Dudek, E. Milios: “Experiments in Free-Space Triangulation Using Cooperative Localization”, IEEE/RSJ/GI International Conference on Intelligent Robots and Systems (IROS), (2003).
22. I.A. Wagner, A.M. Bruckstein: “Cooperative Cleaners: A Case of Distributed Ant-Robotics”, “Communications, Computation, Control, and Signal Processing: A Tribute to Thomas Kailath”, pp. 289–308, Kluwer Academic Publishers, The Netherlands, (1997)
23. Altshuler, Y., Bruckstein, A.M., Wagner, I.A.: “Swarm Robotics for a Dynamic Cleaning Problem”, in “IEEE Swarm Intelligence Symposium 2005”, pp. 209–216, (2005).
24. Shucker, B., Bennett, J.K.: “Target tracking with distributed robotic macrosensors”, Military Communications Conference 2005 (MILCOM 2005), vol. 4, pp. 2617–2623, (2005).
25. Y.Altshuler, V. Yanovsky, I.A.Wagner, A.M. Bruckstein: “The Cooperative Hunters - Efficient Cooperative Search For Smart Targets Using UAV Swarms”, Second International Conference on Informatics in Control, Automation and Robotics (ICINCO), the First International Workshop on Multi-Agent Robotic Systems (MARS), pp. 165–170, Barcelona, Spain, (2005).
26. Kerr, W., Spears, D.: “Robotic simulation of gases for a surveillance task”, Intelligent Robots and Systems 2005 (IROS 2005), pp. 2905–2910, (2005).
27. Passino, K., Polycarpou, M., Jacques, D., Pachter, M., Liu, Y., Yang, Y., Flint, M. and Baum, M.: “Cooperative Control for Autonomous Air Vehicles”, In Cooperative Control and Optimization, R. Murphey and P. Pardalos, editors. Kluwer Academic Publishers, Boston, (2002).
28. Polycarpou, M., Yang, Y. and Passino, K.: “A Cooperative Search Framework for Distributed Agents”, In Proceedings of the 2001 IEEE International Symposium on Intelligent Control (Mexico City, Mexico, September 5–7). IEEE, New Jersey, pp. 1–6, (2001).
29. Stone, L.D: “Theory of Optimal Search”, Academic Press, New York, (1975).
30. Koopman, B.O: “The Theory of Search II, Target Detection”, Operations Research 4, 5, 503–531, October, (1956).
31. Vincent, P., Rubin, I.: “A Framework and Analysis for Cooperative Search Using UAV Swarms”, ACM Simposium on applied computing, (2004).

32. Rekleitisy, I., Lee-Shuey, V., Peng Newz, A., Chosety, H.: “Limited Communication, Multi-Robot Team Based Coverage”, Proceedings of the 2004 IEEE International Conference on Robotics and Automation, New Orleans, LA, April, (2004).
33. Koenig, S., Liu, Y.: “Terrain Coverage with Ant Robots: A Simulation Study”, AGENTS’01, May 28–June 1, Montreal, Quebec, Canada, (2001).
34. I.Rekleitis, A.P New, H.Choset: “Distributed coverage of unknown/unstructured environments by mobile sensor networks”, the Third MRS workshop, (2005).
35. C.S.Kong, N.A.Peng, I.Rekleitis: “Distributed Coverage with Multi-Robot System”, Proceedings of the 2006 IEEE International Conference on Robotics and Automation Orlando, Florida - May (2006).

Cooperative Collision Avoidance between Multiple Robots based on Bernstein-Bézier Curves

Igor Škrjanc and Gregor Klančar

Faculty of Electrical Engineering, University of Ljubljana
Tržaška 25, SI-1000 Ljubljana, Slovenia
igor.skrjanc@fe.uni-lj.si

Abstract. In this paper a new cooperative collision-avoidance method for multiple nonholonomic robots based on Bernstein- Bézier curves is presented. The reference path of each robot from the start pose to the goal pose, is obtained by minimizing the penalty function, which takes into account the sum of all the paths subjected to the distances between the robots, which should be bigger than the minimal distance defined as the safety distance. When the reference paths are defined the model predictive trajectory tracking is used to define the control. A prediction model derived from linearized tracking-error dynamics is used to predict future system behavior. A control law is derived from a quadratic cost function consisting of the system tracking error and the control effort. The results of the simulation and some future work ideas are discussed.

1 Introduction

Collision avoidance is one of the main issues in applications for a wide variety of tasks in industry, human-supported activities, and elsewhere. Often, the required tasks cannot be carried out by a single robot, and in such a case multiple robots are used cooperatively. The use of multiple robots may lead to a collision if they are not properly navigated. Collision-avoidance techniques tend to be based on speed adaptation, route deviation by one vehicle only, route deviation by both vehicles, or a combined speed and route adjustment. When searching for the best solution that will prevent a collision many different criteria are considered: time delay, total travel time, planned arrival time, etc. Our optimality criterion will be the minimal travel time, which directly implies a minimal total length of the robot paths, subject to a minimal safety distance between all the robots.

In the literature many different techniques for collision avoidance have been proposed. The first approaches proposed avoidance, when a collision between robots is predicted, by stopping the robots for a fixed period or by changing their directions. The combination of these techniques is proposed in [1]. The behavior-based motion planning of multiple mobile robots in a narrow passage is presented in [2]. Intelligent learning techniques were incorporated into neural and fuzzy control for mobile-robot navigation to avoid a collision as proposed in [3].

In our paper the control of multiple mobile robots to avoid collisions in a two-dimensional free-space environment is separated into the path planning for each individual robot to reach its goal pose as fast as possible. The second part of the task is to design the control that will ensure the perfect trajectory tracking of the mobile robots.

Several controllers were proposed for mobile robots with nonholonomic constraints. An extensive review of nonholonomic control problems can be found in [4]. In trajectory-tracking control a reference trajectory is usually obtained by using a reference robot; therefore, all kinematics constraints are implicitly considered by a reference trajectory. From the reference trajectory a feed-forward system of inputs combined with a feedback control law are mostly used [5]. Lyapunov stable time-varying state-tracking control laws were pioneered by [9]. The stabilization to the reference trajectory requires a nonzero motion condition. Many variations and improvements to this state-tracking controller followed in subsequent research [10]. A tracking controller obtained with input-output linearization is used in [5], a saturation feedback controller is proposed in [11] and a dynamic feedback linearization technique is used in [6].

The paper is organized as follows. In Section 2 the problem is stated. The concept of path planning is shown in Section 3. The idea of optimal collision avoidance for multiple mobile robots based on Bézier curves is discussed in Section 4. The trajectory-tracking controller design where the control strategy consists of feed-forward and feedback actions is introduced in Section 5. In Section 5.1 the proposed model predictive controller is derived. The simulation results of the obtained collision-avoidance control are presented in Section 6 and the conclusion is given in Section 7.

2 Statement of the Problem

The collision-avoidance control problem of multiple nonholonomic mobile robots is proposed in a two-dimensional free-space environment. The simulations are performed for a small two-wheel differentially driven mobile robot of dimension $7.5 \times 7.5 \times 7.5$ cm. The architecture of our robots has a nonintegrable constraint in the form $\dot{x}\sin\theta - \dot{y}\cos\theta = 0$ resulting from the assumption that the robot cannot slip in a lateral direction where $q(t) = [x(t)\ y(t)\ \theta(t)]^T$ are the generalized coordinates The kinematics model of the mobile robot is

$$\dot{q}(t) = \begin{bmatrix} \cos\theta(t) & 0 \\ \sin\theta(t) & 0 \\ 0 & 1 \end{bmatrix} \begin{bmatrix} v(t) \\ \omega(t) \end{bmatrix} \tag{1}$$

where $v(t)$ and $\omega(t)$ are the tangential and angular velocities of the platform. During low-level control the robot's velocities and accelerations are bounded within the maximal allowed velocities and accelerations, which prevents the robot from slipping.

The danger of a collision between multiple robots is avoided by determining the strategy of the robots' navigation, where we define the reference path to fulfil certain criteria. The reference path of each robot from the start pose to the goal pose is obtained by minimizing the penalty function, which takes into account the sum of all the paths subjected to the distances between the robots, which should be larger than the defined safety distance. When the reference paths are defined the model predictive trajectory tracking is used to define the control.

3 Path Planning based on Bernstein-Bézier Curves

Given a set of control points $P_0, P_1, \ldots, P_b$, the corresponding Bernstein-Bézier curve (or Bézier curve) is given by

$$\mathbf{r}(\lambda) = \sum_{i=0}^{b} B_{i,b}(\lambda)\mathbf{p}_i$$

where $B_{i,b}(\lambda)$ is a Bernstein polynomial, λ is a normalized time variable ($\lambda = t/T_{max}$, $0 \leq \lambda \leq 1$) and $\mathbf{p}_i,\ 0 = 1, \ldots, b$ stands for the local vectors of the control point P_i ($\mathbf{p}_i = P_{i_x}\mathbf{e}_x + P_{i_y}\mathbf{e}_y$, where $P_i = \left(P_{i_x}, P_{i_y}\right)$ is the control point with coordinates P_{i_x} and P_{i_y}, and $\mathbf{e}_x$ and $\mathbf{e}_y$ are the corresponding base unity vectors). The Bernstein-Bézier polynomials, which are the base functions in the Bézier-curve expansion, are given as follows:

$$B_{i,b}(\lambda) = \binom{b}{i} \lambda^i (1-\lambda)^{b-i},\ i = 0, 1, \ldots, b$$

which have the following properties: $0 \leq B_{i,b}(\lambda) \leq 1,\ 0 \leq (\lambda) \leq 1$ and $\sum_{i=0}^{b} B_{i,b} = 1$.

The Bézier curve always passes through the first and last control point and lies within the convex hull of the control points. The curve is tangent to the vector of the difference $\mathbf{p}_1 - \mathbf{p}_0$ at the start point and to the vector of the difference $\mathbf{p}_b - \mathbf{p}_{b-1}$ at the goal point. A desirable property of these curves is that the curve can be translated and rotated by performing these operations on the control points. The undesirable properties of Bézier curves are their numerical instability for large numbers of control points, and the fact that moving a single control point changes the global shape of the curve. The former is sometimes avoided by smoothly patching together low-order Bézier curves.

The properties of Bézier curves are used in path planning for nonholonomic mobile robots. In particular, the fact of the tangentiality at the start and at the goal points and the fact that moving a single control point changes the global shape of the curve. Let us assume the starting pose of the mobile robot is defined in the generalized coordinates as $\mathbf{q}_s = [x_s, y_s, \theta_s]^T$ and the goal pose is defined as $\mathbf{q}_g = [x_g, y_g, \theta_g]^T$, which means that the robot starts in position $P_s(x_s, y_s)$ with orientation θ_s and has a goal defined with position $P_g(x_g, y_g)$ with orientation θ_g. The property of tangentiality requires the definition of the neighboring points $P_1(x_1, y_1)$ and $P_2(x_2, y_2)$, which become

$$P_1(x_s + d\cos\theta_s, y_s + d\sin\theta_s),\ P_2(x_g + d\cos(\theta_g + \pi), y_g + d\sin(\theta_g + \pi)) \quad (2)$$

where d stands for the distance between P_s and P_1 and between P_g and P_2. The distance d is usually defined relatively to the distance between the start and the goal point D ($D = \mid \mathbf{p}_g - \mathbf{p}_s \mid$) defined as $d = \gamma D,\ 0 < \gamma < 0.5$. These four control points P_s, P_1, P_2 and P_g uniformly define the third order Bézier curve. The need for flexibility of the global shape and the fact that moving a single control point changes the global shape of the curve imply the introduction of another point, which will be denoted as $P_o(x_o, y_o)$. By changing the position of point P_o the global shape of the curve changes. This means that having in mind the flexibility of the global shape of the curve and the start and the goal pose of the mobile robot, the path can be planned by four fixed points

and one variable point. The Bézier curve is now defined as a sequence of points P_s, P_1, P_o, P_2 and P_g in Fig 1. This means that we are dealing with Bernstein polynomials of the fourth order ($B_{i,b},\ i = 0, \ldots, b,\ b = 4$). The curve is defined as follows:

$$\mathbf{r}(\lambda) = B_{0,4}\mathbf{p}_s + B_{1,4}\mathbf{p}_1 + B_{2,4}\mathbf{p}_o + B_{3,4}\mathbf{p}_2 + B_{4,4}\mathbf{p}_g \tag{3}$$

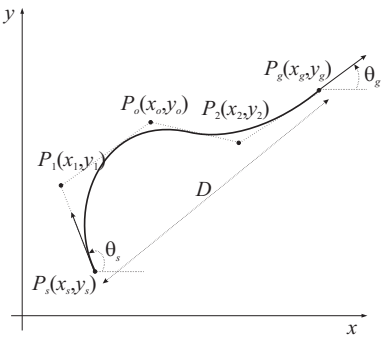


Fig. 1. The Bézier curve.

4 Optimal Collision Avoidance based on Bernstein-Bézier Curves

In this subsection a detailed presentation of cooperative multiple robots collision avoidance based on Bézier curves will be given. Let as assume the number of robots equals n. The i-th robot is denoted as R_i and has the start position defined as $P_{si}\,(x_{si}, y_{si})$ and the goal position defined as $P_{gi}\,(x_{gi}, y_{gi})$. The reference path of i-th robot will be denoted with the Bézier curve $\mathbf{r}_i(\lambda) = [x_i(\lambda), y_i(\lambda)]^T$. By choosing maximal time of the experiment T_{max} ($t = T_{max}\lambda$, $0 \leq \lambda \leq 1$) the robots tangential velocity profiles are determined. T_{max} is determined by the fastest robot R_i as $T_{max} = \frac{\max_i(v_i(\lambda))}{v_{max}}$, $i = 1, \ldots, n,\ 0 \leq \lambda \leq 1$, where v_{max} is maximal allowed tangential robot velocity. Maximal time T_{max} is then common to all robots R_i. In Fig. 2 a collision avoidance for $n = 2$ is presented for reasons of simplicity.

The safety margin to avoid a collision between two robots is, in this case, defined as the minimal necessary distance between these two robots. The distance between the robot R_i and R_j is $r_{ij}(\lambda) = \mid \mathbf{r}_i(\lambda) - \mathbf{r}_j(\lambda) \mid,\ i = 1, \ldots, n,\ j = 1, \ldots, n,\ i \neq j$.

Defining the minimal necessary safety distance as d_s, the following condition for collision avoidance is obtained $r_{ij} \geq d_s,\ 0 \leq \lambda \leq 1,\ i, j$. Fulfilling this criteria means that the robots will never meet in the same region defined by a circle with radius d_s, which is called a non-overlapping criterion. At the same time we would like to minimize the length of the path for each robot, which is defined as s_i. The length $s_i(\lambda)$ is defined as $s_i(\lambda) = \int_0^\lambda v_i(\lambda)d\lambda$, where $v_i(\lambda)$ stands for the tangential velocity in the normalized

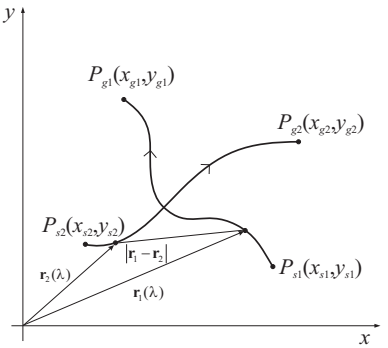


Fig. 2. Collision avoidance based on Bernstein-Bézier.

variable λ and the length of the path of the robot R_i from the start control point to the goal point is now calculated as:

$$v_i(\lambda) = \mid \dot{\mathbf{r}}(\lambda) \mid = \left(\dot{x}_i^2(\lambda) + \dot{y}_i^2(\lambda)\right)^{\frac{1}{2}},\ s_i = \int_0^1 \left((\dot{x}_i^2(\lambda)) + \dot{y}_i^2(\lambda)\right)^{\frac{1}{2}} d\lambda$$

where $\dot{x}_i(\lambda)$ stands for $\frac{dx_i(\lambda)}{d\lambda}$ and $\dot{y}_i(\lambda)$ for $\frac{dy_i(\lambda)}{d\lambda}$. Assuming that the start and goal control points are known, the global shape and length of each path can be optimized by changing the flexible control point P_{oi}. The collision-avoidance problem is now defined as an optimization problem as follows:

$$minimize \sum_{i=1}^{n} s_i \ subject\ to\ d_s - r_{ij}(\lambda) \leq 0,\ \forall i, j,\ i \neq j,\ \ 0 \leq \lambda \leq 1 \qquad (4)$$

The minimization problem is called an *inequality optimization* problem and can be introduced as the minimization of the following penalty function

$$F(\mathbf{P}_o) = \sum_i s_i + c \sum_{ij} \Gamma_{ij},\ \ \Gamma_{ij} = \begin{cases} 1, & \min r_{ij}(\lambda) < d_s \\ 0, & \min r_{ij}(\lambda) > d_s \end{cases},\ \ i, j,\ i \neq j,\ \ 0 \leq \lambda \leq 1 \quad (5)$$

where c stands for a large scalar to penalize the unfulfillment of constraints. The solution of the minimization problem $\min_{\mathbf{P}_o} F$ is a set of n control points $\mathbf{P}_o = \{P_{o1}, \ldots, P_{on}\}$. Each optimal control point $P_{oi},\ i = 1, \ldots, n$ uniformly defines one optimal path, which ensures collision avoidance in the sense of a safety distance and will be used as a reference trajectory of the ith robot and will be denoted as $\mathbf{r}_{ri}(\lambda)$.

To define the feasible reference path that will be collision safe, the real time should be introduced. In the real system the tangential and the angular velocities are limited to ($v_{max},\ \omega_{max}$). Using the relation $v(t) = \frac{v(\lambda)}{T_{max}}$ the maximal time T_{max} can be defined to fulfil the velocity limitation ($T_{max} \geq \frac{\max v(\lambda)}{v_{max}}$).

5 Path Tracking

The previously obtained optimal collision-avoidance path for the ith robot is defined as $\mathbf{r}_{ri}(t) = [x_{ri}(t), y_{ri}(t)]^T,\ i = 1, \ldots, n$. In this section the development of a predictive

path-tracking controller will be presented. The path-tracking control is realized as a sum of the feed-forward and feed-back controls. The feed-forward control for the ith robot is calculated from a feasible reference path $\mathbf{r}_{ri}(t) = [x_{ri}(t), y_{ri}(t)]^T$, which enables us to reach a desired pose. The feed-forward control inputs $v_{ri}(t)$ and $\omega_{ri}(t)$ are derived using a kinematic model (1). The tangential velocity $v_{ri}(t)$ and the tangent angle of each point on the path are calculated as follows

$$v_{ri}(t) = \left(\dot{x}_{ri}^2(t) + \dot{y}_{ri}^2(t)\right)^{\frac{1}{2}}, \quad \omega_{ri}(t) = \frac{\dot{x}_{ri}(t)\ddot{y}_{ri}(t) - \dot{y}_{ri}(t)\ddot{x}_{ri}(t)}{\dot{x}_{ri}^2(t) + \dot{y}_{ri}^2(t)} = v_{ri}(t)\kappa(t) \tag{6}$$

where $\kappa(t)$ is the path curvature. The necessary condition in the path-design procedure is a twice-differentiable path and a nonzero tangential velocity $v_{ri}(t) \neq 0$.

If for some time t the tangential velocity is $v_{ri}(t)$=0, the robot rotates at a fixed point with the angular velocity $\omega_{ri}(t)$ calculated from an explicitly given $\theta_{ri}(t)$.

The feedback control law is derived from a linear time-varying system obtained by an approximate linearization around the trajectory. The obtained linearization is shown to be controllable as long as the trajectory does not come to a stop, which implies that the system can be asymptotically stabilized by smooth time-varying linear or nonlinear feedback. The tracking error $\mathbf{e}(t) = [e_1(t)\, e_2(t)\, e_3(t)]^T$ of a mobile robot expressed in the frame of the real robot reads

$$\mathbf{e} = \begin{bmatrix} \cos\theta & \sin\theta & 0 \\ -\sin\theta & \cos\theta & 0 \\ 0 & 0 & 1 \end{bmatrix} (\mathbf{q}_{ri} - \mathbf{q}) \,. \tag{7}$$

Considering the robot kinematics (1) and derivating relations (7) the following kinematics model is obtained

$$\dot{\mathbf{e}} = \begin{bmatrix} \cos e_3 & 0 \\ \sin e_3 & 0 \\ 0 & 1 \end{bmatrix} \begin{bmatrix} v_{ri} \\ \omega_{ri} \end{bmatrix} + \begin{bmatrix} -1 & e_2 \\ 0 & -e_1 \\ 0 & -1 \end{bmatrix} \mathbf{u} \tag{8}$$

where $\mathbf{u} = [v\, \omega]^T$ is the velocity input vector and v_{ri} and ω_{ri} are already defined in (6). The robot input vector $\mathbf{u}$ is further defined as the sum of the feed-forward and feedback control actions ($\mathbf{u} = \mathbf{u}_F + \mathbf{u}_B$) where the feed-forward input vector, $\mathbf{u}_F$, is obtained by a nonlinear transformation of the reference inputs $\mathbf{u}_F = [v_{ri} \cos e_3 \;\; \omega_{ri}]^T$ and the feedback input vector, is $\mathbf{u}_B = [u_{B_1}\, u_{B_2}]^T$, which is the output of the controller defined in section 5.1.

Using the relation $\mathbf{u} = \mathbf{u}_F + \mathbf{u}_B$, rewriting (8) and furthermore, by linearizing the error dynamics around the reference trajectory ($e_1 = e_2 = e_3 = 0$, $u_{B_1} = u_{B_2} = 0$) the following linear model is obtained

$$\dot{\mathbf{e}} = \begin{bmatrix} 0 & \omega_{ri} & 0 \\ -\omega_{ri} & 0 & v_{ri} \\ 0 & 0 & 0 \end{bmatrix} \mathbf{e} + \begin{bmatrix} -1 & 0 \\ 0 & 0 \\ 0 & -1 \end{bmatrix} \mathbf{u}_B \tag{9}$$

which in the state-space form is $\dot{\mathbf{e}} = \mathbf{A}_c\mathbf{e} + \mathbf{B}_c u_B$. According to Brockett's condition [12] a smooth stabilization of the system (1) or its linearization is only possible with time-varying feedback. In the following the obtained linear model is used in the derived predictive control law.

5.1 Model Predictive Control based on a Robot Tracking-error Model

To design the controller for trajectory tracking the system (9) will be written in discrete-time form as

$$\mathbf{e}(k+1) = \mathbf{A}\mathbf{e}(k) + \mathbf{B}\mathbf{u}_B(k)$$

where $\mathbf{A} \in \mathbb{R}^n \times \mathbb{R}^n$, n is the number of state variables and $\mathbf{B} \in \mathbb{R}^n \times \mathbb{R}^m$, m is the number of input variables. The discrete matrix $\mathbf{A}$ and $\mathbf{B}$ can obtained as follows:$\mathbf{A} = \mathbf{I} + \mathbf{A}_c T_s, \quad \mathbf{B} = \mathbf{B}_c T_s$ which is a good approximation during a short sampling time T_s.

The idea of the moving-horizon control concept is to find the control-variable values that minimize the receding-horizon quadratic cost function (in a certain interval denoted with h) based on the predicted robot-following error:

$$J(u_B, k) = \sum_{i=1}^{h} \epsilon^T(k,i)\mathbf{Q}\epsilon(k,i) + \mathbf{u}_B^T(k,i)\mathbf{R}\mathbf{u}_B(k,i) \tag{10}$$

where $\epsilon(k,i) = \mathbf{e}_{ri}(k+i) - \mathbf{e}(k+i|k)$ and $\mathbf{e}_{ri}(k+i)$ and $\mathbf{e}(k+i|k)$ stands for the reference robot following-trajectory and the robot-following error, respectively, and $\mathbf{Q}$ and $\mathbf{R}$ stand for the weighting matrices where $\mathbf{Q} \in \mathbb{R}^n \times \mathbb{R}^n$ and $\mathbf{R} \in \mathbb{R}^m \times \mathbb{R}^m$, with $\mathbf{Q} \geq 0$ and $\mathbf{R} \geq 0$.

Output prediction in the discrete-time framework In the moving time frame the model output prediction at the time instant h can be written as:

$$\begin{aligned}\mathbf{e}(k+h|k) = \mathit{\Pi}_{j=1}^{h-1}\mathbf{A}(k+j|k)\mathbf{e}(k) + \textstyle\sum_{i=1}^{h}\left(\mathit{\Pi}_{j=i}^{h-1}\mathbf{A}(k+j|k)\right)\mathbf{B}(k+i-1|k)\cdot \\ \cdot\mathbf{u}_B(k+i-1) + +\mathbf{B}(k+h-1|k)\mathbf{u}_B(k+h-1)\,.\end{aligned} \tag{11}$$

Defining the robot-tracking prediction-error vector

$$\mathbf{E}^*(k) = \left[e(k+1|k)^T\ e(k+2|k)^T \dots e(k+h|k)^T\right]^T$$

where $\mathbf{E}^* \in \mathbb{R}^{n \cdot h}$ for the whole interval of observation (h) and the control vector

$$\mathbf{U}_B(k) = \left[\mathbf{u}_B^T(k)\ \mathbf{u}_B^T(k+1) \dots \mathbf{u}_B^T(k+h-1)\right]^T$$

and

$$\boldsymbol{\Lambda}(k,i) = \mathit{\Pi}_{j=i}^{h-1}\mathbf{A}(k+j|k)$$

the robot-tracking prediction-error vector is written in the form

$$\mathbf{E}^*(k) = \mathbf{F}(k)\mathbf{e}(k) + \mathbf{G}(k)\mathbf{U}_B(k) \tag{12}$$

where

$$\mathbf{F}(k) = \left[\mathbf{A}(k|k)\ \mathbf{A}(k+1|k)\mathbf{A}(k|k)\ \dots\ \boldsymbol{\Lambda}(k,0)\right]^T, \tag{13}$$

and $\mathbf{G}(k) = [g_{ij}]\,,\ i = 1, ..., n,\ j = 1, ..., b,\ b = max(h, n)$, $g_{11} = \mathbf{B}(k|k)$,$g_{21} = \mathbf{A}(k+1|k)\mathbf{B}(k|k)$, $g_{22} = \mathbf{B}(k+1|k)$, $g_{n1} = \boldsymbol{\Lambda}(k,1)\mathbf{B}(k|k)$, $g_{n2} = \boldsymbol{\Lambda}(k,2)\mathbf{B}(k+1|k)$, $g_{nh} = \mathbf{B}(k+h-1|k)$. and $\mathbf{F}(k) \in \mathbb{R}^{n \cdot h} \times \mathbb{R}^n$, $\mathbf{G}(k) \in \mathbb{R}^{n \cdot h} \times \mathbb{R}^{m \cdot h}$.

The objective of the control law is to drive the predicted robot trajectory as close as possible to the future reference trajectory, i.e., to track the reference trajectory. This implies that the future reference signal needs to be known. Let us define the reference error-tracking trajectory in state-space as $\mathbf{e}_{ri}(k+i) = \mathbf{A}_{ri}^{i}\mathbf{e}(k)$. for $i = 1, \ldots, h$. This means that the future control error should decrease according to dynamics defined with the reference model matrix $\mathbf{A}_{ri}$. Defining the robot reference-tracking error vector

$$\mathbf{E}_{ri}^{*}(k) = \left[\mathbf{e}_{ri}(k+1)^T\ \mathbf{e}_{ri}(k+2)^T \ldots \mathbf{e}_{ri}(k+h)^T\right]^T,\ \mathbf{E}_{ri}^{*} \in \mathbb{R}^{n \cdot h}$$

for the whole interval of observation (h) the following is obtained

$$\mathbf{E}_{ri}^{*}(k) = \mathbf{F}_{ri}e(k),\ \mathbf{F}_{ri} = \left[\mathbf{A}_{ri}\ \mathbf{A}_{ri}^{2}\ \ldots\ \mathbf{A}_{ri}^{h}\right]^T,\ \ \mathbf{F}_{ri} \in \mathbb{R}^{n \cdot h} \times \mathbb{R}^{n}. \tag{14}$$

Control law The idea of MPC is to minimize the difference between the predicted robot-trajectory error and the reference robot-trajectory error in a certain predicted interval.

The cost function is, according to the above notation, now written as

$$J(U_B) = \left(\mathbf{E}_{ri}^{*} - \mathbf{E}^{*}\right)^T \overline{\mathbf{Q}}\left(\mathbf{E}_{ri}^{*} - \mathbf{E}^{*}\right) + \mathbf{U}_B^T \overline{\mathbf{R}} \mathbf{U}_B\ . \tag{15}$$

The control law is obtained by the minimization ($\frac{\partial J}{\partial \mathbf{U}_B} = 0$) of the cost function and becomes

$$\mathbf{U}_B(k) = \left(\mathbf{G}^T\overline{\mathbf{Q}}\mathbf{G} + \overline{\mathbf{R}}\right)^{-1}\mathbf{G}^T\overline{\mathbf{Q}}\left(\mathbf{F}_{ri} - \mathbf{F}\right)\mathbf{e}(k) \tag{16}$$

where $\overline{\mathbf{Q}} = diag(\mathbf{Q})$ and $\overline{\mathbf{R}} = diag(\mathbf{R})$. This means that $\overline{\mathbf{Q}} \in \mathbb{R}^{n \cdot h} \times \mathbb{R}^{n \cdot h}$ and $\overline{\mathbf{R}} \in \mathbb{R}^{m \cdot h} \times \mathbb{R}^{m \cdot h}$.

Let us define the first m rows of the matrix $\left(\mathbf{G}^T\overline{\mathbf{Q}}\mathbf{G} + \overline{\mathbf{R}}\right)^{-1}\mathbf{G}^T\overline{\mathbf{Q}}\left(\mathbf{F}_{ri} - \mathbf{F}\right) \in \mathbb{R}^{m \cdot h} \times \mathbb{R}^{n}$ as $\mathbf{K}_{mpc}$. Now the feedback control law of the model predictive control is given by

$$\mathbf{u}_B(k) = \mathbf{K}_{mpc} \cdot \mathbf{e}(k),\ \mathbf{K}_{mpc} \in \mathbb{R}^{m} \times \mathbb{R}^{n} \tag{17}$$

6 Simulation Results

In this section the simulation results of the optimal cooperative collision avoidance between three mobile robots are shown. The study was made to elaborate the possible use in the case of a real mobile-robot platform. In the real platform we are faced with the limitation of control velocities and accelerations. The maximal allowed tangential velocity and angular velocity were $v_{max} = 0.5\ m/s$ and $\omega_{max} = 13\ rad/s$, while the maximal allowed tangential wheel acceleration is $a_{max} = 3m/s^2$. Because of relatively hight maximal angular velocity and tangential wheel acceleration only the tangential velocity was taken into account to define the maximal time between the start position of the robots and the goal position, which is defined as $T_{max} \geq \frac{\max v(\lambda)}{v_{max}} = \frac{2.1m}{0.5ms^{-1}} = 4.02s$ where the maximal normalized tangential velocity $\max_i v_i(\lambda) = 2.1m$ is defined from Fig. 3. The starting pose of the first mobile robot R_1 in generalized coordinates is defined as $\mathbf{q}_{s1} = \left[0, 1, \frac{\pi}{2}\right]^T$ and the goal pose as $\mathbf{q}_{g1} = \left[1, 0, -\frac{\pi}{4}\right]^T$. The second robot

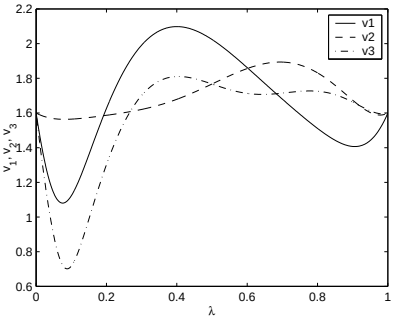


Fig. 3. The velocities of avoiding robots R_1, R_2 and R_3 in normalized time variable.

R_2 starts in $\mathbf{q}_{s2} = \left[1, 0, -\frac{3\pi}{4}\right]^T$ and has the goal pose $\mathbf{q}_{g2} = \left[0, 1, \frac{3\pi}{4}\right]^T$. The third robot R_3 has the start pose $\mathbf{q}_{s3} = \left[0, 0, -\frac{\pi}{4}\right]^T$ and the goal pose $\mathbf{q}_{g3} = \left[1, 1, \frac{\pi}{4}\right]^T$. The x and y coordinates are defined in meters. The safety distance is defined as $d_s = 0.35m$. The parameter d, which is used to define the control points P_{1i} and P_{2i}, equals $0.4m$ ($\min_{ij} D_{ij} = 1m, \gamma = 0.4$). In Fig. 4 the distances between the mobile robots are

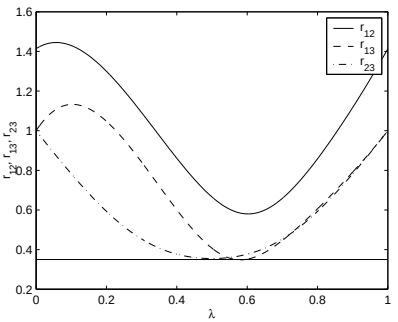


Fig. 4. The distances between avoiding robots R_1, R_2 and R_3.

shown. It is also shown that all the distances r_{12}, r_{13} and r_{23} satisfy the safety-distance condition. They are always bigger than prescribed safety distance d_s.

7 Conclusion

The optimal cooperative collision-avoidance approach based on Bézier curves allows us to include different criteria in the penalty functions. In our case the reference path of each robot from the start pose to the goal pose is obtained by minimizing the penalty function, which takes into account the sum of all the paths subjected to the distances

between the robots, which should be bigger than the minimal distance defined as the safety distance. Current approach as presented does not include explicit velocity and acceleration constraints to be imposed to each robot, this remaining the future research work.

References

1. R.C. Arkin, Cooperation without communication: multiagent schema-based robot navigation, J Robot Syst., 1992, 9, 351–364,
2. L. Shan and T. Hasegawa, Space reasoning from action observation for motion planning of multiple robots: mutal collision avoidance in a narrow passage, J Robot Soc Japan, 1996, 14, 1003–1009.
3. C.G. Kim and M.M.Triverdi, A neuro-fuzzy controller for mobile robot navigation and multirobot convoying, IEEE Trans Sys Man Cybernet SMC Part-B, 1998, 28, 829–840.
4. I. Kolmanovsky and N. H. McClamroch, Developments in Nonholonomic Control Problems, IEEE Control Systems, 1995, 15, 6, 20–36.
5. N. Sarkar, X. Yun and V. Kumar, Control of mechanical systems with rolling constraints: Application to dynamic control of mobile robot, The International Journal of Robotic Research, 1994, 13, 1, 55–69.
6. G. Oriolo, A. Luca and M. Vandittelli, WMR Control Via Dynamic Feedback Linearization: Design, Implementation, and Experimental Validation, IEEE Transactions on Control Systems Technology, 2002, 10, 6, 835–852.
7. C. Canudas de Wit and O. J. Sordalen, Exponential Stabilization of Mobile Robots with Nonholonomic Constraints, IEEE Transactions on Automatic Control, 1992, 37, 11, 1791–1797.
8. A. Luca, G. Oriolo and M. Vendittelli, Control of wheeled mobile robots: An experimental overview, RAMSETE - Articulated and Mobile Robotics for Services and Technologies, 2001, S. Nicosia, B.Siciliano, A. Bicchi, P. Valigi, Springer-Verlag.
9. Y. Kanayama, Y. Kimura, F. Miyazaki and T. Noguchi, A stable tracking control method for an autonomous mobile robot, Proceedings of the 1990 IEEE International Conference on Robotics and Automation, 1990, 1, 384–389.
10. F. Pourboghrat and M. P. Karlsson, Adaptive control of dynamic mobile robots with nonholonomic constraints, Computers and electrical Engineering, 2002, 28, 241–253.
11. T. C. Lee, K. T. Song, C. H. Lee and C. C. Teng, Tracking control of unicycle-modeled mobile robots using a saturation feedback controller, IEEE Transactions on Control Systems Technology, 2001, 9, 2, 305–318.
12. R. W. Brockett, Asymptotic stability and feedback stabilization, Differential Geometric Control Theory, 1983, R. W. Brockett, R.S. Millman and H. J. Sussmann, 181–191.

A Complexity Theory Approach to Evolvable Production Systems

Regina Frei[1], José Barata[1]
and Giovanna Di Marzo Serugendo[2]

[1] New University of Lisbon, Quinta da Torre, 2829-516 Caparica, Portugal
{regina.frei, jab}@uninova.pt
[2] Birkbeck (University of London), Malet Street, London WC1E 7HX, United Kingdom
dimarzo@dcs.bbk.ac.uk

Abstract. Evolvable Production Systems differ from Reconfigurable and Holonic Manufacturing Systems by implying ontology-based process-specific modularity at fine granularity with local intelligence and a distributed control solution based on the Multi-Agent paradigm. Understanding the dynamics of such complex production systems is not feasible with traditional engineering. For creating the manufacturing systems of the future, engineers need to dare a leap in their ways of thinking. Complexity Theory and Artificial Intelligence can be a valuable source of inspiration for manufacturing engineers. This article illustrates how ideas from these scientific areas fit the problems and open questions of manufacturing. Some concepts, as Self-Organization and Emergence, need adaptation to be applicable in production systems; others simply require the right perspective. Finally, a vision of future EPS is outlined.

1 Introduction

Evolvable Production Systems, short EPS [1, 2], are a concrete solution to the requirements from the market such as stated within the Agile, Reconfigurable and Distributed approaches: they include high responsiveness, low down-times, ability to handle small series with many variants, and on-the-fly changeability. Together with ontology-based process-specific modules at fine granularity, a distributed control system using the Multi-Agent paradigm permits to quickly and cost-effectively adapt to ever-changing production requirements. The inspiration from Artificial Intelligence, Mobile Robots, Complexity Theory and Biology as well as other emerging sciences, as detailed in this article, will help EPS to cope with the turbulent environment, many-to-many multi-directional relationships and incomplete data and knowledge.

EPS have similarities with the Bionic, Fractal and Holonic approaches [3, 4], but besides considering system morphology, EPS strongly link product, process and system (see Fig. 1) by the means of detailed ontologies. As EPS, Emergent Synthesis, a Biological Manufacturing Systems approach [5], also focuses on self-organization,

however lacks mechanisms usable for practical implementation and the product-process background.

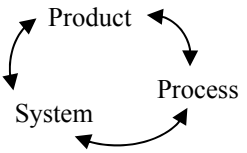


Fig. 1. Strong relations.

The purpose of this article is to show that Complexity Theory, Artificial Intelligence and related domains can be a valuable source of inspiration for manufacturing engineers, and to illustrate in which way many ideas found in these scientific areas fit the problems and open questions of the manufacturing world. Section 2 briefly explains the concept of evolvability in manufacturing as well as the distributed control approach required for Evolvable Production Systems (EPS). Section 3 illustrates the main sources of inspiration for new way of thinking, and some suitable concepts found in Complexity Theory and Artificial Intelligence are detailed. Among others, Emergence and Self-Organization are fundamental for EPS. Section 4 explains in which way they could be understood, and what their implications for production systems are. With their help, systems with far more advanced capabilities can be imagined, as outlined in section 5: the vision of future production systems. A brief summary follows in the conclusion.

2 Evolvable Production Systems

Evolvable Production Systems take complex systems in nature a metaphor for their own need to continuously adapt to an ever-changing environment. In this sense and in the context of manufacturing, Evolvability means the ability of complex systems to co-evolve with the continuously changing requirements, to undergo changes of different significance, from small adaptations on-the-fly to more important transformations. Ontology-based modularity at a fine granularity level, the modules' plugability as well as a powerful control approach based on the multi-agent paradigm are fundamental.

Evolvability is an enabler for tomorrow's production systems. Using a concept similar to LEGO together with local intelligence, they allow the user to build any required system and to modify it at wish. Through their module re-usability and life-cycle support, EPS considerably lower the system cost and enable the automation even in case of low production volumes and small lot sizes with frequent changes. Thanks to standardized, open interfaces, systems can gradually evolve through the addition, removal or exchange of modules.

The EPS control approach, avoiding re-programming, is crucial to ensure the modules' rapid plugability. Distributed approaches have the important advantage of low complexity in the individual control parts. They are modular and, by their nature,

show emergent robustness when facing disturbances, component failure or other critical situations. Agent technology ideally matches distributed systems [6]. Co-BASA [7] is an example of a Multi-Agent Shop-Floor Control System which focuses on rapid system reconfiguration. Equipment resources are represented by agents and form coalitions according to the current production requirements, given by order agents. In the operation phase, product agents ask to be treated in the way specified by their process plan. Agents exhibit both reactive and proactive attitudes and are referred to as "intelligent" and having "social behavior" based on a corresponding ontology.

3 Sources of Inspiration and Relevant Concepts

Numerous scientific domains have emerged in the last few years, investigating phenomena which EPS also exhibit. They can provide helpful tools and valuable theoretical background to cope with the complexity of manufacturing systems (see Fig. 2).

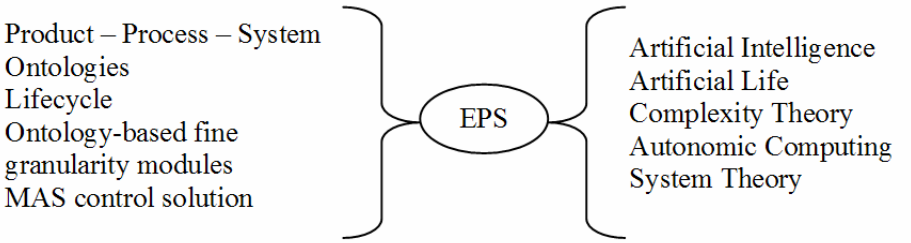


Fig. 2. Fundamental concepts and sources of inspiration for EPS.

3.1 Sources of Inspiration

In **Artificial Intelligence** the goal is often to create autonomous, intelligent behavior, learning capabilities, and adaptation mechanisms in machines used for sophisticated tasks. Typical examples are expert systems, which, in the case of EPS, provide support for human decision making. Machine learning can be helpful for improving equipment calibration procedures or for the automatic creation of complex skills based on simple skills in coalitions of equipment modules.

Complexity Theory looks for simple causes leading to complex behaviors [8]. Complex systems are spatially and/or temporally extended non-linear systems with many strongly-coupled degrees of freedom and high non-linearity. They are composed of numerous often simple elements and characterized by collective properties. EPS consist of equipment modules which are connected to each other and have multilateral interactions. Together, the modules form a system with the desired global behavior.

Chaos Theory studies cases where future outcomes are arbitrarily sensitive to tiny changes in present conditions [9]. The mathematical methods founded by Poincaré and Lorentz try to find patterns in this seemingly chaotic situations. Manufacturing

systems often exhibit sensitivity to specific conditions and to disturbances. Certain factors lead to system breakdown while others have no significant effect. It is difficult to predict the critical circumstances and to cope with them.

A promising engineering approach based on Complexity Theory is described in "Foundations of Complex System Theories" [10]: the Synthetic Microanalysis. Combining the bottom-up and top-down views, it proposes an iterative journey from the whole to its parts and back.

System Theory and Cybernetics. All systems, however complex they are, have some kind of organization [11]. This structures or concepts, studied in System Theory, are often independent from the specific system or domain. In this sense, their understanding can help solving problems in a somehow generic way: the approaches can be applied to other cases – above all complex, adaptive and self-regulating systems. Cybernetics particularly treats the aspects of communication and control by focusing on circular feedback mechanisms in complex systems [11]. EPS need a dynamically modifiable organization. Their structure as well as constituents' interactions is fundamental for the good functioning of the systems. The trade-off between system autonomy and human control is a challenge for engineers.

Artificial Life including Swarm Theory and Mobile Robots. Scientists attempt to create life-like behaviors with the capability of evolution on computers and other "artificial" media. EPS are very similar to artificial living systems. They have a modifiable structure, will exhibit some kind of self-organization, can adapt to their environment, and react to stimuli. They are capable of evolving according to the circumstances, namely in terms of equipment states, and can incorporate newly available technology. As any living organism, they will include efforts to keep themselves in a constant well-functioning state through self-surveillance and self-management.

The concepts of swarm-building living organisms, such as stigmergy and coordination mechanisms found in schools of fish and bird flocks can for instance be used by mobile robots for the coordination with their fellows. The robots' autonomy and their capacity of collaboration are fundamental. Being reactive and proactive devices, they often include reasoning capabilities.

Agentified modules in EPS can be seen like the members of a swarm: their coordination can be based on similar strategies. Even if their mechanical properties are diverse, from a software point of view, they have similar or identical characteristics. They can participate in a coalition or withdraw from it, without disturbing the rest of the group, and thus permit true and immediate Plug&Produce functionality.

Autonomic Computing [12]. Although at another level than the other areas described above, Autonomic Computing provides a fundamental source of inspiration for EPS. Large computer-based systems, forming large networks and having complex and multiple interactions, become increasingly difficult to manage. As a consequence, software will be designed to itself undertake most management tasks, such as self-configuration, self-healing, self-protection and self-optimization. User interaction will be minimized and reprogramming avoided. Valid for computers, the concept of autonomic systems applies also to manufacturing systems in general and EPS in particular. Complexity must be hidden from the user. Systems need easy-to-use human machine interfaces.

3.2 Relevant Concepts

Out of these numerous fields of scientific studies, a set of the most relevant concepts is identified. Many of them are included in several domains and therefore no specific origin is indicated here.

Depending on the context, an **Agent** can be a human person, an association, an animal, or a piece of software, possibly connected to some hardware. The fundamental characteristics are identity, intelligence and the ability to act and react in order to persecute goals. Agents have at least a certain degree of autonomy and can compete or collaborate with others. The realization of Multi-Agent Systems can adopt various software technologies: early attempts used object-oriented or component-based languages and evolved towards programming languages and platforms directly supporting the concepts of agents [6]. Also web-services are an option, as used by Schneider Electric in their Service-Oriented Architecture [13]. When extended by a proactive part, web-services are de facto very close to agents. There are numerous successful experiences with agent-based systems in industry [14-17]. Rockwell Automation even develops agent-based systems where the agents run inside the PLC itself [18] instead of on separate computers. In EPS, Agents naturally represent the basic building blocks embedded into the different components of the production system.

Self-* capabilities as defined by AgentLink III [19] can concern installation, management, healing, configuration and other activities. EPS modules with self-* capabilities allow to minimize user interaction, i.e. to increase system autonomy. Self-Organization is of particular relevance: it is abundant in nature and a promising feature for artificial systems. A distributed diagnosis system for EPS, based on device Self-Diagnosis, is currently being developed at UNINOVA, Portugal.

Emergence. Complex systems most often consist of at least two different levels: the macro-level, considering the system as a whole, and the micro-level, considering the system from the point of view of the local components. Local components behave according to local rules and based on preferably local knowledge; a representation of the entire system or knowledge about the global system functionality is neither provided by a central authority nor reachable for the components themselves. They communicate, interact with each other and exchange information with the environment. From the interaction in this local world emerge global phenomena, which are more than a straight-forward composition of the local components' behaviors and capabilities. Typically, there is a two-way interdependence: not only is the global behavior dependent on the local parts, but their behavior is also influenced by the system as a whole. Emergent phenomena are scalable, robust, and fault-tolerant, i.e. insensitive to small perturbations and local errors as well as component failure, thanks to redundancy. They exhibit graceful degradation, meaning that there is no total break-down because of minor local errors.

Fitness functions and landscapes. In nature, organisms must be fit for survival and in this sense react to the requirements of the ever-changing environment. The closer an organism matches the fitness function, the better adapted it is to the current life

condition. The criteria for endurance or elimination of new characteristics are most often multiple and form a “fitness landscape”. In the scope of EPS, process requirements are the system’s fitness functions / landscapes. Certain specifications are absolute: the marks must be absolutely reached – otherwise the process is not fulfilled. Others may indicate a direction, which the system can try to converge to (e.g. save energy, minimize cycle time, etc.).

Edge of Chaos, Far-from-equilibrium, Self-organized criticality. Constantly stable equilibrium states would block evolution. Dynamic systems get again and again into states where a little stimulus can trigger a major reaction. This gives the systems energy to evolve and makes new phenomena emerge. Illustrative explanations can be found in the books Tipping Point [20] and Critical Mass [21].

Complex Adaptive Systems (CAS) are systems that emerge over time into a coherent form, and adapt and organize themselves without any singular entity deliberately managing or controlling it [22]. Supply Networks have been recognized as CAS [23], and also EPS share many characteristics of them. They are many-body systems, composed of numerous elements of varying sophistication, which interact in a multi-directional way to give rise to the systems global behavior. The system is embedded in a changing environment, with which it exchanges energy and information. Variables mostly change at the same time with others and in non-linear manner, which is the reason why it is so difficult to characterize the system’s dynamical behavior.

4 Self-Organization and Emergence in EPS

In areas such as biology and artificial life, emergence and self-organization have been discussed for many years and accordingly, definitions exist. Also for Multi-Agent Systems, these topics have been investigated [24, 25]. Their interpretation in scope of EPS is detailed here.

4.1 Self-Organization in EPS

Reasons for implementing self-organization in EPS are to minimize and facilitate user interaction, i.e. to hide complexity and increase system autonomy. Building and configuring a system composed of numerous entities with multi-lateral interactions is a highly complex task; the more autonomy the system has, the easier it gets for the user. Production systems tend to have many components of diverse nature which interact in many coupled ways. Agents need the capacity of (re-)organizing their collaboration themselves, in different forms and compositions, according to the needs, without passing through a central coordination point.

Self-organization is robust and adaptive with regard to its environment. In presence of perturbations and change, the system is capable of changing its organization while still maintaining its functionality. This means in practice that the control system should be capable of handling problems and if necessary finding alternative produc-

tion ways. A major challenge in manufacturing applications is to let the system self-organize and at the same time, determine its behavior. Different from natural self-organized systems, artificial systems respectively EPS may require a kind of leader, a broker or (eventually human) decision maker. The control influence of this authority may be punctual in time and scope, e.g. at important strategic points.

4.2 Emergence in EPS

To bring the classical notions of emergence, discussed before, closer to the reality of engineered systems, two classes of emergence are proposed: For "full / complex emergence", the global level must show further development. There is non-linear dependence of the global functionality on the components and their interactions between themselves and the environment. "Basic / primitive emergence" means that the local-to-global dependence may be "quasi-linear" – but still, the appearance of the global phenomenon is not self-evident and needs some kind of "inspiration". An example is the classical Pick & Place mechanism: there are many different ways of putting together a gripper with translation / rotation axes – but not all of them lead to the desired functionality.

Not all equipment units are of the same granularity: an entire robot may as well be defined as a module, as a single actuator or a gripper, a gripper finger may be. Sensors and other fine granularity devices can play an important role in the emergence of complex skills: augmented with the right sensors, an axis does not only move, it can then detect the presence of other objects, determine distances or execute its own movement as a function of the state of others.

Some of the emergent phenomena will be favorable to the accomplishment of the system's task and have considerable potential for advanced system behaviors, such as the emergence of complex capabilities out of simple ones. These favorable emergent phenomena could and should be exploited. Others may be less adapted, disturbing or even harmful: e.g. system integration is supposed to function without unexpected symptoms. In nature, unsuccessful properties will be eliminated by the survival-of-the-fittest selection. Obviously, such a mechanism is not viable in manufacturing environment: harmful behavior cannot be allowed at any moment. How to cope with this problem in the case of EPS? Simulation can be helpful. Safety measures have to be taken in order to avoid problematic and dangerous situations.

5 Vision of Future EPS

Computing is becoming ubiquitous; little computing power devices will be present in every device. Manufacturing systems can then become powerful, easy-to-use and gradually more autonomous. EPS of the future might autonomously cover a large range of procedures, far more than today's production systems can. They will receive specifications of what to do, but not how to achieve it and which resources to use. This could lead to the following scenario:

To release an order into the system, *product agents* will be accordingly configured. They will carry their assembly plan and ask to be treated by *operation agents*. The kind of actions to be executed on the product parts, including specifications on precision, cycle time and other special needs, will be identified. This means that the exact *process requirements* will be determined, e.g. the way of picking a part, the geometrical trajectory to be made, the way of bringing parts from a feeder to its place of insertion, etc. The system must be capable to find and organize the right resources for each function, to grant for their successful execution as well as sustainable system management, as will be detailed below.

The *use of the equipment resources* is yet to be chosen: this happens by exploring the existing system with the help of a dynamically updated map of the shop floor respectively the present modules as well as the modules available in the storage department or eventually in vendors' module pools. In collaboration between operation agents and *resource agents*, several possibilities of executing the required processes will be determined and the best will be chosen. The criteria for this selection can be a standard set or specified by the user. Finally, the user will be informed about possible necessary addition, displacement or removal of modules. The required resources will be autonomously configured / calibrated for the processes, and the needed resource coalitions will be formed to create the complex skills. No re-programming is needed. Agents work autonomously and collaborate with other agents. Modules register in a resource so-called cluster and, from this platform, to dynamically form coalitions with other resource agents according to the incoming production requirements.

The entire *real time execution* is then taken over by the system, which functions with a high degree of autonomy, too. Modules at fine granularity, incorporating reactive and proactive intelligence, will exhibit *self-* capabilities*. Systems are able to do self-surveillance and self-maintenance, i.e. to observe their internal state and if necessary take corresponding measures, to schedule regular maintenance, to announce the eventual need for staff interaction before problems become acute. Autonomic systems self-optimize, meaning that they continuously search for ways to optimize their operation. They can self-diagnose and self-heal in order to predict and avoid respectively solve most problems autonomously and, in case of the need for user interaction, make it as easy as possible by indicating the defective part, the problematic part interaction and proposing corresponding corrective measures to the shop floor staff.

Of course, such a fundamental change in the way systems are built and especially run cannot be achieved from day to the next; the approach has to advance *step by step*. E.g. concerning decision making, the systems can neither be expected to become fully autonomous at once nor can users trust them immediately. Gradually increasing independence is more sustainable. At first, the system automatically collects information in order to support human decision making. In a next phase, it gives advice and proposes solutions, taking note of user preferences. Later, the system indicates the best-fitting solution as well as alternatives and requests user confirmation. And finally, after these learning phases, the system takes decisions alone.

In this sense and as a conclusion, systems might evolve in a way that they develop capabilities which the system designers have never thought of: systems could eventually offer services which they have not originally been built for.

6 Conclusion

In order to cope with today's and tomorrow's manufacturing needs, new solutions are required. *Evolvability* is a key to success: the capability of systems to evolve together with the production requirements as well as the strong product-process-system link are crucial. EPS provide ontology-based process-specific modularity at a fine granularity and a distributed control approach using the *Multi-Agent paradigm*. *Self-Organization* and *Emergence* allow system autonomy, which can considerably facilitate system installation and operation by hiding complexity.

EPS share many aspects of Complex Adaptive Systems and therefore need to be addressed as such. Traditional engineering cannot offer corresponding tools. Complexity Theory and other emerging scientific domains have the potential of providing valuable help to cope with CAS and the engineering of such systems, offering the possibility of implementing advanced system capabilities.

The ideas described in this article require a stepwise approach. Some of them still need theoretical elaboration, while others are already fully or partially implemented on a prototype at UNINOVA, Portugal. The NOVAFLEX assembly system has been agentified: each module is an agent, and the multi-agent control system is operational. The lab installations are built of legacy components from diverse suppliers and include two industrial robots, different grippers, a warehouse and conveyor circuits. A distributed diagnosis system, based on a Service-Oriented Architecture and using device Self-Diagnose, is currently being developed. In parallel, tiny computing devices for supporting MAS are being elaborated. Integrated in any kind of equipment unit, they will make computing capacities ubiquitous, also in the manufacturing world.

References

1. M. Onori, J. Barata, and R. Frei, "Evolvable Assembly Systems Basic Principles " presented at BASYS, Niagara Falls - Canada, 2006.
2. T. Maraldo, M. Onori, J. Barata, and D. Semere, "Evolvable Assembly Systems: Clarifications and Developments to Date," presented at CIRP / IWES 6th International Workshop on Emergent Synthesis, Kashiwa - Japan, 2006.
3. A. Tharumarajah, A. J. Wells, and L. Nemes, "Comparison of the bionic, fractal and holonic manufacturing system concepts," *International Journal Computer Integrated Manufacturing*, vol. 9, pp. 217-226, 1996.
4. M. Ulieru, "Emerging Computing for the Industry: Agents, Self-Organisation and Holonic Systems," presented at Workshop on Industrial Informatics, IECON 2004, Busan, South Korea, 2004.
5. K. Ueda, "Emergent Synthesis Approaches to Biological Manufacturing Systems," presented at DET, Setubal, Portugal, 2006.
6. M. J. Wooldridge, *An Introduction to Multiagent Systems*. New York: J. Wiley, 2002.
7. J. Barata, G. Cândido, and F. Feijão, "A Multiagent Based Control System Applied to an Educational Shop Floor," presented at BASYS, Niagara Falls - Canada, 2006.
8. K. A. Delic and R. Dum, "On the Emerging Future of Complexity Sciences," *ACM Ubiquity*, vol. 7, 2006.

9. M. Gell-Mann, "What is Complexity?," in *Complexity*, vol. 1, www.santafe.edu/~mgm/complexity.pdf ed: John Wiley and Sons, Inc. , 1995.
10. S. Y. Auyang, *Foundations of Complex -System Theories in Economics, Evolutionary Biology, and Statistical Physics*: Cambridge University Press, 1998.
11. F. Heylighen, C. Joslyn, and V. Turchin, "What are Cybernetics and Systems Science?," in *Principia Cybernetica Web* Brussels: http://pespmc1.vub.ac.be/CYBSWHAT.html, 1999.
12. J. O. Kephart and D. M. Chess, "The Vision of Autonomic Computing," *IEEE Computer*, vol. 0018-9162/03, pp. 41-50, 2003.
13. A. W. Colombo, F. Jammes, H. Smit, R. Harrison, J. L. M. Lastra, and I. M. Delamer, "Service-oriented architectures for collaborative automation," presented at IECON, 2005.
14. W. Shen and D. H. Norrie, "Agent-Based Systems for Intelligent Manufacturing: A State-of-the-Art Survey," *Knowledge and Information Systems, an International Journal*, vol. 1, pp. 129-156, 1999.
15. H. V. D. Parunak, "Agents in Overalls: Experiences and Issues in the Development and Deployment of Industrial Agent-Based Systems," *International Journal of Cooperative Information Systems*, 2000.
16. V. Marik and D. C. McFarlane, "Industrial Adoptation of Agent-Based Technologies," *IEEE Intelligent Systems*, vol. 1542-1672/04, pp. 22-30, 2004.
17. L. Monostori, J. Vancza, and S. R. T. Kumara, "Agent-Based Systems for Manufacturing," presented at BASYS, Canada, 2006.
18. V. Mařík, P. Vrba, K. H. Hall, and F. P. Maturana, "Rockwell automation agents for manufacturing," presented at AAMAS, Utrecht, NL, 2005.
19. M. Luck, P. McBurney, O. Shehory, and S. Willmott, *Agent Technology Roadmap*, 2005.
20. M. Gladwell, *The Tipping Point: how little things can make a big difference*. London: Abacus, 2000.
21. P. Ball, *Critical Mass: how one thing leads to another*. London: Arrow Books, 2004.
22. J. H. Holland, *Hidden Order: How Adaptation Builds Complexity*, 1995.
23. T. Y. Choi, K. J. Dooley, and M. Rungtusanatham, "Supply Networks and Complex Adaptive Systems: Control versus Emergence," *Operations Management*, vol. 19, pp. 351-366, 2001.
24. G. Di Marzo Serugendo, M.-P. Gleize, and A. Karageorgos, "Self-Organisation and Emergence in MAS: An Overview," *Informatica*, vol. 30, pp. 45-54, 2006.
25. S. A. Brueckner, G. Di Marzo Serugendo, A. Karageorgos, and R. Nagpal, "Engineering Self-Organising Systems," in *LNAI 3464*. Berlin Heidelberg: Springer, 2005, pp. 297.

Multi-Robot Task Allocation with Tightly Coordinated Tasks and Deadlines using Market-Based Methods

Robert Gaimari, Guido Zarrella and Bradley Goodman

The MITRE Corporation, 202 Burlington Road, Bedford MA 01730, USA
rgaimari@mitre.org, jzarrella@mitre.org, bgoodman@mitre.org

Abstract. The domain of Collaborative Time Sensitive Targeting requires agents to assess and prioritize tasks, dynamically form heterogeneous teams of agents to perform the tightly coordinated tasks, and to complete them within time deadlines. In this paper, we describe extensions to market-based, multi-robot task allocation to allow for these requirements.

1 Introduction

The motivation for the research described in this paper is to extend the state of the art in market-based multi-robot planning algorithms to handle the challenges presented by real world domains with tightly-coordinated and time-constrained tasks. Past research into market-based robot coordination algorithms [2, 3, 11] has been motivated by an attempt to design an algorithm to reason in an efficient fashion about resource utilization and task allocation while preserving the ability to quickly and robustly respond to a dynamic environment.

Prior work has demonstrated the effectiveness of the TraderBots algorithms [2, 3] in several domains. It has been used in domains with tightly coordinated tasks requiring heterogeneous, dynamically formed teams [7]. It has also been used in domains requiring homogeneous teams to perform tasks with time deadlines [8]. There are other real world applications that bring together elements from both of these areas. The Collaborative Time Sensitive Targeting (TST) problem, for instance, is a domain in which agents must assess and prioritize multiple tasks, form heterogeneous teams dynamically, and perform tightly-coordinated tasks with tight deadlines. Search and rescue is one real-world example of a TST problem. Multiple rescue workers, each having different skills, must work together to save the maximum number of victims within a limited time period. New information will be discovered during the rescue, forcing the workers to reassess and reprioritize.

The TST domain requires tightly-coupled coordination between agents, as agents with complementary capabilities are required to form sub-teams in order to successfully perform a task. The TST domain also requires agents to reason about tasks with time deadlines.

Section 2 of this paper explains our task domain, a type of TST problem. Section 3 describes the standard approach of market-based, multi-robot task allocation, as

presented in [2]. Section 4 describes our implementation in detail, along with our extensions to the standard approach. And in Section 5, we show our results.

2 Task Domain

Our task domain is a variant of the TST problem. In our scenario, autonomous robots perform a Search and Rescue mission by locating and treating sick sea animals. Over the course of an exercise (90 "minutes" long, sped up in the simulation), messages come in from outside the system with reports of the general locations where sick animals might be found. These animals move over time. The message will list the name of the animal (e.g. "Sick Manatee"), the approximate latitude-longitude location (either a single point, or an area of ocean), a deadline which the task has to be completed by (e.g. cure the manatee within 30 minutes or it will die), and the maximum reward offered for completing the task. If a task is completed before the deadline, the robots receive the full reward; if the deadline passes, they receive no reward, even if they complete the task. The robots already have an incentive to finish tasks as quickly as possible; the faster they complete one task, the faster they can begin a new one. If we wanted to make finishing as quickly as possible even more important, we might allow the maximum reward to drop as the deadline approached, gradually falling to zero.

In this domain, all tasks are of equal priority; therefore, the maximum rewards are the same for all tasks. If we wanted to indicate that some tasks have higher priority than others, we could set some rewards higher than others, making them more valuable to perform.

We have a heterogeneous set of robots available to perform these jobs. They are in three main groups:

1. **Radar Sensors** – Planes and boats with radar and sonar sensing capabilities. They are very fast and have large sensing range; so they can get to the location quickly and easily pinpoint where the animal is located. However, since they are sensing electronically, they cannot diagnose the illness.

2. **Video Sensors** – Planes, boats, and helicopters with video sensing capabilities. Because they have video, they are able to diagnose the disease the animal has and report it to the rest of the team. However, they are very slow and have limited sensing range.
3. **Rescue Workers** – Boats or submarines capable of capturing or curing animals in distress. They are generally about as fast as radar sensors. However, they have no sensors of their own; they must rely on reports from the sensor robots for navigation data. Also, they can act only with the proper diagnosis from a video sensor.

The robots within each group vary in their specific characteristics, such as speed and sensing range. Also, there are three separate areas of ocean to be searched; the set of

robots on each map is confined to that area. Initially, some robots are stationary near their home base and will remain there until they receive a task; others have initial paths that they follow once the clock begins, whether they have a task or not.

To complete a task, there must be a minimum of two robots: a video sensor to find the animal and make the diagnosis, and a rescue worker to administer the treatment. A radar sensor is not required, but because of its speed and sensor range, it can reduce the cost of finding the animal.

3 Market-Based Task Allocation

In a market-based system, such as the one described in [2], the problem space is modeled as an economy. The currency may be simply an abstract measure, or it may represent something concrete, such as time, fuel, resources, etc. It represents the value of performing tasks and achieving goals.

There are a set of tasks which need to be performed. These tasks will generally take one or a group of agents to complete. Each task is a source of revenue for the agents; each has a monetary reward associated with it, which is given to the agent(s) that successfully complete the task. These rewards vary according to a number of measures, such as relative priority, difficulty, risk, etc.; they are set at the beginning of the exercise, generally by the human assigning the tasks. Performing a task also costs an agent a certain amount of money, as resources must be consumed to complete them.

The players in this economy are the robots. They may be physical or virtual, depending upon the jobs they must do. In a homogenous group, all of the robots have the same capabilities and any job may be done by any robot. More complex systems may be made up of heterogeneous groups. In these systems, jobs may require several robots working together as a team to complete the mission. Individual agents are "self-interested"; that is, each agent works to earn as much profit as possibly by minimizing its own costs and maximizing its own revenue. The goal of the system as a whole is to minimize the cost of resources consumed by the team while maximizing the value of the tasks completed. Free market economic theory holds that a collection of self-interested agents will self-organize through spontaneous cooperation and competition to create an emergent, globally efficient behavior. Market-based planning algorithms seek to mimic this behavior with a simulated economy.
Tasks are distributed through auction. As each task is introduced to the system, it is given to a special type agent called an "OpTrader". An OpTrader sends out an announcement to all of the participating robots describing the task and the maximum reward available for performing the task. This is the call for bids. Each robot interested in placing a bid for the job does three things:

1. Calculate the estimated cost for performing the task. In a domain where there is no ambiguity, the robot may be able to determine exactly how much performing the task will cost. But in most realistic systems, there will be hidden information that

will not be revealed until the task is already under way. For example, there may be unknown obstacles between the robot's current location and the destination; or the destination may only be partially defined.

2. Calculate the desired profit. The profit must be high enough to make it worth the robot's time to perform the task. The robot must decide how much payment it will require to make the task worthwhile. A common method for calculating a profit margin is to use a set percentage of the costs or available revenue.

3. Calculate the bid. This is generally the cost plus the profit margin. If this bid amount is lower than the maximum reward available, then send it to the auctioneer.

The OpTrader will gather all of the bids and award the ownership of the task to the robot with the lowest bid. The difference between the maximum reward and the lowest bid is kept by the OpTrader as its profit.

Once the initial round of bids is completed, and all tasks have been awarded, each robot that owns a task may decide to put it up for another round of auction. Now that each robot has a plan of where it will be going and what it will be doing, it can have a better idea of its costs for performing other tasks. If it has already committed to performing a task in a certain location, then it might be relatively inexpensive to also perform another task in the same area. Inter-robot trading (reauctioning) allows the tasks to be redistributed to the robots that can perform that at lower costs. To complete such a deal both robots must be happy with the exchange – the buyer robot has a new task at low cost, and the seller robot has a nice profit (the difference between the maximum reward it announced and the buyer's bid) for no extra work.

This inter-robot trading will continue until there are no more mutually profitable deals to be made, at which time the robots will begin performing their tasks. As they move and explore the environment on their paths, they will be gathering additional information on the environment. As new and more accurate data becomes available, the robots may periodically put some of the tasks on their agendas up for reauctioning again; others may have discovered things that would allow them to more accurately estimate cost and buy the tasks from the seller.

When a robot completes a task, it sends a message to the agent it bought the task from, requesting payment. That agent will send a message up the chain to the next level where it got the task from, and so on. Once it reaches the top (the OpTrader), payments will work their way back down the chain, until each robot along the way has been paid what it was promised.

Auction-based task assignment and multi-robot coordination allows the system to approach an approximation of the optimal solution of minimal cost and maximal reward while keeping communication costs relatively low, avoiding single points of failure, and robustly handling robot failure and communication difficulties.

4 Implementation and Extensions

We developed our simulation and agent environment entirely in Java using the JADE agent framework [1]. Task allocation through auction was handled using inter-agent communication. The market-based task allocation is performed using the methods described in Section 3, with our extensions.

4.1 Agents

The primary type of agent in our system is the TraderAgent. Each TraderAgent is associated with a single robot in the simulation environment. When they are first initialized, TraderAgents have a back and forth communication with their assigned robots to learn their capabilities (what type of robots they are, their speeds, sensing or rescue ranges, and so on) and current status (their current locations, their remaining fuel, and, in the case of sensors, lists of objects detected and their locations). Each clock tick of the environment thereafter, each robot will send its agent the latest status report. If the TraderAgent wants its robot to move to a new location, it sends a message directing it on the new path.

A TraderAgent's primary job, as the name implies, is to trade tasks. When a new task is announced, an agent may attempt to buy it through auction. This process is described in Section 4.2. An agent which owns a task may put it up for auction, either to off-load the task to a new robot, or in team building. These re-auctioning methods are described in Section 4.3.

When tasks are introduced to the system, they are initially given to the special OpTrader agent. This agent differs from a standard TraderAgent only in that it does not have a robot associated with it. The OpTrader immediately puts tasks given to it up for auction, with a maximum offered reward that is part of the task description. When the task is completed, it is responsible for sending the payment for the job to the original purchasing agent. Because it is not associated with a robot, it cannot be a member of a team. In a system where robustness is essential, there could easily be multiple copies of these agents, either ready to take over in the event of a problem or performing their jobs in parallel with their counterparts; however, in our system, one of each is sufficient.

4.2 Bidding on Auctions

When a TraderAgent receives an auction announcement, it follows the three steps described Section 3, on standard auction-based systems.

1. The agent calculates its estimated cost for performing the task. In our system, cost is based upon the time spent performing the task. The locations given in the auction message are general locations, not specific; once sensors arrive there, there may need to be some exploration before finding the actual animal. Also, this cost

may be affected by other locations that the agent already intends to visit. If the new area is nearby somewhere the agent is already headed, the estimated cost for the new task would be lower.

2. It calculates the desired profit. The TraderAgent calculates, as its base profit, a percentage of the difference between the maximum reward offered and its estimated cost. In addition, an opportunity cost for accepting the task is calculated. This represents the likelihood that the agent will be able to purchase other tasks at a future date. The agents know the locations and capabilities of other nearby robots with similar capabilities, since this information is periodically broadcast locally. The agent selects a number of random locations on the map and simulates how many of these locations it would be likely to win in a hypothetical auction. This represents opportunity cost. An agent with a high OC can safely lose out on the current auction without fearing for its own economic well-being. Therefore, a high OC agent is able to request a higher profit margin. Meanwhile agents in less desirable circumstances are willing to lower their profit margin in an attempt to win the current auction. This means that the lowest cost agent may not always win an auction, particularly in cases where the overall team can benefit by saving a high OC agent for future tasks. We apply this opportunity cost function to help prevent over-committal of scarce resources.

3. The TraderAgent decides whether or not to place the bid, based upon cost plus desired profit. If this value is less than the maximum reward, then it places the bid. If the value is higher than the maximum reward, or if the TraderAgent realizes it cannot complete the task in the required amount of time (causing the task to be finished after the task deadline), it will decide not to place a bid and it sends a “no thanks” message to opt out of the auction.

4.3 Re-Auctioning Tasks

There are two situations in which an agent will attempt to re-auction a task it owns but has not yet completed. The first is in the team building stage. This step is critical to robots performing tightly coupled tasks. After a robot wins a task at auction, it is sent a list of other robots that already have ownership of the task. These robots make up the team that will eventually perform the task. If the team is not yet complete, for instance if no rescue worker has joined the team, the robot will place this task on the auction block with a caveat that it is only accepting bids from a certain type of buyer. At the conclusion of the auction, the winning bidder is assigned ownership of the task, but the seller retains ownership as well, and all members of the team update their team lists.

Robots will also periodically attempt to place tasks they own up onto the auction block for the purposes of offloading the task onto another robot. In this case the task can only be sold to a robot with the same capabilities as the seller. If there are no bids the seller retains ownership of the task. On the other hand, if new information has arrived and cost estimates have been updated, a different robot may choose to buy the task. This will occur only when the new robot is willing to charge less to perform the

task than the seller's expected costs, allowing both robots to earn a profit. In this case, the winning bidder replaces the seller on the original team list, and the seller is freed of its obligation.

4.4 Collecting Payment for Completed Tasks

As described earlier, the tasks in this system must be performed by a team of robots including at least one video sensor and rescue worker. This team can also optionally include a radar sensor. The rescue worker is the robot that actually finishes the task, which it accomplishes by administering medicine curing the animal. Once the rescue worker has done this, its TraderAgent reports the fact to its teammates. Each teammate then requests payment from the agent that it purchased the task from.

Our system varies in this regard, however, from the standard auction-based algorithm. In our task the teams are made up of robots that perform their jobs at greatly different speeds. Even though the estimated cost for a radar sensor may be very low, the actual cost is greatly increased by having to wait around while the slow video sensor travels to the location and does its own part of the job. Since the agents associated with fast robots generally win the tasks before the slower ones, there is no way to know ahead of time what the additional costs might be.

There are a number of ways that this inefficiency could be dealt with. For example, we could include information about slower robots in the cost estimation functions of each TraderAgent. Another option would be to have the agents, once a team is completed, share their cost estimates and allow the faster ones to recalculate. We chose a method that requires a lower amount of inter-robot communication, in which agents incur penalties on the agents below them in the chain.

The request for payment is the same. Each agent requests the amount of money it bid for the task from the agent it bought the task from. However, as the payments are distributed down the chain, each agent compares the actual cost to the estimated cost it had initially planned upon. The difference between these is deducted from the amount paid to the next agent. This agent adds, as a penalty to the next agent down, the difference in its actual cost and the cost estimate, plus the amount it was penalized by its seller. This penalty will move down the chain, until it finally winds up where it belongs, on the slowest member of the team. These payments reflect the amount of money the original agents would have bid if they had known the true cost based upon the slowest robot. See an example in Figure 1. This example assumes that there was no inter-robot trading or adding and removing of team members, but the details would be essentially the same even with other agents in the chain. Future versions of the algorithm may be able to use these penalties to learn the hidden costs of working with certain other robots.

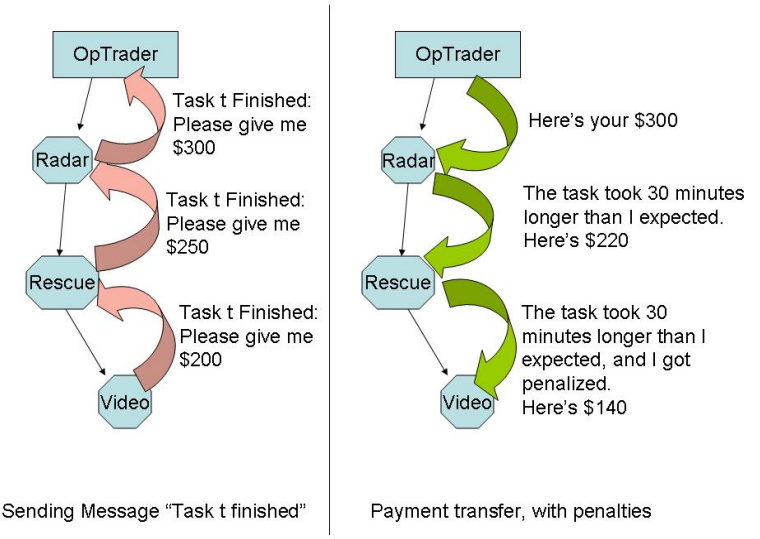


Fig. 1. Task Completion.

5 Results

The TST domain provides an ideal testbed for the market-based task allocation algorithm, since completion of each TST task requires effective communication and task assignment strategies. Furthermore the TST problem provides clear metrics for evaluating group performance, as we can measure both the percentage of tasks completed and the time needed to complete each task. We use this domain to demonstrate our hypothesis that as the self-interested agents earn money in the virtual economy, they also contribute toward the team goals and overall group performance, even when performing tightly coupled tasks under pressure of time deadlines.

We built a test problem for our robot planning algorithm by performing a direct translation of tasks used in a set of human experiments we performed into our own modified aquatic TST domain. The translation was completely isomorphic, meaning that the speeds, positions, and other attributes of all elements of the environment were identical. Only the labels and images were changed for the purposes of the demo. The robots planned and executed their own actions for 90 simulated minutes while following the rules of the team-based economic system outlined above. A total of six tasks were fed electronically at timed intervals into the robot auction environment.

The results of the robot and human teams are displayed in Table 1. The results show that the robots were able to demonstrate effective team building and task assignment strategies. The team of robots completed five of the six tasks with plenty of time to spare before the task deadlines.

Table 1. Time (hh:mm:ss) to complete each task, comparing market-based robot task allocation algorithm and human team performance.

Task:	Manatee	Killer Whale	Blue Whale	Dolphin	Sea Turtle	Toxic Leak
Robot Completion Time	0:42:00	0:40:00	1:22:00	1:09:00	1:07:00	N/A
Avg Human Team Completion Time	0:56:12	0:59:39	1:21:24	N/A	1:20:36	N/A

The robot team results also compare very favorably to the human teams' results. The robots completed five of six tasks, while no human team completed more than four. The agents were able to complete the Dolphin task, which none of the ten human teams had successfully prosecuted. The robots were also significantly faster than the best human teams in three of the four tasks that were solved by both humans and robots. The simulated agents did fail to complete one task, but none of the human teams were able to successfully complete that task either. For more details on the human version of the experiment, and comparing human/robot results, see our paper submitted to the main conference.

6 Conclusion

We have shown that market-based multi-robot task planning can be successfully extended into domains requiring tightly-coordinated actions to solve tasks with time deadlines. In the case of our simplified TST domain, teams of autonomous robots were able to provide significant gains over the performance of teams of humans.

We are not suggesting removing humans from the loop of the TST process. In real world TST of any kind, there are many decisions that cannot possibly be made without humans. Instead, our hope is that lessons learned from this research that can be applied to real world problems.

Acknowledgements

The MITRE Technology Program supported the research described here. We are also grateful for the assistance of Brian C. Williams and Lars Blackmore at the Massachusetts Institute of Technology

References

1. Bellifemine F., Poggi A., Rimassa G. (2001). Developing multi-agent systems with a FIPA-compliant agent framework. *Software-Practice and Experience*, *31*, 103–128.
2. Dias M.B. (2004). TraderBots: A New Paradigm for Robust and Efficient Multirobot Coordination in Dynamic Environments. Doctoral dissertation, Robotics Institute, Carnegie Mellon University.
3. Dias M.B., Zlot R.M., Zinck M.B., Gonzalez J.P., Stentz A. (2004). A Versatile Implementation of the TraderBots Approach for Multirobot Coordination. Retrieved from www.ri.cmu.edu/pubs/pub_7477.html
4. Gerkey B., Mataric M. (2004). Multi-robot Task Allocation: Analyzing the Complexity and Optimality of Key Architectures. *International Journal of Robotics Research*, *23* (9).
5. Goodman B., Linton F., Gaimari R., Hitzeman J., Ross H., Zarrella G. (2005). Using Dialogue Features to Predict Trouble During Collaborative Learning. *User Modeling and User-Adapted Interaction*, *15* (1-2), 85-134.
6. Grunwald T., Clark D., Fisher S. S., McLaughlin M., Narayanan S., Piepol D. (2004). Using Cognitive Task Analysis to Facilitate Collaboration in Development of Simulator to Accelerate Surgical Training. *Studies in Health Technology and Informatics*, 98, 114-120.
7. Jones E., Browning B., Dias M. B., Argall B., Veloso M., Stentz A. (2006, May). Dynamically Formed Heterogeneous Robot Teams Performing Tightly-Coordinated Tasks. Proceedings of *International Conference on Robotics and Automation*.
8. Jones E., Dias M. B., Stentz A. (2006, October). Learning-enhanced Market-based Task Allocation for Disaster Response. Tech report CMU-RI-TR-06-48, Robotics Institute, Carnegie Mellon University.
9. Lagoudakis M., Markakis E., Kempe D., Keskinocak P., Kleywegt A., Koenig S., Tovey C., Meyerson A., and Jain S. (2005). Auction-based multi-robot routing. *Robotics: Science and Systems*. Retrieved from http://www.roboticsproceedings.org/rss01/p45.pdf
10. Schneider J., Apfelbaum D., Bagnell D., Simmons R. (2005). Learning Opportunity Costs in Multi-Robot Market Based Planners. *IEEE Conference on Robotics and Automation*.
11. Walsh W., Wellman M., Wurman P., MacKie-Mason J. (1998). Some Economics of Market-Based Distributed Scheduling. *International Conference on Distributed Computing Systems*. 612-618.

BESA-ME: Framework for Robotic MultiAgent System Design

David M. Flórez, Guillermo A. Rodríguez, Juan M. Ortiz and Enrique González

SIDRe Research Group, Pontificia Universidad Javeriana, Bogotá, Colombia
egonzal@javeriana.edu.co, d.florez@javeriana.edu.co

Abstract. BESA-ME is a software middleware designed to make easier and to improve the construction of robotic control systems based on multi-agent techniques. BESA is a behavior-oriented, event-driven and social-based general purpose architecture designed to build concurrent applications using the multi-agent paradigm. The BESA abstract model incorporates the concept of behavior and the management of asynchronous events, which are very useful in the construction of robotic systems, thus it allows to design robot control architectures in a natural and direct fashion. BESA-ME, micro-edition, is the adapted BESA model that is well suited to be implemented over microcontrollers in embedded systems. Initially, it has been developed for the PIC18F chip family, and then adapted for dsPIC60F chip family, both under the real time operating system FreeRTOS ™.

1 Introduction

When dealing with the construction of complex systems, a design methodology usually includes splitting the system in a set of smaller simple tasks. This approach allows dealing with complexity and makes possible to improve the efficiency of the system implementation by distributing the execution of these tasks on a set of processing units. The agent paradigm provides a conceptual framework where systems can be analyzed and synthesized as a collection of interacting entities, using a high level abstraction, which fits in a natural fashion with the implicit concurrent requirements of robotic systems.

The development of new products implies the incorporation of innovative and more complex functionalities. The actual tendency of the technology used to improve the system efficiency is to construct multi-processor, multi-core or multi-thread machines [1]. This approach has the advantage of being easily scalable; however, in this case it is necessary to include specialized mechanisms to solve synchronization and communication problems. In addition to the high computational power requirements, some applications like smart home systems [2], robotics, automation, industrial machines and multi-robot systems [3] require that the embedded hardware communicates frequently with external stand alone systems. As a consequence, design of software for embedded systems must: generate modular concurrent solutions to deal with complexity, take advantage of hardware parallelism, and be adequate to interact in a natural fashion with external stand alone systems. It's

requirements to have models and platforms that ease the system design, taking advantage of the parallel processing capabilities of the actual and future processors.
In the context of robotic systems, the design problems previously introduced are amplified due to the fact that robots must be able to evolve in a not completely observable environment under real time constrains. In order to deal with these critical conditions, the designer must also take into account the following issues:

- manage complex synchronization of non-deterministic events coming from sensors and controlled actuators.
- distribute the application into several processes, using communication taking advantage of the parallel and specialized hardware.
- use an unified model in both the embedded and the external components of the system.
- use of a holistic approach where the robot control unit is modeled as a unique composed system; the system is seen as a collection of logical units, which can be physically deployed in the available hardware (embedded or not).

The actual software tools to model, design and implement embedded distributed and real-time systems covers different needs. Real Time Operating Systems (RTOS) are used for the implementation of systems with time-response constraints [9]. For instance, TURTLE [6] offers an environment based on the Unified Modeling Language (UML) for the formal model [7]. Another interesting approach uses the synchronous language LUSTRE, designed for the development of critical control software [8]. These approaches are process oriented, where the basic processing units are processes or tasks that usually communicate by message passing mechanisms. Even if the conceptual model provided by the notion of process is very useful, general and flexible, it has a low level of abstraction, which makes it difficult to be used in the design of complex systems. For instance, when designing a robot or multi-robot system, it would be preferable to use a conceptual model with higher degree of abstraction, where notions as behaviors and goal-oriented entities could be modeled in a more direct way.

A well suited approach to model complex problems is the Multi-Agent System (MAS) paradigm. Different activities can be distributed into several cooperative autonomous entities, and interactions are the basis for the dynamics of the system. Agents respond to events coming from its environment or derived from its interactions with other agents. The communication is the basis to construct the social level which emerges from the inter-agent interaction [10]. A MultiAgent System, MAS, is a computational cooperative system capable of executing concurrent tasks through its agents.

BESA is a MAS architecture [5] [17] that aims to solve the problems that where depicted in the precedent paragraphs. BESA provides an abstract model to construct multi-agent systems. The initial implementation of the BESA model was developed to work in a Java distributed environment. The BESA micro-edition, BESA-ME, is introduced in this paper. The BESA-ME model, architecture and implementation have been developed for embedded systems using RTOS as software support, running in microcontrollers and DSP hardware platforms. This development is mainly motivated

to deal with the requirements involved in the design of multirobot systems, where a recursive organizational approach is applied.

In this paper the BESA-ME conceptual model and architecture are explained. Then, the RTOS software and hardware considerations are analyzed in order to adjust the model to fit the constraints of a practical embedded application. Finally, the implementation strategy is depicted and the obtained results are analyzed.

2 BESA Architecture

The BESA architecture defines a conceptual model for the implementation of an agent framework. The construction of a complex and concurrent application must use this agent conceptual abstraction to model the system. Then the implementation of the system can be performed in a direct fashion exploiting the facilities provided by the BESA application framework.

2.1 Abstract Model

The BESA abstract model is supported by three principal concepts, which provide a theoretical frame that integrates a behavior-oriented, event-driven and social based approach in a coherent structure.

- Behavior-oriented: agents are composed by a collection of behaviors, simple entities charged of the treatment of a set of related events.
- Event-driven: asynchronous events unleash the behavior execution. They represent signals that could be perceived by the agents. The behavior execution is controlled determined by a guard based selector mechanism; guards forbid events to be processed if a desired condition is not attained.
- Social-based: the multi-agent system is created to form a social organization, where well-known communication patterns can be used; it is also possible to utilize mediator agents that help in the correct development of interaction protocols between agents.

BESA is a concurrent oriented architecture; agents are internally seen as a multithread system and the non-determinism, implicit in an event driven system, is managed by a select (alting) mechanism.

2.2 Agent Level

The agents give a response to events coming from the environment or from other agents. BESA events have a specific semantics and are marked with an event type label. The agent has an associated treatment for each type of event. The treatment of an event must include the rational response of the agent. When processing an event, the data attached to the event and the internal state must be taken into account. The response to an event can be produced by any kind of decision mechanism (neural networks, fuzzy logic, procedural code, rule based system, etc.). This response can

include: the selection of the appropriated actions that must be executed, sending a new event to other agents, and the modification of the internal state of the agent. Events exchanged between agents usually follow well-defined communication patterns, known as interaction protocols.

A BESA agent is structured with three main components: the channel, a set of behaviors and the agent state. Figure 1 shows these components and the way they are connected. The channel manages a mailbox where events are received, providing the unique entry point to the agent. The received events are assigned to different ports depending on their event type. Finally, they are transferred to the correct behavior, if the guard condition is verified.

The behavior is an execution space that is activated when an associated event is received. Then the corresponding treatment is executed, thus producing the rational response to the received event. An agent can have several concurrent behaviors, thus allowing to process several events at the same time. Notice that the way that the parallel execution is performed depends on the capabilities of the executing environment (hardware and software). Incoming events are received by the channel and transferred to a queue in the appropriated behavior; events received by the same behavior are processed in a sequential order. A guard verifying function and a treatment function are associated to each type of event.

The agent state provides a mechanism to store information about the agent, the environment, other agents and the global system; it is used to keep data in a persistent way. This information usually is used in the treatment of events. The state is a structured shared memory; thus, the concurrent access to this space must be synchronized. The synchronization could be implemented with semaphores or other operating system mechanisms.

2.3 Social Level

The social level aims to provide a set of predefined mechanisms and interaction protocols that could be directly used to manage agent interaction. Some of the BESA cooperation and communication services provide specialized ready to use agents that act as mediators in the social organization. For instance, the group communication service uses a mediator agent, actuating as a router, distributing events in the same order to all the agents subscribed to the group. A more detailed presentation of the social level is out of the scope of this paper. In [17], there is a more detailed description of this level.

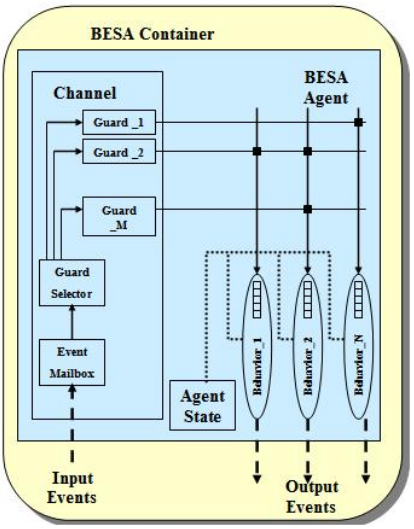


Fig. 1. Internal structure of a BESA Agent.

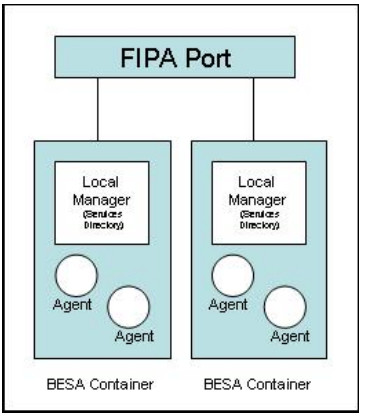


Fig. 2. BESA System Layer Model.

2.4 System Level

The system level is formed by a set of agent containers running in physical or virtual machines. The container can be seen as the execution space for the agents (Figure 2). A BESA container has a local manager in charge of handling the agent's life cycle and the directory services. The container also assures the correct communication between agents "living" in different containers. The system level is designed to comply with the FIPA standard [11]. The inter-agent interaction in the same container is performed without the local manager intervention.

In order to make easy the communication between agents, BESA includes a directory service. The white pages component allows to locate agents by an ID. The yellow pages component makes possible to locate a group of agents that can provide a specific service.

3 BESA-ME DESIGN

The goal of the BESA-ME design is to find how to implement the BESA model in an embedded system, taking into account the BESA requirements and the constraints of this kind of platforms. The more important operating requirements of a BESA framework are: the management of the shared resources and shared memory spaces, and the multi-task operation and the concurrent treatment of events.

Real time operating systems are the proper tool to provide the services required to implement BESA-ME. In particular, the communication and synchronization of the

agent behaviors requires semaphores and messages queues with blocking sending. The concurrent operation of agents and their behaviors can be achieved by a multitask preemptive operating system kernel. As BESA-ME is a framework for the construction of embedded applications, usually implemented in micro- controllers and DSPs, it is convenient to use a Real Time Operating System (RTOS) to provide the required functionalities [5]. A BESA-ME application can be seen as a layered structure, as shown in Figure 3. Upper layers use the abstraction and services provided by the lower layers. The user application is designed using the abstraction of the BESA abstract model. The BESA-ME components are implemented as concurrent tasks using the inter-task communication mechanisms provided by the RTOS.

Finally, it is essential to adjust the BESA general model in order to fit the constraints of a practical embedded application. In the BESA-ME implementation model, some BESA elements are excluded in order to improve the performance and reduce the amount of required memory. The conditions of the guard selector mechanism are eliminated, thus reducing the control of the non-determinism implicit in an event driven system, but improving the filtering speed of events in the channel and allocating more memory to store events in the ports. In the container only the white pages directory is used, so the agents can locate an agent if they know its "alias".

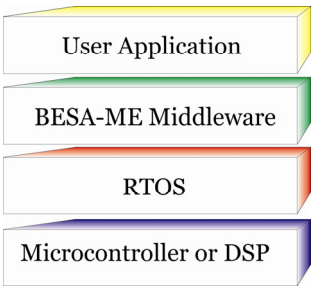


Fig. 3. Layers in a BESA-ME application.

4 Implementation

BESA-ME has been successfully implemented for the microcontroller family PIC18F and the DSP dsPIC60F. The available RTOS that were selected as candidates to support the BESA-ME implementation include the Salvo-RTOS [12], the µC/OS-II [13], the CMX-TINY [14] and FreeRTOS [15]. Though several of the studied RTOS were offered the required services, after a detailed analysis of their functionalities, the FreeRTOS™ was selected. It was chosen because it's a RTOS with a GNU license, it is well-structured and easy to use; it also supports many microprocessors, micro-controllers and DSPs.

4.1 Agent Level

The BESA agent is modeled by a data structure called stAgentBESA. It contains pointers to the agent shared state and channel, an array of pointers to the behaviors associated to the agent and the agent handler that contains the alias and the agent id.

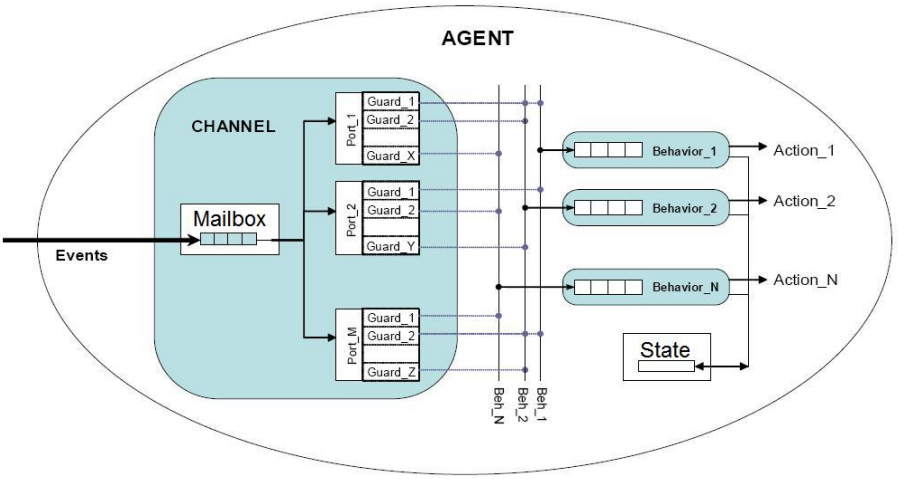


Fig. 4. BESA-ME agent level model. The boxes represent the data structures of the agent.

The Channel Task

The Channel Task has been implemented as a RTOS task, which is blocked waiting for incoming events. This waiting mechanism is implemented using a RTOS message queue service. When an event is received in the *Channel Reception Queue* this task looks for the incoming event type and compares it with the defined registered ports; if a match was found, the channel task sends a message to the behaviors interested in this event type. The same code is used to implement and create the required channel instances thanks to the parametric data structure used to represent an agent. When an agent is built, a channel task is created. A similar approach is used for the behaviors tasks. The channel parameters are contained in a data structure that includes a pointer to the channel reception queue, an array of pointers to the channel ports and some control variables.

The Behavior Task

The behaviors have been implemented as RTOS tasks, which are blocked waiting for messages sent from the channel. The messages contain the event attached data and a pointer to the adequate treatment function. The association between event types and their treatment functions is defined when the guards are created and bind to ports.

When a message is received, the adequate treatment function is executed; it receives the event data and a pointer to the agent state. The state is a shared memory that has a read/write protection mechanism built with a binary semaphore. In the treatment function the microcontroller peripherals are used, mathematic operations

are performed, the external hardware is controlled, events to other agents are sent, and the logic that implements the rationality of the agent is included.

4.2 System Level

An agent system, implemented using BESA-ME is a distributed system composed by one or more BESA containers running in one ore more physical or virtual machines (Figure 5). In BESA-ME, there is only one container running in each processing device (microcontroller or DSP). Each container has a unique ID registered in the white pages directory. When a send event service is invoked, a search for the address of the agent is performed. If the destination agent exists in the local container, the event is directly sent to the appropriated message queue. If the destination is not local, the event is sent to the other registered containers; only the container where the receiver agents exist takes into account the event and transfers it to the appropriated agent.

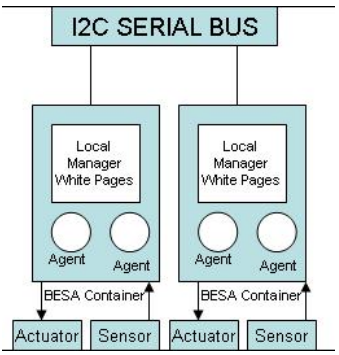


Fig. 5. BESA-ME System Layer Model.

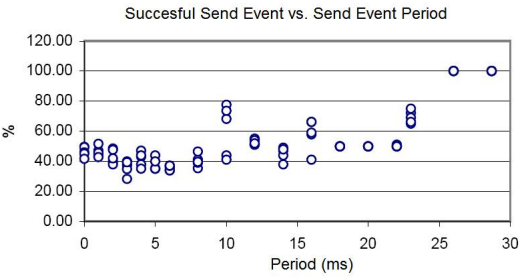


Fig. 6. Percentage of events received successfully by an external agent.

In the actual implementation of BESA-ME, the communication between containers is achieved through the use of a wired bus and the I2C low level communication protocol. The I2C protocol has been implemented in an interrupt service routine. The BESA-ME communication process is controlled with the I2C control bits at the physical layer and a BESA acknowledge to inform if the event has been taken properly by the destination agent. The information required to construct event and acknowledgment packets is written in a transmit buffer protected by a binary semaphore. The semaphore is free when the whole message has been send. Another semaphore prevents that one other local agent start an external communication before the reception of a pending acknowledgment packet.

5 Experimental Results

In order to verify that the BESA-ME execution is correct, a test protocol has been designed. The controlled independent variables include the task's stack size, the number of BESA-ME elements (agents, behaviors, guards) and the event production frequency. The measured dependent variables include the response time of the system to extern and local events, the number of errors in the inter-agents communication, and the evaluation of the general performance of the system.

The functional test was implemented in a couple of interconnected microcontrollers PIC18F8720 and PIC18F8620, with a processor frequency of 4 MHz and the I2C communication frequency of 100 KHz. This value of the intervenient variables were found by gradual increments. In the agent level the maximum number of behaviors is 11, with only one agent in the container. For one behavior the maximum number of guards is 20, with only one port (one event type). The maximum number of ports is 50. In the system level the maximum number of agents per container is 5. An extensive communication test was applied to the BESA-ME system level [16]. In summary, the results include the response time to events and the errors rate, in local and extern communication. In figure 6 can be observed how the event frequency affects the communication between agents placed in different containers.

The AgentCoop project aims to build a multirobot platform. This multirobot system is controlled using the MRCC model (MultiResolution Cooperation Control), which provides a flexible framework to built cooperative multiagent systems. In order to deal with the complexity of this application, the AgentCoop architecture has been designed using the BESA conceptual model. The robotic system is composed by a set of agents that can be deployed in a distributed environment, where high performance stand alone computers coexist with embedded processors The design of the AgentCoop platform represents a complex problem with the characteristics previously analyzed in section 1. The robots of the AgentCoop project are controlled using the dsPIC30F6012. Thus, BESA-ME has been easily migrated to work with this platform, proving the flexibility and modularity of the actual BESA-ME framework.

6 Conclusions

BESA-ME is a useful tool to solve complex problems in embedded applications. It allows to split complex objectives into simpler tasks, and to allocate them to dedicated agents. BESA-ME solve the low level synchronization problems, providing a high level abstraction model to make a more easy design and implementation of system. The use of the BESA-ME model allow the development of MAS oriented distributed applications for microcontrollers in a easy and efficient way, reducing the development time.

One of the more important BESA-ME facilities is the flexibility; if a system could not be fitted in only one microcontroller it can be distributed in two ore more processors only by changing configuration and declaration parameters of the middleware. The required external communication between micro-controllers is automatically detected and managed by the BESA-ME communication services.

A comparative evaluation of BESA-ME is projected. This evaluation will be implemented comparing the AgentCoop development model for BESA-ME and the implementation model if using only an RTOS without BESA-ME.

References

1. INTEL Corporation. “Intel® Dual-Core Processors: The first step in the multi-core revolution”, www.intel.com/technology/computing/dual-core. January 15th 2007.
2. SMARTHOME. “Getting Started”, http://www.smarthome.com/starters.html. January 13th 2007.
3. The Robotics Institute. “Distributed Robots Architecture”, Carnegie Mellon University. http://www.frc.ri.cmu.edu/projects/dira. January 12th 2007.
4. Liu, Jiming. “Multi – Agent Robotic Systems”. CRC Press, Boca Ratón, Fl. 2001.
5. Gonzalez Enrique, Avila Jamir, Bustacara César. "BESA: Behavior-Oriented, Event-Driven And Social-Based Agent Framework". PDPTA'03, Las Vegas-Usa, Csrea Press, Vol. 3, Junio 2003, PP 1033-1039. ISBN 1-892512-43-2.
6. P.De Saqui SanneS. “Conception basée modèle des systèmes temps réel et distributes”, Rapport Laas No05403, Toulouse, France. 7 Juillet 2005.
7. Roques, Pascal, “UML en action”. Editions Eyrolles, France. June 2004.
8. Verimag. “Synchrone”, http://www-verimag.imag.fr/SYNCHRONE/. October 2006.
9. Stallings, William. “Operating Systems: Internals and Design Principles“. Prentice Hall. USA. 2001.
10. Weiss, Gerhard. “Multiagent systems: A modern approach to Distributed Artificial Intelligence”. MIT Press, U.S.A 1999.
11. Foundation for Intelligent Physical Agents. “FIPA Specifications”, http://www.fipa.org, November 5th 2004.
12. PUMPKIN INC. “Salvo User Manual”, http://www.pumpkininc.com/, October 23rd 2004.
13. MICRIUM. “µC/OS-II Kernel Benefits”, http://www.ucos-ii.com/products/rtos/kernel/rtos.html, November 18th 2006.
14. CMX Systems. “TINY PLUS Real-Time Multi-Tasking Operating System for Microprocessors and icrocomputers” http://www.cmx.com/, November 18th 2006.
15. Barry, Richard. “FreeRTOS”, http://www.freertos.org, November 2004 – November 2006.
16. Florez, David, Rodrigez, Guillermo, Ortiz Juan. “BESA-ME: Arquitectura de sistemas multi-agente para microcontroladores”. Pontificia Universidad Javeriana. Bogotá, Colombia. 2005.
17. Gonzalez Enrique, Bustacara César, Avila Jamir,. "Agents For Concurrent Programming". CPA 2003, Enschende-Holanda, IOS PRESS, Septiembre 2003, PP 157-166. ISBN 1-58603-381-6.

Robustness Against Deception in Unmanned Vehicle Decision Making

William M. McEneaney[1] and Rajdeep Singh[2]

[1] Depts. of Mechanical/Aerospace Engineering and Mathematics
University of California at San Diego, La Jolla, CA 92093-0112, USA
Research partially supported by DARPA Contract NBCHC040168. and AFOSR Grant FA9550-06-1-0238
wmceneaney@ucsd.edu
[2] Integrated Systems & Solutions, Information Assurance Group, Lockheed-Martin
La Jolla, CA, USA
rajdeep.singh@lmco.com

Abstract. We are motivated by the tasking problem for UAVs in an adversarial environment. In particular, we consider the problem where, in addition to purely random noise in the observation process, the opponent may be applying deception as a means to cause us to make poor tasking choices. The standard approach would be to apply the feedback-optimal controls for the fully-observed game, to a maximum-likelihood state estimate. We find that such an approach is highly suboptimal. A second approach is through a concept taken from risk-sensitive control. For the third approach, we formulate and solve the problem directly as a partially-observed stochastic game. A chief problem with such a formulation is that the information state for the player with imperfect information is a function over the space of probability distributions (a function over a simplex), and so infinite-dimensional. However, under certain conditions, we find that the information state is finite-dimensional. Computational tractability is greatly enhanced. A simple example is considered, and the three approaches are compared. We find that the third approach is yields the best results (for such a case), although computational complexity may lead to use of the second approach on larger problems.

1 Introduction

For a discrete deterministic game, one can apply dynamic programming techniques to compute the value function (and "optimal" controls), defined over the state space. For discrete *stochastic* games, the value function is defined over the space of all possible probability distributions over the state space. Consequently, the problem is much more computationally intensive. Finally, for discrete stochastic games with imperfect observations, the problem is yet more complex, and even simple games and their information state formats become quite difficult to analyze.

We will be concerned here with a specific class of discrete stochastic games under imperfect observations. The choice of this class will be affected by both the intended application and computational feasibility considerations. The motivational application here is the military command and control (C^2) problem for air operations, with unmanned/uninhabited air vehicles (UAVs). See [2], [5], [16], [21], [28], [31], [24], [25]

for related information. This application has specific characteristics such that we will be able to construct a reasonable problem formulation which is particularly nice from the point of view of analysis and computation.

We first outline the mathematical machinery. The details of the development are discussed elsewhere due to paper length issues. After discussion of the algorithms, we apply the techniques on a seemingly simple problem in order to determine their effectiveness. We refer to the players in the game as Blue and Red, where the Blue player has imperfect observations. We compare three Blue approaches on this simple game problem. The most naive is for Blue to simply take the maximum likelihood estimate of the Red state, and to apply a feedback control at this system state. As one can easily imagine, this approach is open to exploitation by Red deception. The second Blue approach will apply a heuristic derived from the theory of Risk-Sensitive Control. This technique is more cautious in its use of observational data. The third Blue approach (a deception-robust approach) is through the direct solution of the imperfect information stochastic game. As one would expect, there is an improvement in outcome with the risk-sensitive and deception-robust approaches described herein when compared with the standard maximum likelihood/certainty equivalence approach (although there is a critical parameter in the risk-sensitive approach). On the other hand, there are significant computational requirements when using these new approaches.

2 Modeling the Game

We model the state dynamics as a discrete-time Markov chain. The state will take values in a finite set, $\mathcal{X}$. Time will be denoted by $t \in \{0, 1, 2, \ldots, T\}$. We will consider only the problem where there are exactly two players. Blue controls will take values in a finite set, U, and Red controls will take values in a finite set, W. Given Blue and Red controls, and a system state, there are probabilities of transitioning to other possible states. We let $P_{i,j}(u, w)$ denote the probability of transitioning from state i to state j in one time step given that the Blue and Red controls are $u \in U$ and $w \in W$, respectively. Also, $P(u, w)$ will denote the matrix of such transition probabilities. We must allow for feedback controls. That is, the control may be state-dependent. For technical reasons, we will find that we specifically need to consider Red feedback controls. Suppose the size of $\mathcal{X}$ is n, i.e. that there are n possible states of the system. Then we may represent a Red feedback control as $\boldsymbol{w} \in W^n$, an n-dimensional vector with components having values in W. Specifically, $\boldsymbol{w}_i = \bar{w} \in W$ implies that Red plays $\bar{w}$ if the state is i. Define matrix $\widetilde{P}(u, \boldsymbol{w})$ by

$$\widetilde{P}_{i.j}(u, \boldsymbol{w}) = P_{i,j}(u, \boldsymbol{w}_i) \qquad \forall\, i, j \in \mathcal{X}. \tag{1}$$

Let ξ_t denote the (stochastic) system state at time t. Let q_t be the vector of length n whose i^{th} component is the probability that the state is i at time t, that is the probability that $\xi_t = i$. Then if Blue plays u and Red plays $\boldsymbol{w}$, the probability propagates as

$$q_{t+1} = \widetilde{P}'(u, \boldsymbol{w}) q_t. \tag{2}$$

We suppose there is a terminal cost for the game which is incurred at terminal time, T. Let the cost for being in terminal state $\xi_T = i \in \mathcal{X}$ be $\mathcal{E}(i)$, which we will also

sometimes find convenient to represent as the i^{th} component of a vector, $\boldsymbol{\mathcal{E}}$ (where we note the abuse of notation due to use of $\mathcal{E}$ for two different objects). Suppose that at time $T-1$, the state is $\xi_{T-1} = i_0$, and that Blue plays $u_{T-1} \in U$ and Red plays $\boldsymbol{w} \in W^n$. Then, the expected cost would be $\mathbf{E}[\mathcal{E}(\xi_T)] = q_T' \boldsymbol{\mathcal{E}}$ where $q_T = \widetilde{P}'(u, \boldsymbol{w}) q_{T-1}$ with q_{T-1} being 1 at i_0 and zero in all other components.

We also need to define the observation process. We suppose that Red has perfect state knowledge, but that Blue obtains its state information through observations. Let the observations take values $y \in Y$. We will suppose that this observation process can be influenced not only by random noise, but also by the actions of both players. For instance, again in a military example, Blue may choose where to send sensing entities, and Red may choose to have some entities act stealthily while having some other entities exaggerate their visibility, for the purposes of deception. We let $R_i(y, u, w)$ be the probability that Blue observes y given that the state is i and Blue and Red employ controls u and w. We will also find it convenient to think of this as a vector indexed by $i \in \mathcal{X}$.

We suppose that at each time, $t \in \{0, 1, \ldots T-1\}$, first an observation occurs, and then the dynamics occur. We let q_t be the a priori distribution at time t, and $\widehat{q}_t$ be the a posteriori distribution. With this, the dynamics update of (2) is rewritten as

$$q_{t+1} = \widetilde{P}'(u_t, \boldsymbol{w}_t) \widehat{q}_t \tag{3}$$

with controls $u_t, \boldsymbol{w}_t$ at time t. The observation, say $y_t = y$, at time t updates q_t to $\widehat{q}_t$ via Bayes rule,

$$[\widehat{q}_t]_i = \frac{P(y_t = y \,|\xi_t = i, u, w)[q_t]_i}{\sum_{k \in \mathcal{X}} P(y_t = y \,|\xi_t = k, u, w)[q_t]_k}. \tag{4}$$

Then (3), (4) define the dynamics of the conditional probabilities.

2.1 Risk-Averse Controller Theory

In linear control systems with quadratic cost criteria, the control obtained through the *separation principle* is optimal. That is, the optimal control is obtained from the state-feedback control applied at the state given by

$$\overline{x} = \operatorname*{argmax}_i \left[q_t(i) \right].$$

A different principle, the *certainty equivalence principle*, is appropriate in robust control. We have applied a generalization of the controller that would emanate from this latter principle. This generalization allows us to tune the relative importance between the likelihood of possible states and the risk of misestimation of the state. Let us motivate the proposed approach in a little more detail.

In deterministic games under partial information, the certainty equivalence principle indicates that one should use the state-feedback optimal control corresponding to state

$$\overline{x} = \operatorname{argmax} \left[\mathcal{I}_t(x) + V_t(x) \right] \tag{5}$$

where $\mathcal{I}$ is the information state and V is the value function [13] (assuming uniqueness of the argmax of course). In this problem class, the information state is essentially the

worst case cost-so-far, and the value is the minimax cost-to-come. So, heuristically, this is roughly equivalent to taking the worst-case possibility for total cost from initial time to terminal time. (See, for instance, [20], [17], [22], [29], [30].)

The deterministic information state is very similar to the log of the observation-conditioned probability density in stochastic formulations for terminal/exit cost problems. In fact, this is exactly true for a class of linear/quadratic problems. In such problems, the $\mathcal{I}_t$ term in (5) is replaced by the log of the probability density, and a risk-sensitivity coefficient appears as well. Although we are outside of that problem class here, we nonetheless apply the same approach, but where now the correct value of this risk-sensitivity parameter is not as obvious. In particular, the risk-sensitive algorithm is as follows: Apply state-feedback control at

$$x^* = \underset{i}{\operatorname{argmax}}\{\log[\widehat{q}_t(i)] + \kappa V_t(i)\} \tag{6}$$

where $\widehat{q}$ is the probability distribution based on the conditional distribution for Blue given by (3), (4) and a stochastic model of Red control actions, and V is state-feedback stochastic game value function (c.f. [13]). Here, $\kappa \in [0, \infty)$ is a measure of risk aversion. Note that $\kappa = 0$ implies that one is employing a maximum likelihood estimate in the state- feedback control (for the game), i.e. $\operatorname{argmax}_i\{\log([\widehat{q}_t]_i)\} = \operatorname{argmax}_i\{[\widehat{q}_t]_i\}$. Note also (at least in linear-quadratic case where $\log[\widehat{q}_t]_i = \mathcal{I}_t(i)$ modulo a constant), $\kappa = 1$ corresponds to the deterministic game certainty equivalence principle [17], [20], i.e. $\operatorname{argmax}\{\mathcal{I}_t(i) + V_t(i)\}$. As $\kappa \to \infty$, this converges to an approach which always assumes the worst possible state for the system when choosing a control – regardless of observations. (See [28] for further discussion.)

2.2 Deception-Robust Controller Theory

The above approach was cautious (risk averse) when choosing the state estimate at which to apply state-feedback control. We now consider a controller which explicitly reasons about deception. This approach typically handles deception better that the risk-averse approach, but this improvement comes at a substantial computational cost. For a given, fixed computational limit, depending on the specific problem, it is not obvious which approach will be more successful.

Here we find that the truly proper information state for Red is $\mathcal{I}_t : Q(\mathcal{X}) \to \mathbf{R}$, where $Q(\mathcal{X})$ is the space of probability distributions over state space $\mathcal{X}$; $Q(\mathcal{X})$ is the simplex in $\Re^n$ such that all components are non-negative and such that the sum of the components is one. We let the initial information state be $\mathcal{I}_0(\cdot) = \phi(\cdot)$. Here, ϕ represents the initial cost to obtain and/or obfuscate initial state information. The case where this information cannot be affected by the players may be represented by a max-plus delta function. The information state at time t evaluated at probability distribution q, $\mathcal{I}_t(q)$, essentially represents the cost to the opponent to generate distribution q as the naive/Bayesian distribution in a Blue estimator. That is, through obfuscation of the initial intelligence and use of controls w_r up to time t, the propagation (3), (4) would lead to some q at time t if such w_r were known. $\mathcal{I}_t(q)$ would be the maximal (worst from Blue perspective) cost to generate q by any Red controls that would yield that

particular q at time t. Although Blue does not know the Red controls, it can nonetheless compute $\mathcal{I}_t(\cdot)$. For details on this propagation and theory, see [26].

In the case here, where the state-space is finite of size $n = \#\mathcal{X}$, $\mathcal{Q}$ is some a simplex in $\mathbb{R}^n$. Thus, $\mathcal{I}_t$ belongs to a space of functions over an $n-1$ dimensional simplex, and consequently an element of an infinite-dimensional space. However, in the cases where ϕ is either a max-plus delta function, or a piecewise-continuous function, $\mathcal{I}_t$ is finite dimensional. This is crucial to the computability of this controller. Note that in either of these cases, the complexity of $\mathcal{I}_t$ is proportional (in the worse case) to $(\#W)^t$ at the t time-step. Pruning strategies for reduction of this complexity are critical (c.f., [23]).

We now turn to the second component of the theory, computation of the state-feedback value function. In this context, our value function is a *generalized* value function in that it is a function not only of the physical state of the system, but also of what probability distribution Blue believes reflects its lack of knowledge of this true physical state. The full, generalized state of the system is now described by the true state taking values $x \in \mathcal{X}$ *and* the Blue conditional probability process taking values $q \in Q(\mathcal{X})$. We denote the terminal cost for the game as $\mathcal{E} : \mathcal{X} \to \mathbf{R}$ (where of course this does not depend on the internal conditional probability process of Blue). Thus the state-feedback value function at the terminal time is

$$V_T(x, q) = \mathcal{E}(x). \tag{7}$$

The value function at any time, $t < T$, takes the form $V_t(x, q)$. It is he above minimax expected payoff where Blue assumes that q is the "correct" distribution for x at time t, that at each time Blue will know the correct q, and that Red will know both the true physical state and this distribution, q. In particular, q will propagate according to (2), and the state will propagate stochastically, governed by (1). Loosely speaking, this generalized value function is the minimax expected payoff if Blue believes the state to be distributed by q_r at each time $r \in (t, T]$, while Red knows the true state (as well as q_r). A rigorous mathematical definition can be found in [26]. The backward dynamic program that compute V_t from V_{t+1} is as follows.

1. First, let the vector-valued function $\boldsymbol{M}_t$ be given component-wise by

$$[\boldsymbol{M}_t]_x(q, u) = \max_{\boldsymbol{w} \in W^n} \Big[\sum_{j \in \mathcal{X}} \widetilde{P}_{xj}(u, \boldsymbol{w}) V_{t+1}(j, q'(q, u, \boldsymbol{w})) \Big] \tag{8}$$

where $q'(q, u, \boldsymbol{w}) = \widetilde{P}^T(u, \boldsymbol{w})$ and the optimal $\boldsymbol{w}$ is

$$\boldsymbol{w}_t^0 = \boldsymbol{w}_t^0(x, q, u) = \operatorname*{argmax}_{\boldsymbol{w} \in W^n} \Big\{ \sum_{j \in \mathcal{X}} \widetilde{P}_{xj}(u, \boldsymbol{w}) V_{t+1}(j, q'(q, u, \boldsymbol{w})) \Big\}.$$

2. Then define L_t as

$$L_t(q, u) = q' \boldsymbol{M}_t(q, u), \tag{9}$$

and note that the optimal u is $u_t^0(q) = \operatorname{argmin}_{u \in U} L_t(q, u)$.

(10)

3. With this, one obtains the next iterate from

$$V_t(x,q) = \sum_{j\in\mathcal{X}} \widetilde{P}_{xj}(u_t^0, \boldsymbol{w}_t^0) V_{t+1}(j, q'(q, u_t^0, \boldsymbol{w}_t^0)) = [\boldsymbol{M}_t]_x(q, u_t^0)$$

and the best achievable expected result from the Blue perspective is

$$V_t^1(q) = q'\boldsymbol{M}_t(q, u_t^0). \tag{11}$$

Consequently, for each $t \in \{0, 1, \ldots, T\}$ and each $x \in \mathcal{X}$, $V_t(x, \cdot)$ is a piecewise constant function over simplex $Q(\mathcal{X})$. Due to this piecewise constant nature, propagation is relatively straight-forward (more specifically, it is finite-dimensional in contradistinction to the general case).

The remaining component of the computation of the control is now discussed. This is typically performed via the use of the certainty equivalence principle (cf. [1], [17]), and we employ the principle here as well. To simplify notation, note that by (9) and (8), for any u,

$$L_t(q,u) = \mathbf{E}_q\Big[\max_{\boldsymbol{w}\in W^n} \sum_{j\in\mathcal{X}} \widetilde{P}_{Xj}(u, \boldsymbol{w}) V_{t+1}(j, q'(q, u, \boldsymbol{w}))\Big].$$

Let us hypothesize that the optimal control for Blue is

$$u_t^m \doteq \operatorname*{argmin}_{u\in U}\left[\max_{q\in Q(\mathcal{X})} \{\mathcal{I}_t(q) + L_t(q,u)\}\right]. \tag{12}$$

In order to obtain the robustness/certainty Equivalence result below, it is sufficient to make the following Saddle Point Assumption. We assume that for all t,

$$\sup_{q_{\bar{t}}\in Q_t} \min_{u\in U}\Big[I_{\bar{t}}(q_{\bar{t}}) + L_{\bar{t}}(q_{\bar{t}}, u)\Big] = \min_{u\in U} \sup_{q_{\bar{t}}\in Q_t}\Big[I_{\bar{t}}(q_{\bar{t}}) + L_{\bar{t}}(q_{\bar{t}}, u)\Big]. \tag{A-SP}$$

This type of assumption is typical in game theory. Although it is difficult to verify for a given problem, the alternative is a theory that cannot be translated into a useful result. Finally, after some work [26], one obtains the robustness result:

Theorem 1. *Let $\bar{t} \in \{0, T-1\}$. Let $\mathcal{I}_0$, $u_{[0,\bar{t}-1]}$ and $y_{[0,\bar{t}-1]}$ be given. Let the Blue control choice, $u_{\bar{t}}^m$, given by (12) be a strict minimizer. Suppose Saddle Point Assumption (A-SP) holds. Then, given any Blue strategy, $\lambda_{[\bar{t},T-1]}$ such that $\lambda_{\bar{t}}[y_\cdot] \neq u_{\bar{t}}^m$, there exists $\varepsilon > 0$, $q_{\bar{t}}^\varepsilon$ and $\boldsymbol{w}_{[\bar{t},T-1]}^\varepsilon$ such that*

$$\sup_{q\in Q_{\bar{t}}} \{\mathcal{I}_{\bar{t}}(q) + L_{\bar{t}}(q, u_{\bar{t}}^m)\} = Z_{\bar{t}} \leq \mathcal{I}_{\bar{t}}(q_{\bar{t}}^\varepsilon) + \mathbf{E}_{X\sim q_{\bar{t}}^\varepsilon}\Big\{\mathbf{E}[\mathcal{E}(X_T^\varepsilon) \mid X_{\bar{t}}^\varepsilon = X]\Big\} - \varepsilon$$

where X^ε denotes the process propagated with control strategies $\lambda_{[\bar{t},T-1]}$ and $\boldsymbol{w}_{[\bar{t},T-1]}^\varepsilon$.

3 A Seemingly Simple Game

We now apply the above technology to an example problem in Command and Control for UCAVs. This game will seem to be quite simple at first. However, once one introduces the partial information and deception components, determination of the best (or even nearly best) strategy becomes quite far from obvious.

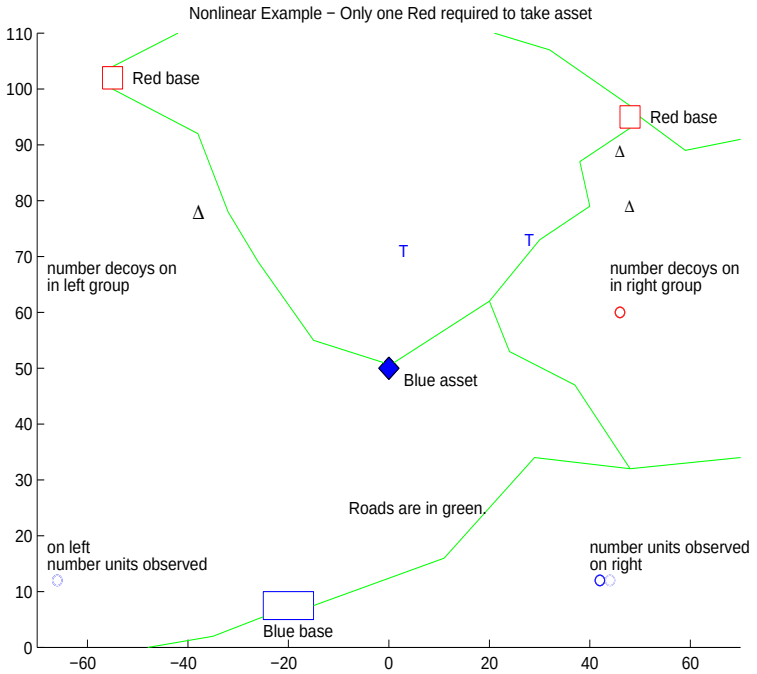


Fig. 1. Snapshot of Gameboard.

In this game the Red player has four ground entities (say, tanks) and the Blue player has two UCAVs. The objective of Red player is to capture the high-value Blue assets by moving at least one non-decoy Red entity to a Blue asset location by the terminal time, T. Red can use stealth and decoys to obscure the direction from which the attack will occur, while the Blue player uses the UCAVs to destroy the moving Red entities. Red entities do not have any attrition capability against the Blue UCAVs. Blue UCAVs require at least two time steps to travel from one route to the other.

The simulation snapshot in Figure 1, is taken after time step 2, from the graphic for a MATLAB simulation that runs the example game. Red is moving its currently surviving three entities (depicted as triangles) downward, while Blue is attempting to prevent any Red entities from reaching the Blue asset through use of its UCAVs (depicted as blue T's). Red is currently employing a decoy on the right, while using stealth on the left.

Winning and losing are measured in terms of the total cost at the terminal time. The cost at terminal time is computed as follows: each Red surviving entitiy costs Blue 1 point and if Blue loses the high-value asset, it costs Blue 20 points.

4 Comparison of the Approaches

Let us briefly foray into a comparative study between the naive approach (i.e., feedback on maximum-likelihood state), the risk-averse algorithm and the deception-robust approach for Blue. The critical component of the risk-averse approach is the choice of the risk level, κ. For the example studied in this chapter we vary κ between 0 and 10 to demonstrate the nature of the risk-averse approach in general. Firstly, for the case $\kappa = 0$, we have the risk-averse approach equivalent to the naive approach; apply the state-feedback control at the MLS estimate. As κ increases we expect the approach to achieve a lower cost for Blue, since it is taking into account the expected future cost $V(X_t)$ (as a risk-sensitive measure). Note however that in the adversarial environment the effect of the Red player's control on the Blue player's observations has more complex consequences than that of random noise. As shown in the Figure 2, the risk-averse approach gets the best cost for Blue at κ between 0.5 and 0.6 (note again that this choice will be problem specific). As κ increases beyond this point, the expected cost begins increasing, and has a horizontal asymptote which corresponds to a Blue controller which ignores all the observations and assumes the worst-case possible Red configuration.

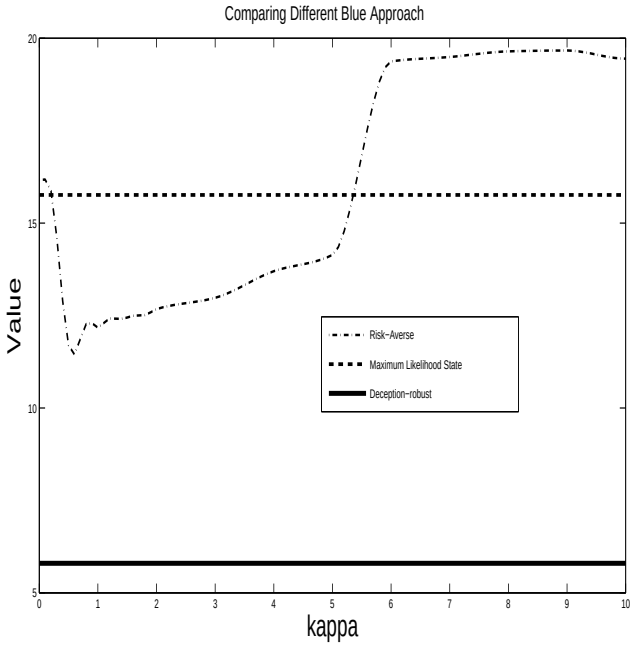


Fig. 2. Comparison of Approaches.

The bumpiness in the results is due to the sampling error (8000 Monte Carlo runs were used for each data point in the plot.) Also note that for large κ, the risk-averse approach does worse than the naive approach. For this specific example, the risk-averse

approach does not achieve the same low cost as achieved by using the deception-robust approach.

References

1. T. Basar and P. Bernhard, *H_∞ –Optimal Control and Related Minimax Design Problems*, Birkhäuser (1991).
2. D.P. Bertsekas, D.A. Castañon, M. Curry and D. Logan, "Adaptive Multi-platform Scheduling in a Risky Environment", Advances in Enterprise Control Symp. Proc., (1999), DARPA–ISO, 121–128.
3. P. Bernhard, A.-L. Colomb, G.P. Papavassilopoulos, "Rabbit and Hunter Game: Two Discrete Stochastic Formulations", Comput. Math. Applic., Vol. 13 (1987), 205–225.
4. T. Basar and G.J. Olsder, *Dynamic Noncooperative Game Theory*, Classics in Applied Mathematics Series, SIAM (1999), Originally pub. Academic Press (1982).
5. J.B. Cruz, M.A. Simaan, et al., "Modeling and Control of Military Operations Against Adversarial Control", Proc. 39th IEEE CDC, Sydney (2000), 2581–2586.
6. R.J. Elliott and N.J. Kalton, "The existence of value in differential games", Memoirs of the Amer. Math. Society, 126 (1972).
7. W.H. Fleming, "Deterministic nonlinear filtering", Annali Scuola Normale Superiore Pisa, Cl. Scienze Fisiche e Matematiche, Ser. IV, 25 (1997), 435–454.
8. W.H. Fleming, "The convergence problem for differential games II", Contributions to the Theory of Games, **5**, Princeton Univ. Press (1964).
9. W.H. Fleming and W.M. McEneaney, "Robust limits of risk sensitive nonlinear filters", Math. Control, Signals and Systems 14 (2001), 109–142.
10. W.H. Fleming and W.M. McEneaney, "Risk sensitive control on an infinite time horizon", SIAM J. Control and Optim., Vol. 33, No. 6 (1995) 1881–1915.
11. W.H. Fleming and W.M. McEneaney, "Risk–sensitive control with ergodic cost criteria", Proceedings 31^{st} IEEE Conf. on Dec. and Control, (1992).
12. W.H. Fleming and W. M. McEneaney, "Risk–sensitive optimal control and differential games", (Proceedings of the Stochastic Theory and Adaptive Controls Workshop) Springer Lecture Notes in Control and Information Sciences 184, Springer–Verlag (1992).
13. W.H. Fleming and H.M. Soner, *Controlled Markov Processes and Viscosity Solutions*, Springer-Verlag, New York, 1992.
14. W.H. Fleming and P.E. Souganidis, "On the existence of value functions of two–player, zero–sum stochastic differential games", Indiana Univ. Math. Journal, **38** (1989), 293–314.
15. A. Friedman, *Differential Games*, Wiley, New York, 1971.
16. D. Ghose, M. Krichman, J.L. Speyer and J.S. Shamma, "Game Theoretic Campaign Modeling and Analysis", Proc. 39th IEEE CDC, Sydney (2000), 2556–2561.
17. J.W. Helton and M.R. James, *Extending H_∞ Control to Nonlinear Systems: Control of Nonlinear Systems to Achieve Performance Objectives*, SIAM 1999.
18. D.H. Jacobson, "Optimal stochastic linear systems with exponential criteria and their relation to deterministic differential games", IEEE Trans. Automat. Control, 18 (1973), 124–131.
19. M.R. James, "Asymptotic analysis of non–linear stochastic risk–sensitive control and differential games", Math. Control Signals Systems, 5 (1992), pp. 401–417.
20. M. R. James and J. S. Baras, "Partially observed differential games, infinite dimensional HJI equations, and nonlinear H_∞ control", SIAM J. Control and Optim., 34 (1996), 1342–1364.
21. J. Jelinek and D. Godbole, "Model Predictive Control of Military Operations", Proc. 39th IEEE CDC, Sydney (2000), 2562–2567.

22. M.R. James and S. Yuliar, “A nonlinear partially observed differential game with a finite-dimensional information state”, Systems and Control Letters, 26, (1995), 137–145.
23. W.M. McEneaney and R. Singh, “Robustness against deception”, *Adversarial Reasoning: Computational Approaches to Reading the Opponent’s Mind*, Chapman and Hall/CRC, New York (2007), 167–208.
24. W.M. McEneaney and R. Singh, “Deception in Autonomous Vehicle Decision Making in an Adversarial Environment”, Proc. AIAA conf. on Guidance Navigation and Control, (2005).
25. W.M. McEneaney and R. Singh, “Unmanned Vehicle Operations under Imperfect Information in an Adversarial Environment ”, Proc. AIAA conf. on Guidance Navigation and Control, (2004).
26. W.M. McEneaney, “Some Classes of Imperfect Information Finite State-Space Stochastic Games with Finite-Dimensional Solutions”, Applied Math. and Optim., Vol. 50 (2004), 87–118.
27. W.M. McEneaney, “A Class of Reasonably Tractable Partially Observed Discrete Stochastic Games”, Proc. 41st IEEE CDC, Las Vegas (2002).
28. W.M. McEneaney, B.G. Fitzpatrick and I.G. Lauko, “Stochastic Game Approach to Air Operations”, IEEE Trans. on Aerospace and Electronic Systems, Vol. 40 (2004), 1191–1216.
29. W.M. McEneaney, “Robust/game–theoretic methods in filtering and estimation”, First Symposium on Advances in Enterprise Control, San Diego (1999), 1–9.
30. W.M. McEneaney, “Robust/H_∞ filtering for nonlinear systems”, Systems and Control Letters, 33 (1998), 315–325.
31. W.M. McEneaney and K. Ito, “Stochastic Games and Inverse Lyapunov Methods in Air Operations”, Proc. 39th IEEE CDC, Sydney (2000), 2568–2573.
32. G.J. Olsder and G.P. Papavassilopoulos, “About When to Use a Searchlight”, J. of Math. Analysis and Applics., Vol. 136 (1988), 466–478.
33. T. Runolfsson, “Risk–sensitive control of Markov chains and differential games”, Proceedings of the 32nd IEEE Conference on Decision and Control, 1993.
34. P. Whittle, “Risk–sensitive linear/quadratic/Gaussian control”, Adv. Appl. Prob., 13 (1981), 764–777.

Ultrasound Sensor Array for Robust Location

José N. Vieira[1], Sérgio I. Lopes[2], Carlos A. C. Bastos[1] and Pedro N. Fonseca[1]

[1] Departamento de Electrónica, Telecomunicações e Informática / IEETA
Universidade de Aveiro 3810-193 Aveiro, Portugal
{vieira, cbastos, pf}@det.ua.pt
http://www.ieeta.pt
[2] {sergio.lopes}@ieeta.pt

Abstract. In this paper we present a system for localizing a team of soccer robots using ultrasound. The proposed system uses chirp signals to obtain a better signal to noise ratio with good time resolution and improved interference immunity. An array of four ultrasonic sensors is used to obtain spatial diversity and reduce the localization error. An efficient DSP algorithm for base-band conversion and decimation of the received pulses is also presented. The proposed system based on the TI DSP 2812 is very efficient and allows the localization of the robots up to 8 meters with an angular error with a maximum standard deviation of 2°.

1 Introduction

Ultrasonic Sensors (USS) systems are widely used in robotics for obstacle localization and mutual robot localization [1], [2]. Most of the commercial available ultrasonic systems use a sinusoidal pulse to measure the Time of Flight (TOF). Usually these systems perform badly in the presence of interferences and various authors have presented more complex alternatives using different types of modulation of the emitted ultrasonic pulse [3] and [4].

This paper describes an ongoing work to build a localization system, using USS for a team of soccer robots. The main achievements of this work are the efficient implementation of the baseband converter of the received bandpass chirp pulses, and the array with four sensors using an algorithm that integrates several measurements in time and space resulting in a small error for distances up to 8 meters.

The specifications for this system require that it should be able to locate each of the mobile robots (MR) with an accuracy better than ± 0.5 meters for distances up to 8 meters.

Since the goalkeeper has reduced mobility and its position can easily be determined by the vision location system, all the positions of the other robots are referenced to its position. The goalkeeper has an array of four aligned sensors at 20cm from each other as can be seen in figure 1. Each field MR has four emitters and receivers equally spaced over a circle and working like a transponder. The measurement of the position of each MR is performed in the following way: the goalkeepere emits an ultrasound start signal, then each MR answer after $n \times 50ms$ where n is the MR number ID. This guarantees the time interleaving of the answers. The goalkeeper evaluates the distance to each robot n by

$$d_n = c(T_n - n \times 50ms) = cTOF_n, \tag{1}$$

where T is the total measured time between the start signal and the received signal from the MR, TOF is the time of flight and c the sound speed.

2 Localization System

A previous version of the localization system had only two receivers located 20cm apart. This system was very sensible to small errors in the TOF measurements leading to an unusable system [5]. We solved this problem by acting on various aspects of the system. Firstly, we increased the applied voltage to the transmitter from 8 to 16Volts. Then we built an array with four aligned sensors spaced 20cm from each other to obtain spatial diversity and as a consequence increased stability in position evaluation. Finally, we improved the algorithm that evaluates the (x_m, y_m) by integrating several sets of four measures T_i to get an extra gain in the position stability.

The digital signal processing of the localization system is carried out by a DSP2812. If all the digital signal processing was performed at the sampling frequency of 160kHz, the DSP 2812 would not have enough processing power to calculate the TOF for the four channels. In order to circumvent this limitation, we implemented a baseband converter that outputs the decimated quadrature and phase components of the input signal. The baseband converter was implemented directly in DSP2812 assembly language and it uses about 50% of the available DSP processing power. The baseband converter reduces the sampling frequency by a factor of 32, from 160kHz to 5kHz. The processing of the converter output is much less demanding on the processing power and was implemented in C.

2.1 Time of Flight Measurement

The transmitted chirp pulse is generated by sampling the signal

$$c(t) = h(t)\cos\left(\omega_1 t + \beta t^2\right), \qquad t \in [0 \ldots T],$$

with $\beta = (\omega_2 - \omega_1)/(2T)$, where ω_1 and ω_2 are the initial and final frequencies of the chirp, T is the duration of the pulse and $h(t)$ is a Hamming window. The window $h(t)$ is used to reduce the side lobes that appear on the autocorrelation of the chirp. The carrier frequency is defined as $\omega_c = (\omega_1 + \omega_2)/2$.

Figure 2 shows the chirp autocorrelation with and without the Hamming window. The reduction of the sidelobes is important to avoid false peak detection.

Baseband converter For bandpass signals, the Nyquist theorem states that a signal has to be sampled at a frequency not less than twice the bandwidth of the signal. Since the bandwidth of $c(t)$ is typically 2kHz (in our system) it is possible to reduce the sampling frequency to a much lower value (5kHz) by performing a bandpass to lowpass transformation (see figure 3). This technique is well known and widely used on radar and ultrasound sensing [6]. Several methods are available to perform this conversion e.g. [7].

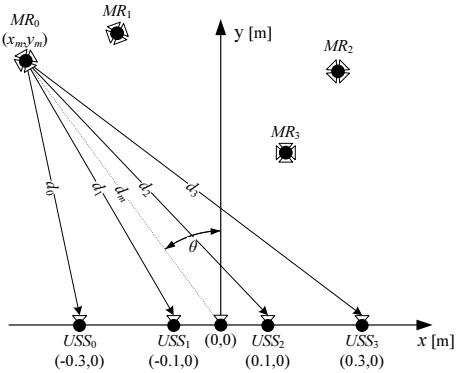


Fig. 1. Locating the MR with the four sensor USS_i array and the goalkeeper emitter (located at the origin).

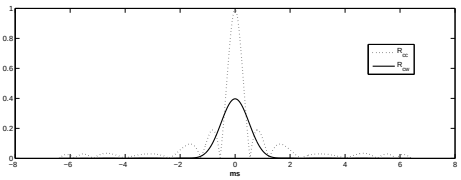


Fig. 2. Cross correlation of the chirp (dotted line) and the chirp multiplied by a Hamming window (solid line).

In this system we used low cost ultrasonic transducers from Murata that have a large aperture (MA40S4R and MA40S4S), essential to cover all the field. Their bandpass frequency response is centered at 40kHz with a useful bandwidth of 2kHz. The emitted pulses are chirps ranging from 39 to 41kHz with a duration of 6.4ms. As the baseband converted chirp has a bandwidth of only 1kHz the signal is decimated by 32 resulting in a sampling frequency of 5kHz which is enough to represent the signal.

Figure 4 shows the structured of the baseband converter as implemented in this work. We managed to simplify the calculations needed to perform the baseband conversion by specifying an integer relation between the sampling frequency and the carrier frequency and by integrating the modulators within the filters structure. The filter was implemented using a polyphase decomposition [8]. In order to calculate both outputs x_i and x_q, the system only has to perform N/D product-accumulation operations for each input signal sample, where N is the number of coefficients of the filter and D the decimation factor. As the filter $H(z)$ has $N = 256$ coefficients and the decimation factor is $D = 32$ the system only has to perform 8 mult/adds for each filter phase $H_k(z)$, with

$$H(z) = \sum_{k=0}^{D-1} z^{-k} H_k(z^D).$$

The first simplification on the system is achieved by making the sampling frequency four times greater than the carrier frequency (40kHz). This results in the following sequences at the output of the modulators

$$\cos\left(2\pi\frac{f_c}{f_s}n\right) = \{1, 0, -1, 0, \ldots\}$$
$$\sin\left(2\pi\frac{f_c}{f_s}n\right) = \{0, 1, 0, -1, \ldots\}$$

with $n \in \mathbb{Z}$. After the multipliers we have the following signals

$$x(n)\cos\left(2\pi\frac{f_c}{f_s}n\right) = \{x(0), 0, -x(2), 0, \ldots\} \tag{2}$$
$$x(n)\sin\left(2\pi\frac{f_c}{f_s}n\right) = \{0, x(1), 0, -x(3), \ldots\}\,. \tag{3}$$

We can see that each lowpass filter only has to process half the samples because the other half is zero. Using a polyphase decomposition of the lowpass filters we can move the decimator from the output of the filters to within the filter structure.

Finally, as the signals at the input of the filters have half of the samples equal to zero and noting that both lowpass filters are equal, we can add the two signals from (2) to obtain the signal

$$x(n)\left[\cos(\omega_c n) + \sin(\omega_c n)\right] = \\ = \{x(0), x(1), -x(2), -x(3), \ldots\}\,.$$

If the decimation factor is even we can decompose the decimators in the polyphase filter structure as shown in figure 4 where the input signal is separated in two different phases. The phase corresponding to the signal from (2) will be processed by the even filter phases, while the signal from (3) will be processed by the odd filter phases. At the output of the filter we have two summations, one for each filter phase obtaining the two outputs, the in-phase x_i and the quadrature x_q.

This compact structure allowed the optimization of the assembly code by minimizing the need of pointer manipulation.

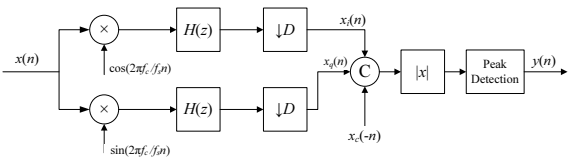


Fig. 3. Block diagram of the base band converter.

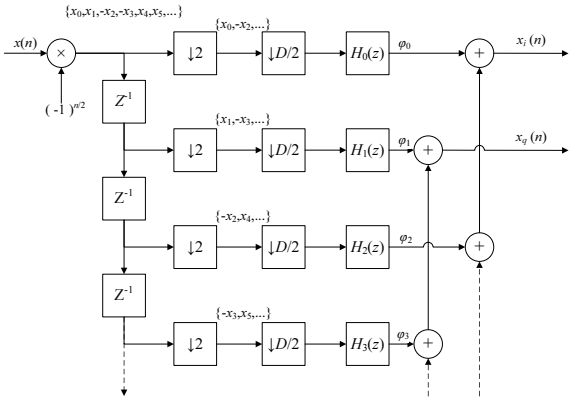


Fig. 4. Equivalent baseband converter implemented in an efficient way.

Envelope Interpolation The envelope of the correlator output signal needs to be computed in order to determine the time position of its peak. Since the sampling period of the signal is $1/5000 = 0.2$ms, the system cannot directly discriminate time differences smaller than this limit. In order to improve the time resolution of the peak detection we followed the [7] approach and interpolated the envelope of the decimated signal using quadratic interpolation.

It is well known that there is a unique quadratic function that passes through any three points. Moreover, the interpolating polynomial of degree $N-1$ through the points $y_1 = y(t_1), y_2 = y(t_2), y_3 = y(t_3)$ is given explicitly by Lagrange's classical formula, presented in equation (4) for $N = 3$,

$$\begin{aligned} y(t) =& \frac{(t-t_2)(t-t_3)}{(t_1-t_2)(t_1-t_3)} y_1 + \\ &+ \frac{(t-t_1)(t-t_3)}{(t_2-t_1)(t_2-t_3)} y_2 + \\ &+ \frac{(t-t_1)(t-t_2)}{(t_3-t_1)(t_3-t_2)} y_3. \end{aligned} \tag{4}$$

From the previous equation we can get the maximum of $y(t)$ by finding the value t that nulls the first derivative of $y(t)$. This value is given by

$$t = \frac{t_3(k_1+k_2) + t_2(k_1+k_3) + t_1(k_2+k_3)}{2(k_1+k_2+k_3)}$$

where

$$k_1 = \frac{y_1}{(t_1-t_2)(t_1-t_3)}$$

$$k_2 = \frac{y_2}{(t_2-t_1)(t_2-t_3)}$$

$$k_3 = \frac{y_3}{(t_3 - t_1)(t_3 - t_2)}$$

The TOF is obtained by subtracting, from t, all the delays introduced by the hardware, the software and the time multiplexing.

2.2 Position Estimation via Data Fusion

The position estimation is based on the TOF data fusion method [9], by combining estimates of the TOF of the MR signal arriving at the four sensors of the Goalkeaper (GK). As we will show, the problem can be solved by linear regression of several measures from the four sensors. Considering that at a room temperature of 22° the speed of sound is $c = 344$ m/s [10], the distance between the MR and each of the elements of the USS array is given by equation (1). By solving the following equation for all sensors:

$$d_i^2 = (x_i - x_m)^2 + (y_i - y_m)^2, \tag{5}$$

where $i = 0, 1, 2, 3$ is the number of the USS, we can estimate the coordinates (x_m, y_m) of the MR. The x coordinates of the four sensors of the array USS are given by

$$x_0 = -0.3,\ x_1 = -0.1,\ x_2 = 0.1,\ x_3 = 0.3,$$

where the distances are in meters. The y coordinate is zero for all sensors.

This simplifies equation 5 which can be rewritten as the following distance equations,

$$d_i^2 = (x_i - x_m)^2 + y_m^2 \tag{6}$$

Subtracting the (6) for $i = 0$, from the same equation for $i = [1 \dots 3]$ we get,

$$\begin{aligned} d_1^2 - d_0^2 &= x_1^2 - x_0^2 - (2x_1 - 2x_0)x_m \\ d_2^2 - d_0^2 &= x_2^2 - x_0^2 - (2x_2 - 2x_0)x_m \\ d_3^2 - d_0^2 &= x_3^2 - x_0^2 - (2x_3 - 2x_0)x_m \end{aligned} \tag{7}$$

By rearranging terms, the above three equations can be written in matrix form as

$$\begin{bmatrix} x_1 - x_0 \\ x_2 - x_0 \\ x_3 - x_0 \end{bmatrix} \begin{bmatrix} x_m \end{bmatrix} = \frac{1}{2} \begin{bmatrix} x_1^2 - x_0^2 - d_1^2 + d_0^2 \\ x_2^2 - x_0^2 - d_2^2 + d_0^2 \\ x_3^2 - x_0^2 - d_3^2 + d_0^2 \end{bmatrix} \tag{8}$$

which can be rewritten as

$$Ax_m = b, \tag{9}$$

where

$$A = \begin{bmatrix} x_1 - x_0 \\ x_2 - x_0 \\ x_3 - x_0 \end{bmatrix}, \quad b = \frac{1}{2} \begin{bmatrix} x_1^2 - x_0^2 - d_1^2 + d_0^2 \\ x_2^2 - x_0^2 - d_2^2 + d_0^2 \\ x_3^2 - x_0^2 - d_3^2 + d_0^2 \end{bmatrix} \tag{10}$$

The solution is given by 11 and is the value that minimises the mean quadratic error. This equation is implemented directly in the DSP, since for every measure A and b are constants.

$$x_m = (A^T A)^{-1} A^T b \tag{11}$$

The y coordinate can be estimated by averaging the four values of y_m,

$$y_m = \sqrt{d_i^2 - (x_i - x_m^2)} \tag{12}$$

which can be approximated by,

$$y_m = \sqrt{d_m^2 - x_m^2}, \tag{13}$$

where d_m is the average of the four $d_i's$. For distances above one meter the error produced by this simplification is less than 3%.

We can use temporal averaging to reduce the robot position estimation error. If we combine N measures then solving the following system of equations

$$\begin{bmatrix} A_0 \\ A_1 \\ \vdots \\ A_{N-1} \end{bmatrix} \begin{bmatrix} x_m \end{bmatrix} = \begin{bmatrix} b_0 \\ b_1 \\ \vdots \\ b_{N-1} \end{bmatrix} \tag{14}$$

produces a position estimation with less error at the expense of dynamic system response to the movement of the robot. To obtain the localization results shown in this work we used $N = 8$.

3 Hardware Prototype

We chose the TMS320F2812 DSP from Texas Instruments to perform the signal processing tasks. This DSP has 16 analog inputs sampled by a high speed ADC. Figure 5 shows a block diagram with the architecture of the acquisition system. The USS are mounted on a circuit board that performs a pre-amplification (gain=30) of the ultrasonic signal followed by bandpass filtering.

The amplitude of the received signal varies with the inverse of the distance between the transmitter and the receiver. To adjust the amplitude of the received signal to the range of the ADC we built a Programable Gain Amplifier (PGA) for each channel. The PGA is software controlled by the DSP.

As we intend to use the system to test different waveforms for the emitted signals we used an eight channel 12 bits DAC. The output amplifier connected to the ultrasonic transmitter delivers 16Vpp.

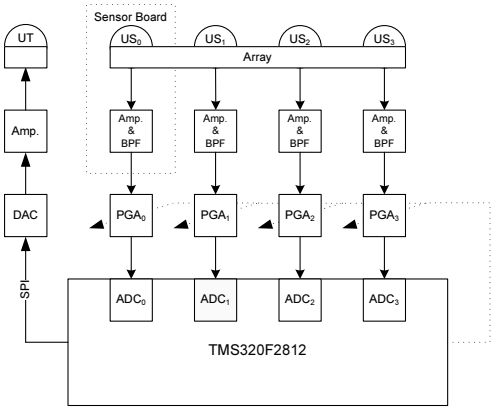


Fig. 5. The goalkeper has an ultrasonic transmitter (UT) to send the broadcast signal and an USS array with four receivers (USS_0 to USS_3). The other robots have four pairs of transmitters/receivers equally spaced around a circle.

Table 1. Anechoic chamber results: θ_r - real angle, θ_m - measured angle, $||\theta_r - \theta_m||$ - absolute error of the measured angle and σ_{θ_m} - standard deviation for each angle.

θ_r	θ_m	$\|\|\theta_r - \theta_m\|\|$	σ_{θ_m}
-90°	-83.63°	6.37°	1.90°
-80°	-69.31°	10.69°	0.21°
-70°	-68.55°	1.45°	0.37°
-60°	-62.27°	2.27°	0.40°
-50°	-36.11°	13.89°	0.22°
-40°	-42.72°	2.72°	1.20°
-30°	-34.95°	4.95°	0.14°
-20°	-19.15°	0.85°	0.31°
-10°	-8.73°	1.27°	0.24°
0°	0.36°	0.36°	0.24°
10°	9.53°	0.47°	0.50°
20°	24.54°	4.54°	0.43°
30°	43.11°	13.11°	0.88°
40°	39.45°	0.55°	0.23°
50°	42.59°	7.41°	0.18°
60°	47.49°	12.51°	0.79°
70°	69.31°	0.69°	0.21°
80°	70.66°	9.34°	0.37°
90°	83.86°	6.14°	2.32°

4 Experimental Results

Anechoic Chamber Tests. To gauge the localization system we tested it on two different situations, in an anechoic chamber and in a robot soccer field. In the anechoic chamber we only tested the angle measurement in the -90° $\leq 0 \leq$ 90° range using 10° intervals with the transmitter at a distance of 7.5m. For each angle we took 5 measures. The results of this test are shown in table 1.

In this experiment we observed that for angles between 30°and 60°the system was very sensitive to small angle variations and that the received signal envelope was double peaked due to multipath interference.

Robot Soccer Field Tests. We also performed localization tests in a robotics soccer field. From this tests we got values for the absolute error position in a 1m grid for half

the field from 0 to -3m in the x coordinate and from 1 to 8m in the y coordinate. For each position 50 measures were taken allowing the calculation of some statistical parameters such as mean and the standard deviation of the measured distance and angle.

A $X-Y$ plot with the test measurements and the estimated mean positions is shown in figure 6.

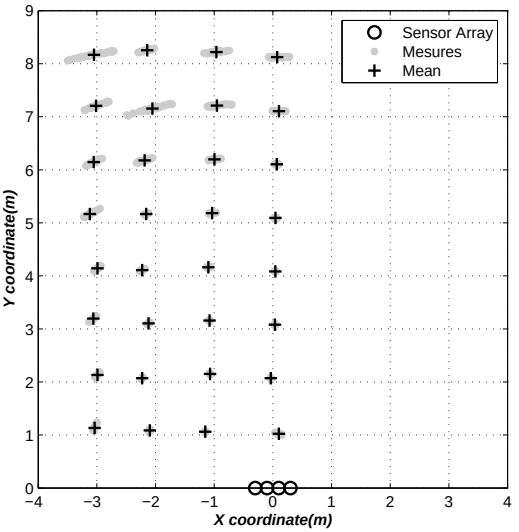


Fig. 6. Position measures obtained in the soccer field with the USS array.

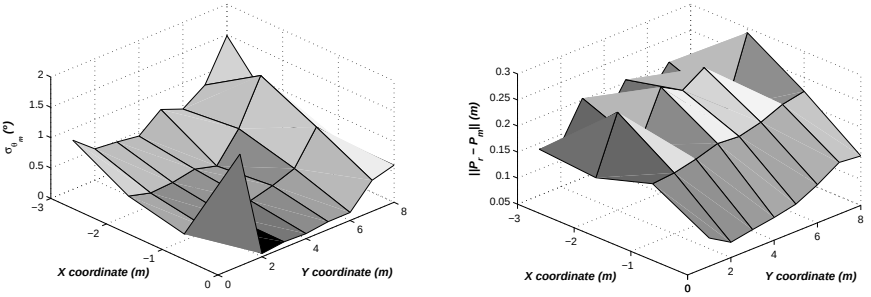
The error associated with the polar coordinates, distance and angle, of a position test are presented in figures 7 a) and b). The distance has a maximum standard deviation $\sigma_{d_m} = 1$cm and the angle has a maximum standard deviation of $\sigma_{\theta_m} = 1,84°$.

From the results shown in figure 6 it is observed that as the distance increases, the variability of the measured position also increases. This can be explained by the degradation of the signal to noise ratio of the received signals as the distance increases.

5 Conclusion

In this work we presented a complete system to localize soccer robots. An ultrasonic sensor array combined with time data fusion, reduced the standard variance of the angle measurements from 10° to 2°.

To improve the signal to noise ratio of the received signal (for large distances) the system transmitted chirp pulses with a duration of 6.4ms. To be able to process the four receiver channels simultaneously with a low cost DSP we implemented an efficient baseband converter.



(a) Angle (b) Position

Fig. 7. Standard angle error and absolute position error.

References

1. Moravec, H.P., Elfes, A.: High resolution maps from wide angle sonar. In: Proceedings IEEE International Conference on Robotics and Automation, Washington D. C., IEEE (1985) 116–121
2. Elfes, A.: Sonar-based real world mapping and navigation. IEEE Transactions on Robotics and Automation **4** (1987) 249–265
3. Sabatini, A.M., Spinielli, E.: Correlation techniques for digital time-of-flight measurement by airborne ultrasonic rangefinders. In: Proceendings of the IROS'94, Munich, Germany (1994)
4. Jrg, K.W., Berg, M.: Sophisticated mobile robot sonar sensing with pseudo-random codes. Robotics and Autonomous Systems **25** (1998) 241–251
5. Vieira, J.M., Lopes, Srgio, I., Bastos, C.A.C., Fonseca, P.N.: Sistema de localizao utilizando ultra-sons. In: Robtica 2006, Guimares, Portugal (2006)
6. Cook, C.E., Bernfeld, M.: Radar Signals - An Introduction to Theory and Application. Artech House, Norwood (1993)
7. Sabatini, A.M., Rocchi, A.: Sampled baseband correlators for in-air ultrasonic rangefinders. IEEE Transactions on Industrial Electronics **45** (1998) 341–350
8. Vaidyanathan, P.P.: Multirate Systems and Filter Banks. 1 edn. Signal Processing. Prentice-Hall, New Jersey (1993)
9. Sayed, A.H., Tarighat, A., Khajehnouri, N.: Network-based wireless location. IEEE Signal Processing Magazine **22** (2005) 24–40
10. Kočiš, S., Figura, Z.: Ultrasonic Measurements and Technologies. Sensors Physics and Technology. Chapman & Hall, London, UK (1996)

POSTERS

An Elementary Communication Framework for Open Co-operative RoboCup Soccer Teams

Luís Mota[1,2] and Luís Paulo Reis[2]

[1] DCTI-Instituto Superior de Ciências do Trabalho e da Empresa, Portugal
luis.mota@iscte.pt
[2] LIACC-Faculdade de Engenharia da Universidade do Porto, Portugal
lpreis@fe.up.pt

Abstract. One of the present day challenges in RoboCup is the development of Open Co-operative teams, where different research labs join efforts to build a common team. Such teams bring together robots with heterogeneous hardware, architectures and control software, which hinders straightforward co-operation. The robots in these teams might co-operate through a-priori strategic knowledge and structured communication during the game. This paper presents the kernel of a communication framework, defining a robotic soccer vocabulary, as well as rules to manage communication.

1 Introduction

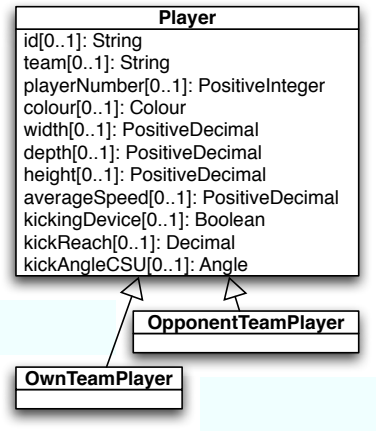


Fig. 1. Player definition.

RoboCup[3] has the goal of ”By the year 2050, develop a team of autonomous robots that can win against the human world soccer champion team.” This team will surely be formed by heterogeneous robots, a selection of the best players, which will outperform

[3] http://www.robocup.org

any single-origin team. If this is to be the case, how will such a team be built and managed, and how will it play?

This subject has recently been the subject of a prospective analysis[1]. In the present paper, a Communication Framework that leads to implementing these scenarios is defined. To fulfil this scenario, there will be the need for a vocabulary relative to robotic soccer, presented in section 2. The management of interactions between players during the game must also be determined, and a proposal is made in section 3. Finally, we present a summary and look into future work in section 4.

2 Robotic Soccer Domain Concepts

2.1 Physical Objects and Positioning

As pointed out in the scenario presented in [1], robots will have to share the world state, having thus to use a proper vocabulary describing players. Information about their colour and shape should also be expressable. They should be characterised by their skills, like average motion speed and kicking device. This modelling can be seen as an UML diagram in figure 1. Other relevant physical objects are the ball and the referee. Positioning of objects should be shared among team-mates, to enhance the state of the

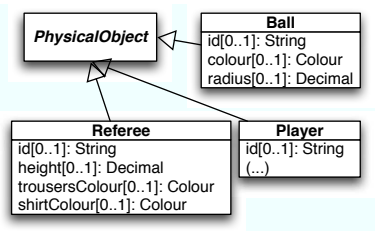


Fig. 2. Physical Objects.

world. The absolute pose of an object is based on a right-hand cartesian co-ordinate system, with the origin placed at the centre of the field, the x-axis pointing at the blue goal and the z-axis up. The robot's orientation, i.e., the direction it is facing, is modelled as a yaw angle relative to the x-axis on the xy plane. The full definition can be found in figure 3.

The uncertainty in positioning determination must be dealt with. In fact, no measurement is entirely reliable and different sensors introduce different kinds of uncertainty. We chose to use the Standard Uncertainty[2]. *AbsolutePositioningWithSU* extends '*AbsolutePositioning*'. There can also be uncertainty about the identity of the observed object (*targetIdentificationConfidence*). In the scenario in [1], most of the positioning exchanged are determined from the viewpoint of the robot, and are thus relative to it. The class (*RelativePositioning*) represents relative positioning with respect to the observer, useing polar co-ordinates. '*RelativePositioningWithSU*' extends the former. Coach-Unilang[3] introduces a definition of field regions, including predefined areas and freely definable areas like circles, which will be included in this framework.

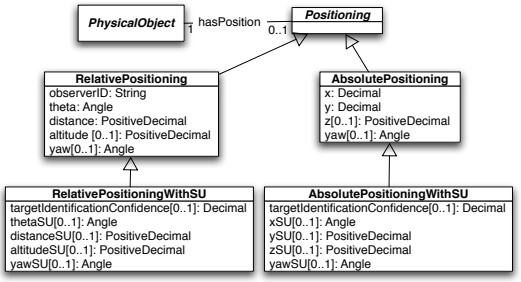


Fig. 3. Positioning related concepts.

2.2 Game Events, Player Moves, Actions and Tactics

During the game, some events occur and may be reported to team-mates, since they are relevant to the world state. Such events are related to temporarily absent players, which may influence decisions or even strategy changes. These events are: sentOff(player), returnedToGame(player), malfunctioning(player) and functioning(player).

Co-operation can be enhanced by the intentional exchange of messages to co-ordinate robots' behaviour. When a robot well positioned to score a goal decides to ask its team-mate holding the ball to perform a pass. Coach-Unilang[3] defines a set of actions, which will be used. Some of these actions have added arguments. These actions and moves are: shoot(), pass(player), forward(fieldRegion), dribble(direction), run(direction), hold(), clear(), intercept(), tackle(player), mark(player), markPassLine (player1, player2), gotoBall() and move(fieldRegion).

Tactics define the players' preferred positioning on the field, as well as the team's pressure and mentality. These definitions will influence the players' options. During a game, a tactics change may have to be communicated to all the players. A set of classes for this purpose can be seen in figure 4. Most of the attributes in the '*Tactics*' class have a discrete set of possible values, e.g. from *veryDefensive* to *veryOffensive* or from 0 to 100. There are predefined formations, like *442* and *433*. There may be the need to use arbitrary formations, using the *ArbitraryFormation* class, as represented in figure 4 by *FormationPosition*. In this class, the positioning of each player is characterised by an horizontal and vertical position. *playerRole* will define the attitude of the player.

3 Inter-robot Communicative Interactions

Since the information in the previous section is to be shared between heterogeneous agents, one also needs to establish how this exchange will be managed. The autonomous agents' community has been dealing with these problems for several years, and one can profit from the results previously obtained.

The transmission of observed information needs only a simple interaction, where one player (*Sender*) will inform some other players (*Receivers*). The acknowledgement is optional. This interaction protocol is represented as an AUML diagram[4]. This proto-

[4] http://www.auml.org

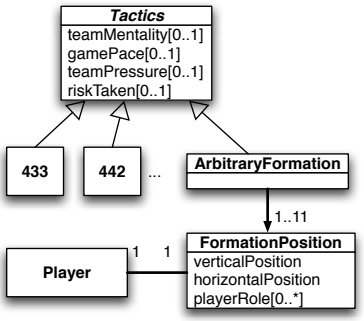


Fig. 4. Tactics related concepts.

col will also be used to advertise choices. An example, where a robot informs others that it intends to shoot at the opposite goal, uses the *intends* (*I*) operator[4], is as follows:

```
(inform :sender robot1 :receiver robot2 (...) :contents (I robot1 (shoot)))
```

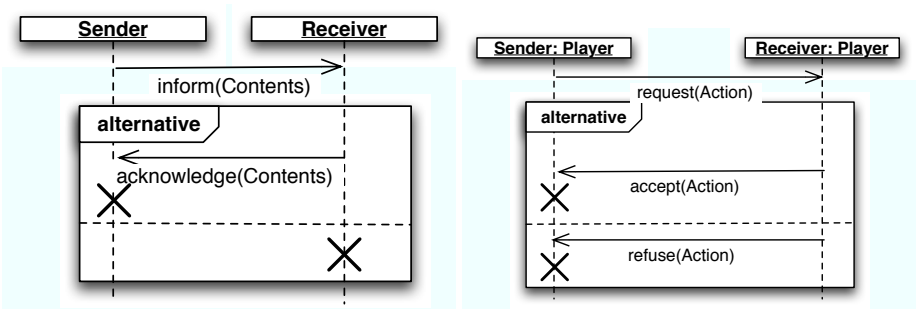


Fig. 5. Inform and Request interaction protocols.

Other interactions are more complex: if a player wants a team-mate to perform a specific action, it will have to request this action, and the requested player will have to either accept or reject the request. Such an interaction resembles the FIPA Request Protocol[5] (figure 3).

4 Summary, Conclusions and Future Work

A communication framework has been defined, contributing to the development of joint, multi-partner, heterogeneous, co-operative and open RoboCup soccer teams. This framework introduces a vocabulary defining a fundamental set of concepts needed by robots during a match. Two kinds of interactions have been defined. The first kind allows robots to share information about the game and their individual intentions. The

second enables momentary co-operation that will lead to more complex moves involving several robots.

This framework is therefore a fundamental set of concepts and protocols for robots to communicate. In order to take co-operation to a higher level, it will need concepts such as role changes and set plays. Further, there is also the need for game statistics, which enable the modelling of the opponent team and could be the basis for a better choice of tactics, prior to and during the game. All these concepts will be considered in the future as possible extensions.

References

1. Mota, L., Reis, L.P., Burkhard, H.D.: Communication challenges raised by open co-operative teams in robocup. In: Encontro Científico do Festival Nacional de Robótica 2006. (2006)
2. ISO: Guide to the expression of uncertainty in measurement. International Organization for Standardization (ISO). (1995)
3. Reis, L.P., Lau, N.: Coach unilang - a standard language for coaching a (robo) soccer team. In Birk, A., Coradeschi, S., Tadokoro, S., eds.: RoboCup-2001: Robot Soccer World Cup V. Volume 2377 of Lecture Notes in Artificial Intelligence., Berlin, Springer Verlag (2002) 183–192
4. FIPA: FIPA SL Content Language Specification. Foundation for Intelligent Physical Agents. (2002)
5. FIPA: FIPA Request Interaction Protocol Specification. Foundation for Intelligent Physical Agents. (2002)

Towards a Generic Anticipatory Agent Architecture for Mobile Robots*

Noury Bouraqadi[1] and Serge Stinckwich[2]

[1] Dépt I.A. – Ecole des Mines de Douai – France
bouraqadi@ensm-douai.fr
http://csl.ensm-douai.fr/noury
[2] GREYC – CNRS / Université de Caen – France
Serge.Stinckwich@info.unicaen.fr
http://www.iutc3.unicaen.fr/serge

Abstract. An anticipatory agent [1] is a hybrid agent which is able to predict changes of itself and its environment. Such agents prove interesting [2] [3] [4] in embedded systems such mobile robots. Indeed, they combine a reactive fast layer with a cognitive layer capable to perform corrective actions to avoid undesired situations before they occur actually. We present in this paper a generic architecture, that we plan to use as a guideline for developing anticipatory agents embedded into robots for search & rescue missions. Our approach relies on software components in order to explicit the anticipatory mechanisms.

1 Davidsson Quasi-Anticipatory Agent Architecture

From the definition of Rosen [5], Davidsson defines a very simple class of anticipatory agent system: it contains a causal system S and a model M of this system that provides predictions of S. As the model M is not a perfect representation of the reactive system, this is called a quasi-anticipatory system. This architecture is rather coarse-grain. It is only composed of 5 parts:

- Sensors: provide information about the agent environment.
- Effectors: allow the agent to act upon its environment.
- Reactor: drives the effectors in reaction to latest information provided by sensors.
- World Model: is an abstract view of the agent's environment based on data collected using sensors.
- Anticipator: modifies the reactor to avoid undesirable predicted world state.

2 MALEVA: A Software Component Model Expliciting Data and Control Flows

Software component [6] is a programming paradigm that aims at going beyond Object-Oriented programming from the point of view of modularity, reuse and improvement of

* This work is partially supported by the CPER TAC 2004-2006 of the region Nord-Pas de Calais and the european fund FEDER.

software quality. Indeed, a software component is a software entity which explicits its dependencies and interactions with other components and resources it relies on.

In this paper, we use the MALEVA hierarchical component model [7] in order to define and implement our anticipatory hybrid agent architecture. Indeed, MALEVA components are close to building blocks of the Brooks subsumption model in their encapsulation and interaction through data exchange [8].

A MALEVA component is a run-time software entity providing encapsulation like objects, while expliciting its interactions with other components. MALEVA components interact only through their interfaces. Interfaces can be of two kinds: data interfaces or control interfaces. Data interfaces are dedicated to data exchange, while control interfaces are dedicated to control flow.

A component can be either active or passive. A passive component is a component that does perform some computation only after being triggered through one of its control input interfaces. Once the component computation is over, it stops until being again triggered. Contrary to a passive one, an active component don't need to be triggered to act. It uses a thread in order to run autonomously.

3 Overview of our Generic Anticipatory Agent Architecture

As shown on figure 1, our agent architecture is an assembly of five components: sensors, effectors, reactor, reaction ticker and anticipator. The first three parts (namely: sensors, effectors and the reactor) are application specific. However, the reactor is instrumented in order to provide two generic interfaces for modifications input: one for modification data flow and the second for modification control flow. The former allows the anticipator to provide modifications to be performed on the reactor, while the latter allows the anticipator to trigger the modifications. These two interfaces can be viewed as the so-called "meta-interfaces" in the work on Open Implementation [9], since they allow a disciplined modification of the reactor.

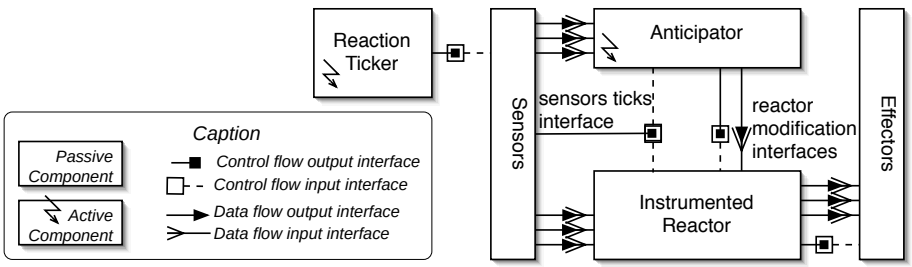


Fig. 1. Our Anticipatory Agent Architecture.

The "Reaction Ticker" is a generic active component that drives the agent's reaction. It defines the frequency at which the agent will sense its environment and react to changes. Indeed, the "Reaction Ticker" triggers the Sensors component every *m* milliseconds, where *m* is the duration between two ticks and depends on the application context.

The Sensors component[3] collects data from the agent's world and propagates it through its data interfaces to both the reactor and the anticipator. Then, it triggers the activity of both the anticipator and the reactor. When getting triggered, the reactor decides the appropriate reaction to perform and translates this decision into data propagated to the Effectors component[4].

4 The Anticipator Component

The Anticipator component is an active composite component (see figure 2). It is active since it includes its own ticker that allows it to run concurrently to the reactor. The ticking frequency is higher than the "Reaction Ticker" one, since the anticipator has to work faster than the reactor, in order to make useful predictions. By expliciting the ticking control through the "Anticipation Ticker", this frequency can be easily changed. This feature is very important since it allows to tune the anticipator consumption of resources (computing, energy, ...), particularly in case of embedded devices with low capabilities. This frequency can even be changed dynamically, according to resource evolutions, such as the battery level in a mobile robot.

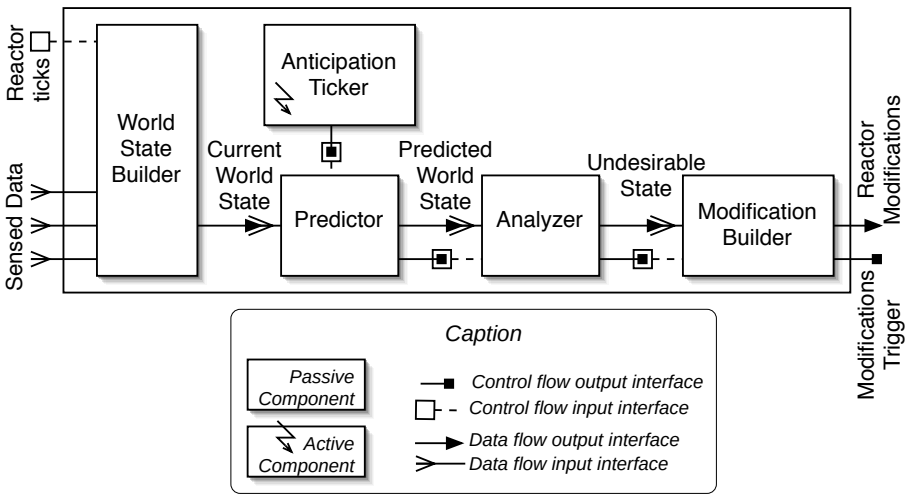


Fig. 2. The Anticipator Architecture.

Each tick of the "Anticipation Ticker" makes the Predictor component predict the next world state and the next reactor action. Then, the Analyzer component analyzes the

[3] Actually, the Sensors component can be a composite with multiple subcomponents corresponding to different sensors.

[4] Actually, the Effectors component can also be a composite with multiple subcomponents corresponding to different effectors.

predicted world state and identifies undesired situations. In case of undesired states, the "Modification Builder" component plans the appropriate modifications and transmits them to the reactor.

It worth noting that, except the data input interfaces connecting the "World State Builder" to sensors, all other interfaces are generic. Therefore this architecture can be reused in multiple contexts.

5 Conclusion and Ongoing Work

In this paper, we draw the foundations for a generic agent architecture based on the Davidsson anticipatory model. This architecture can be reused in multiple contexts and may also serve at the basis for a methodology to design an anticipatory agent

Implementing the examples ("bot in a maze") described in the Davidsson paper, enabled us to prove that it is possible to propose a sufficiently generic architecture for an anticipatory agent regarding the application domain. A more complete validation will be soon carried out with experiments under development of a vacuum cleaner robot simulation. We also plan to experiment our architecture on mobile robots in a search & rescue project.

Another question we would like to explore is how to instrument the reactor component and how to automate the transformation in order to introduce a modification interface. This point is rather complex and varies according to the reactor architecture and its properties. For example, in the case of the subsumption model [8], we need to establish the interaction between the modification interface and the reactor layer.

References

1. Davidsson, P.: A linearly quasi-anticipatory autonomous agent architecture : Some preliminary experiments. In: Distributed Artificial Intelligence Architecture and Modelling. Number 1087 in Lecture Notes in Artificial Intelligence, Springer Verlag (1996) 189–203
2. Stolzmann, W.: Anticipatory classifier systems. In Koza, J.R., Banzhaf, W., Chellapilla, K., Kalyanmoym, D., Dorigo, M., Fogel, D.B., Garzon, M.H., Goldberg, D.E., Iba, H., Riolo, R., eds.: Genetic Programming 3: Proceedings of the Third Annual Conference, University of Wisconsin, Madison, Morgan Kaufmann (1998) 658–664
3. Einarson, D.: Hierarchical models of anticipation. In: CASYS'2001. (2001)
4. Shang, F., Cheng, J.: Implementation issues of anticipatory reasoning-reacting systems. In: International Workshop on Research Directions and Challenge Problems in Advanced Information Systems Engineering. (2003)
5. Rosen, R.: Anticipatory Systems - Philosophical, Mathematical and Methodological Foundations. Pergamon Press (1985)
6. Szyperski, C.: Component Software - Beyond Object-Oriented Programming. 2nd edition. Addison-Wesley (2002)
7. Briot, J.P., Meurisse, T., Peschanski, F.: Une expérience de conception et de composition de comportements d'agents à l'aide de composants. L'Objet **11** (2006) 1–30
8. Brooks, R.A.: A robust layered control system for a mobile robot. IEEE Journal of Robotics and Automation **2** (1986) 14–23
9. Kiczales, G.: Beyond the black box: Open implementation. IEEE Software (1996)

An Experiment in Distributed Visual Attention [1]

P. Bachiller[1], P. Bustos[1], J. M. Cañas[2] and R. Royo[1]

[1] University of Extremadura, 10071, Cáceres, Spain
{pilarb, pbustos}@unex.es
[2] University Rey Juan Carlos, 28933, Móstoles, Spain
jmplaza@gsyc.escet.urjc.es

Abstract. Attention mechanisms of biological vision have been applied to machine vision for several applications, like visual search and object detection. Most of the proposed models are centred on a unique way of attention, mainly stimulus-driven or bottom-up attention. We propose a visual attention system that integrates several attentional behaviours. To get a real-time implementation, we have designed a distributed software architecture that exhibits an efficient and flexible structure. We describe some implementation details and real experiments performed in a mobile robot endowed with a stereo vision head.

1 Introduction

The visual attention system in a mobile robot acts as a dynamical device that interacts with the environment to select what might be relevant to current active tasks. At the same time, it should maintain responsiveness to unforeseen events. More specifically, it should enclose the following functions [12]: selection of regions of interest in the visual field; selection of feature dimensions and values of interest; control of information flowing through the visual system; and shifting from one selected region to the next in time or the "where to look next" task.

Attention can be classified according to various aspects. In psychology, the terms generally used are active (voluntary) and passive (involuntary) attention [3]. From a stimulus point of view, there is overt or covert attention depending on the way the stimulus is attended [10]. Overt attention is the act of directing our eyes towards a stimulus source. Covert attention is the act of mentally focusing on a particular stimulus without any motor action. Attending to the mechanism that drives attentional control, there are two kinds of execution methods: one is bottom-up or stimulus-driven, which shifts attention to regions with visual features of potential importance; another is top-down or goal-directed, which uses knowledge of the visual features of the desired target to bias the search process.

In recent years, visual attention has taken an important place in robotics research. Most of the proposed models have focused on pure bottom-up [2][12] and some on

[1] This work was supported by the Department of Science and Technology of the Extremadura Government (grant 3PR05A044), by the Madrid Government (grant S-0505/DPI/0176) and, also, by the Spanish Ministry of Education and Science (grant DPI2004-07993-C03-01).

top-down [9] attention. There have been some efforts on combining both forms of attention by weighting bottom-up saliency maps with top-down information [8,11].

The approach proposed in this paper is mainly characterized by the integration of different attention categories at a single system endowed with a flexible and adaptable architecture. The proposed system is modelled as a collection of processes collaborating to fix a visual target and to choose the next one. The complexity of the resulting global system requires the use of distributed software engineering techniques. We use the Internet Communication Engine (Ice) middleware platform [1] and a custom component model specifically tailored to build distributed vision architectures.

2 Architecture of the Proposed System

The visual attention system proposed in this paper integrates several ways of attention to work successfully at different situations. It has been designed and tested on a mobile robot with a stereo vision head. The net of processes that compose the system are a set of Ice components collaborating to fix a visual target. As shown in figure 1, the elements in the architecture are roughly organized in two branches that converge in the lower part of the graph. From this point a closed-loop connection feeds back to the upper initial part. The two branches divide the visual function in a manner analogous to the "what" and "where" pathways proposed in neuroscience [7]. This division allows for a specialization of functions, dedicating specific resources to each branch and sharing what is common from lower-level processes. The "what" branch tries to find and track a specific target in the image using bottom-up computed regions of interest (ROI) and top-down specification of targets. The "where" branch extracts geometric information from stabilized ROI's and selects those regions that meet certain requirements, such as being on the floor plane, being too close or being in the current heading direction. The information from both branches has to be integrated solving an action selection problem. Given a current task or set of tasks, where to look next? This is accomplished by the lower component in the graph which outputs commands to the underlying motor system. In our system "looking to something" implies a 3D positioning of the robot with respect to the target, which we call a 3D saccadic movement. Following this reasoning, solving a generic navigation task such as going some-where following a predetermined set of (remembered) landmarks, amounts to generating the "correct" set of saccadic movements that will approach the current target while avoiding potential obstacles. This set cannot be computed a priori as long as it is the result of extended dynamical interactions between the robot and its environment. It is partially defined in the programmed code and partially selected from finding out how the outside world is. The whole system works as a complex mechanical device that is attracted towards some features and rejected from others. The specific interleaving between approaching and avoiding is given by an implicit time relation that links internal parameters and external geometry.

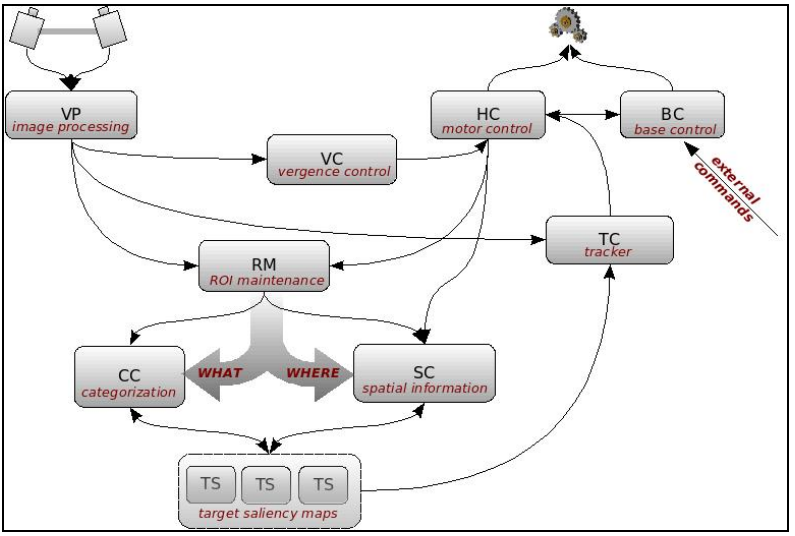


Fig. 1. Architecture of the visual attention system.

3 Components and Connections

Each component in the architecture is a C++ coded UNIX process using the Ice middleware. Similar to CORBA, each component supplies a public interface that can be used by other participants to call its methods remotely. We now describe the components depicted in figure 1 and its connections:

- **Visual processor** (VP): Acts as a vision server capturing images from the cameras and computing Harris-Laplace regions of interest at multiple spatial scales as described in Lowe[4]. For each set of ROI's, it fills a shared double buffer to supply quick responses to client requests.
- **Head controller** (HC): Head motor controller. Computes direct and inverse kinematics of the binocular head. It waits for client commands and passes them to a dedicated microcontroller that executes the PID loops. It keeps a copy of the state of the motors and of the head and can answer queries about them directly.
- **Base controller** (BC): Base motor controller. Computes direct and inverse base kinematics and waits for client commands that are passed to a dedicated microcontroller that executes the PID loops. It keeps a copy of the state of the motors and of the mobile base and can answer queries about them directly.
- **Vergence controller** (VC): Works independently to ensure the convergence of both cameras to the same spatial point. Vergence control is done by a multi scale cross correlation between the centre of the dominant camera and the epipolar homologous window in the other one. It looks for the maximum of the resulting structure (maximum of the image window at the whole scale-space) and performs the shift in the slave camera that leads to the convergence of both. To optimize resources the search is done in an increasingly wider window triggered by a failure in the direct matching of foveas. Vergence controller is a client of VP and HC.

- **ROI Maintenance component** (RM): Maintains a stable representation in time of recently perceived regions of interest. It provides a map of regions built in the camera reference system. It works as a short-term memory maintaining information about each region such as: raw image window, permanence time, attention time and update time. As VP does, this component always maintains an available ROI list that can be sent to its clients on demand.
- **Categorization component** (CC): Classifies regions of interest into known categories that can be used as landmarks. The current implementation uses Euclidian distance to compare SIFT[5] and RIFT[6] descriptors of candidate ROI's with previously stored examples. It accepts a target category so it tries to find a region compatible with the target and returns a list of candidates.
- **Spatial component** (SC): Compute spatial and geometric features of the last received ROI list and organize them in a head reference system. Properties computed by this component are: 3D position of the region using vergence and disparity; and planarity and plane orientation of the overall region by estimating the best homography. More useful properties will be implemented in near future to determine local shape with greater precision so more sophisticated hierarchical categorization can be accomplished in collaboration with CC.
- **Target selectors** (TS): Integrate local representations from CC and SC according to some criteria. Attending to their functionality, they request for specific patterns and features to CC and SC, respectively. CC provides a list of classified regions that are integrated with spatial information from SC to construct a final saliency map. This map is used to select a focus of attention that can be sent to the tracker component. This last action only takes place when the component is active. TS are specialized on a specific action. They are in communication with other components outside the attention system that activate them to take the attention control in order to carry out an action. An example of target selector component is the landmark selector, which is linked to a follow-landmark action. Another one is the obstacle selector, related to an action of avoiding-obstacles. The attention-action relationship is not a mandatory condition. A target selector can be linked to an idle behaviour that allows the system to maintain a pure bottom-up attention.
- **Tracker component** (TC): Receives the location of a region from the active TS and maintains the focus of attention on such region until another position is received. To achieve this goal, the TC implements a predictive tracking algorithm that combines the distance among RIFT descriptors of the regions and normalized correlation in YRGB space.

4 Interaction Dynamics

The architecture just described provides an attentional mechanism that can be incorporated in a wider network of components adding behaviour-based control, task selection, planning, topological maps and other abilities. In the experiment shown here, we use a couple of coordinating behaviours -approach and avoid- that activate the target selectors (TS's) of the attentional system. Together, they can be seen as a visual-goto-point (VGP) compound behaviour which is the basic serializing constituent

of most complex navigation tasks. The attentional mechanism provided to VGP endows it with inner dynamic loops that take care of target detection, recognition, searching and tracking, lost target recovery and unexpected obstacle detection. In addition, these features are the result of parallel activities that get serialized to gain access to the orientable cameras.

When VGP activates, two activities take place simultaneously: a target landmark is downloaded to the attentional system through a TS, and another TS is activated to detect potential obstacles in front of the robot. Both TS's are coordinated in a very basic way by the VGP to achieve the current goal. The law to follow is hierarchical: "if there is free way, approach the target". First, the landmark TS activates to search the target. Once it is fixated, the obstacle TS activates to locate potential hazards in the course towards the detected landmark. If the near space is free of obstacles, the cameras will search again the landmark and the base of the robot will reorient towards the gaze direction and start moving forward. Then again, the obstacle TS will use its short term spatial memory representation and covert attention capabilities to gaze towards any close enough obstacle in the way. If it happens, the base will reorient in a direction perpendicular to the pan angle in order to avoid the nearest obstacle. Once the danger is over, the landmark TS will regain control to relocate the target and establish a new heading direction. This alternating dynamics keeps going on until the goal landmark is within some specified distance and orientation.

Several kinds of attention can be observed in the last example. When the obstacle TS chooses a target to attend to, it performs overt attention on the obstacle so avoidance based on pan angle can take place. But when the landmark TS is waiting, it performs covert attention on the target so it can be quickly fixated again when it gets activated. In a similar way, we can speak about bottom-up and top-down attention. ROI's detected by the VP drive attention in a bottom-up way selecting those areas of the image most informative. From this set, a few are chosen attending to task dependent constraints such as target landmark or specific known geometric properties of obstacles.

5 Experiments

The visual attention system has been tested in a threaded mobile robot endowed with a stereo vision head. It encloses five degrees of freedom with digital PID controlled servos and two ISight Firewire cameras (figure 2). This type of robots has been developed in our Laboratory and is widely used in prototyping and algorithm testing.

The architecture of components just described runs on a local cluster of computers. Each process is an independent C++ application, which includes Ice objects and proxies to communicate with other processes. An Ice object is an entity used by a server to respond to client requests. An Ice proxy represents an Ice object local to the client that can communicate remotely to the server. Ice provides a remote method invocation capability that can use both TCP and UDP as the underlying protocol. In our current implementation, the network of processes is distributed among four physical processors – dual Opteron board for VP, HC and BC; 3GH-HT P4 for VC, TC and RM; and AMD64 dual core for CC, SC and TS- as seen by the Linux operat-

ing systems. The computers are locally linked by a 1 Gb ethernet switch providing enough bandwidth for real time communication among components.

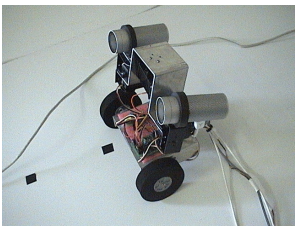


Fig. 2. Robot used in the experiments.

We have designed a simple experiment for initial testing and validation of the proposed architecture. The robot has to localize and approach a landmark (star) in its near space avoiding an obstacle that blocks its heading direction. The running system incorporates all the components described before. As target selectors (TS's) we use landmark and obstacle selectors working in cooperation. Actions linked to selectors are approach and avoid, configuring a sort of visual goto-point.

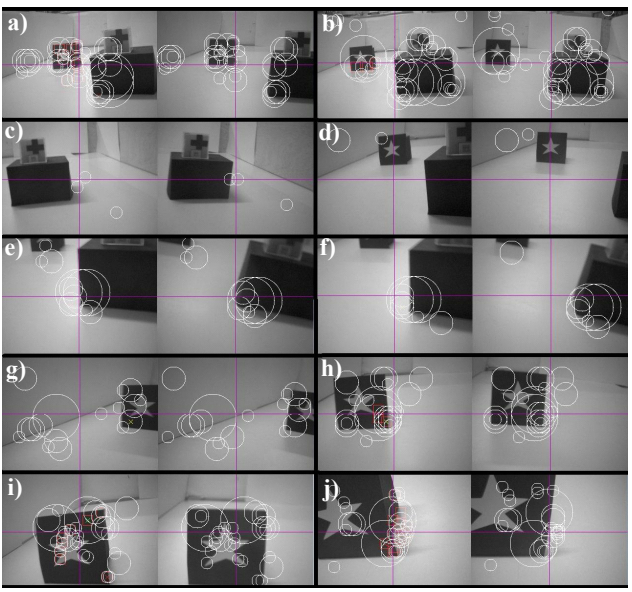


Fig. 3. Experiment of visual navigation avoiding obstacles.

Changes of attention alternating these two kinds of targets can be appreciated in the sequence above (figure 3). Initially, a) attention is fixated on the landmark and an action of approaching begins. Then, b) and c) frames, obstacles gain control of attention guiding the robot to avoid them. After several frames - d) - landmark is fixated again providing a new goal heading. To keep up with the new situation, the obstacle selector changes its focus of attention, e) and f). Once all the obstacles have been

avoided g), attention is again centred on the landmark making the robot to approach it and finally reach the goal position, j).

6 Summary and Conclusions

In this paper, we have shown an experiment in distributed visual attention on a mobile robot with a stereoscopic head. Our goal has been to test the potential of combining ideas from visual neuroscience, distributed software engineering and robotics. We think that the proposed architecture will ease the way to model and implement more perceptual and cognitive capabilities in our robots. This first result shows that the complexity of distributed bottom-up and top-down attention can be integrated in a visual navigation framework to solve what we have called the visual-goto-point problem. Much work remains in this multidisciplinary area, but the possibilities offered by new multicore processors in conjunction with communications middleware will open new spaces for bio-inspired robotics modelling and building.

References

1. Internet Communication Engine. http://www.zeroc.com/ice.html
2. L. Itti and C. Koch. A saliency-based search mechanism for overt and covert shifts of visual attention. Vision Research, 40(10-12):1489–1506, May 2000.
3. W. James, The principles of psychology, New York: Holt, 1890, pp. 403-404
4. D. G. Lowe. Object recognition from local scale-invariant features. In ICCV, pages 1150–1157, 1999.
5. D. G. Lowe, Distinctive image features from scale-invariant keypoints, International Journal of Computer Vision, vol. 60, no. 2, pp. 91–110, 2004.
6. K. Mikolajczyk and C. Schmid. Indexing based on scale invariant interest points. In ICCV01, pages 525–531, Vancouver, Canada, July 2001.
7. A.D. Milner and M.A. Goodale. The visual brain in action. Oxford UniversityPress. Oxford 1995
8. V. Navalpakkam, L. Itti, An Integrated Model of Top-down and Bottom-up Attention for Optimal Object Detection, In: Proc. IEEE Conference on Computer Vision and Pattern Recognition (CVPR), pp. 1-7, Jun 2006
9. R. P. Rao, G. Zelinsky, M. Hayhoe, and D. H. Ballard. Eye movements in iconic visual search. Vision Research, 42(11):1447–1463, Nov 2002.
10. A. Treisman and R. Paterson, Emergent features, attention and object perception, J. Exp. Psychol: Human Perception and Performance, 1984, 10:12-31.
11. A. Torralba et al. Contextual guidance of attention in natural scenes: The role of global features on object search. Psycological Review, October 2006
12. J.K. Tsotsos, et.al., Modeling visual attention via selective tuning, Arti. Intell., 1995, 78:507-545.

Author Index

Proceedings of the
3rd International Workshop on
Multi-Agent Robotic Systems - MARS 2007
ISBN: 978-972-8865-85-6
http://www.icinco.org