ResearchGate

See discussions, stats, and author profiles for this publication at: https://www.researchgate.net/publication/4150083

Decentral control of a robot-swarm

Conference Paper · May 2005

DOI: 10.1109/ISADS.2005.1452083 · Source: IEEE Xplore

CITATIONS
3

READS
65

7 authors, including:

Jochen Kerdels
FernUniversität in Hagen
36 PUBLICATIONS **131** CITATIONS

SEE PROFILE

Some of the authors of this publication are also working on these related projects:

CManipulator View project

All content following this page was uploaded by Jochen Kerdels on 06 May 2014.

The user has requested enhancement of the downloaded file.

Decentral Control of a Robot-Swarm

I.Dahm, M.Hebbel, M.Hülsbusch, J.Kerdels, W.Nistico, C.Schumann, M.Wachter
Computer Engineering Institute, University Dortmund, Otto-Hahn-Strasse 4, D-44221 Dortmund
ingo.dahm@udo.edu

Abstract

In this paper, we present an approach to construct a universal architecture to control a team of autonomous robots. The suggested approach is robust against communication problems and robots' hardware failures. The system profits from its decentral architecture. We present, how the control structure is designed and implemented and how basic tasks are scheduled and distributed running the approach. Thereto, we present the three modules of the architecture: A robust communication, a decentral data storage by sensor-fusion, and a decentral task-scheduling. As a matter of fact, we do not focus on a specific implementation but on the concept behind our system-wide approach. For the proof of concept, we reference to an implementation which was honored by the first place of the Open Challenge at RoboCup world championship in Lissboa 2004.

1. Introduction

Autonomous teams of robots are expected to solve several scientific and industrial challenges. Robots can work at places, which are uncomfortable, dangerous or even unreachable for human beings (e.g. observations on planet mars) [1]. It is difficult, expensive or just impossible to teach, control or maintain the robots at such places. Thereto, a robot team must act reactive, fault-tolerant and autonomously. Thus, it is a challenge to implement an efficient decentral control architecture [2], [3], [4]. For the following, such an architecture consists of three major components:

- a robust communication,
- a fault-tolerant data distribution,
- and an efficient task scheduling.

In the next three sections, we suggest implementations for each of this modules. Further, we show exemplary, how this approaches are adjusted in order to fit the special side-constraints of a real application. To benchmark the power of our decentral approach, we chose the so-called 'RoboCup ' Scenario. In this application, a number of four robots fights against another four robots to shoot a ball in the opponents goal. Side-constraints like gaming rules and robots own stability must be considered [5]. In our case, we use Sonys ERS-7 robots to participate in the so-called Four-Legged-League as illustrated in Fig. 1. We selected the freely available GermanTeam code of 2003 (GT2003) as starting point of our research [6].

2. Robust Communication

The Sony ERS7 is equipped with a 802.11b wireless LAN device to communicate with its teammates. This communication is used to synchronize the world-model between the robots and for task scheduling as described later.

The robot's operating-system includes native support for IP-based communication using the UDP or TCP protocol. It turned out that UDP is the better choice for sending short packets of data from one robot to another. The reasons for using UDP were:

- TCP produces much overhead
- TCP connections have to be reestablished if lost

In our approach every robot sends information about its world model and its task-scheduling related data to every other robot of his team. There are two ways to spread these information among the team. The first way is an UDP-broadcast which sends the data to all other robots and computers on the same network. On the other hand, every individual of the swarm has to process every packet even if the data is not sent from a team-mate. Thereto, we use single-casts: every robot sends out a separate packet to every other robot in the team.

Because every robot needs the IP-addresses of the other robots in its team, we created the so-called

0-7803-8963-8/05/$20.00 ©2005 IEEE.

Dog-Discovery-Protocol (DDP). Every robot sends an UDP-broadcast packet to every other robot on the network at a rate of 500 mHz. This packet contains a team-identifier of the robot. Every robot in communication range receives this packet and checks for the right team-identifier. If it equals its own identifier, the receiver adds the IP-address of the sender to a list of teammates. If for some reason a robot does not receive any packet from a known sender, the suspicious address is removed from the list.

This communication scheme is proofed to be robust by testing in the development and at several robot-soccer games on the different events and locations. Each robot quickly finds its teammates and starts communicating with them. Also after a robot runs out of battery or crashes, all other robots still communicate to each other. After rebooting the defect robot reintegrates itself into the team-communication within seconds.

On big RoboCup events like the German-Open or the world championship there are many wireless LAN networks which interfere with each other. In this environment the packet-loss (packets of size P) is high. Thus, some of the sent data is lost. On the other hand, a minimum bandwith B is granted to each team by the technical rules [5]. Therefore, data is sent not faster as P/B (in our case: 100ms) to other robots such that data distribution and task scheduling modules have as current as possible data.

Fig. 1. Sony's ERS-7. This is the newest 20 degree-of-freedom, fully autonomous four-legged robot used in RoboCup robot soccer games.

3. Data Distribution and Fusion

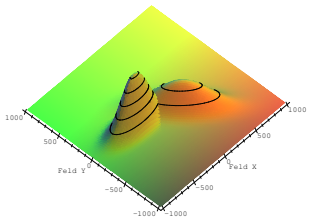
When using distributed systems in a realtime environment, you have to deal with several problems. One of them is inaccurate sensor information caused by quantization effects, detection problems or analog to digital conversion. This leads to inaccurate localization of robots, team-mates and objects. As positioning is a vital task of a swarm, it is a challenge to solve this problem. Moreover, it seems to be beneficial to combine many imprecise observations in order to calculate a fused information with higher correctness. The standard way to solve the problem is the so-called sensor fusion. Thereto, several constraints must be considered:

1) A common timebase is needed. Thus, the distributed system can find matching sensory information due to their time-stamp.
2) If sensor fusion is used for localization, then the position of each sensor has to be well known, too.
3) It is meaningful to know about the reliability of all sensor data. Hence, information can be weighted. This leads to improved confidence information of the merged data.

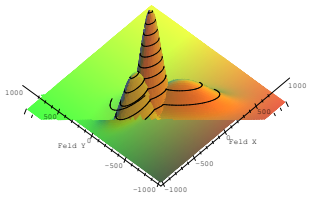
In the next section, we present the idea of sensor fusion in our approach.

A. Implementation

Object tracking in an important task for sensor fusion, especially in case of robot soccer. Typically, more than one robot are able to recognize the ball at the playing field. Unfortunately, due to low camera quality, the estimated position is inaccurate. A robot is able to to determine the bearing of accurately, but the distance has a tolerance of up to 30 percent as illustrated by Fig. 3(a). To determine the distance more accurately, additional information is needed. As a basic approach, the position can be estimated by a simple cross bearing. If there are more than two robots a more complex strategy is needed. For this purpose, we suggest the use of a gaussian fusion algorithm [6]. Every percept is represented by a two dimensional Gaussian distribution which reflects the two-dimensional probability function of the observed object. Two or more Gaussian distributed probability functions can be merged, by weighted multiplication, whereas the weight illustrates the reliability of the corresponding sensor. As the algorithm is associative, a variable number of percepts can be merged [7]. Thus, the suggested methodology is a robust way to share sensor information in a team of robots.



(a) Two ball percepts before merge



(b) Two single percepts and resulting ball position

Fig. 2. Example of sensor fusion with ball percepts from two robots

4. Task Scheduling

As presented before, we expect a working communication system for best results in task scheduling. On the other hand, the approach works as well, if robots knock down or communication failures occur.

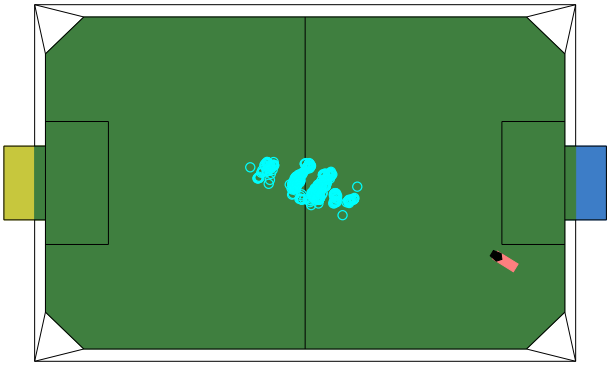
Thus, let's assume there is a team of robots which are able to communicate with each other. Further, this robots have to solve a task which can be divided into subtasks. Let's assume there are τ different tasks $T_1 \dots T_\tau$ which can be executed on at least one individual R_j of the robotic team. When there are more tasks than robots ($\tau >$), we have to deal with a matching problem. This can be solved in a similar fashion as the well-known problem of task distribution in a metacomputer network [8], [9].

To solve it efficiently, priorities must be assigned to all tasks. The next step is to obtain knowledge about the actual situation and the environmental conditions:

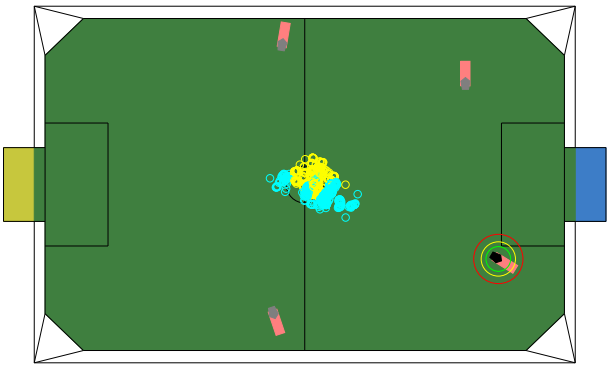
A. *Classification*

Typically, for each robot R_i the environment at time t can be described by its own world model $M_i(t)$ [10]. Without loss of generality, $M(t)$ can be classified into a set of situations s_k with $1 \leq k \leq \sigma$.

We assume that in every situation s_k exists an optimal priority-rating v_i of the tasks T_i. Thereto, let $\vec{v} = (v_1, v_2, \dots v_\tau)$ be a τ-dimensional weight vector, where the priority of each task $T_1, \dots, T_\tau$ is stored. If there are σ different situations, then there will be σ corresponding weight vectors $\vec{v}^1 \dots \vec{v}^\sigma$, too. The



(a) Using sensor data from a single robot



(b) Using sensor fusion. The yellow circles are fused ball positions, the white ones are from the robot in the bottom right corner.

Fig. 3. Calculated ball position of 250 sensor readings. The ball is in the center of the field.

classification of the actual situation can be done easily by using the world model of the robot. As presented in the last section, we make use of a global world model. This data structure is calculated from local observations (sensor input) and communicated sensor data from other robots.

Thus, after classification of the actual world-state, finding the corresponding optimal rating is a matching problem [11].

B. *Matching*

To find the best solution of the matching problem, we suggest to estimate the ability w_j^i of robot R_i to solve task T_j. The current world-state belief $M_i(t)$ of robot i is assumed to be classified into a situation $s_{j(M_i)}$ in the first step. Finding the optimal task distribution means to maximize the dot-product of task-rating vector and priority-vector according to the actual situation $v(s_j)$. This can be done efficiently by applying the Hungarian Algorithm [12]. If all robots had the same information

about their environment (i.e. if they shared the same world model), every robot would find the same solution of the matching problem.

The matching would be identical and unique, resulting in a deterministic problem solution. However, due to the physical limits of communication speed and inaccuracies in measurements, the world models of the robots differ from each other. Thereto, it is necessary to estimate the decisions of all teammates and address the hazards which might occur.

C. Estimation

Our estimation follows the basic approach: Every robot assumes, that its own world model is identical to all other robots world models. Formally spoken: Robot R_i assumes all robots use world model $M_i(t)$. The simplest way to minimize the estimation error is to prevent $M_i(t)$ from diverge to $M_j(t)$. This can be done by using a global world model and frequent exchanges of observations and sensor data as done in our original code [6], [10].

Thereto, inaccuracies occur only due to the limitations of communication speed and package loss. Anyway, as conflicts might occur, we need an arbitration technique.

D. Arbitration

Hazards can occur whenever reality and estimate differ from each other [13]. A serious problem is a non-unique task assignment: first, this can lead to allocation problems; second, it blocks robot resources which should be utilized more efficiently. As a result, an efficient arbitration is required to avoid such conflicts. As the robotic system works in real-time, it's required an approach that fits this additional constraint.

As a basic approach, we broadcast the computed scheduling table of every robot $R_i \, \forall \, i \in [1, \ ]$. Thus, hazards in matching can principally be detected and alternative tasks can be scheduled.

In the VR-system, the arbitration should not create a heavy load on the processors and the communication. Thus, we suggest to minimize the communication load by transferring only a time-stamp and the scheduling table: all data that is needed to be exchanged between the robots is the task-rating for all tasks.

When the product $\cdot \tau$ is small, the size of the transmitted data is reasonable (like one UDP packet). The consumed processing power is negligible compared to the communication time. In our case, we use five robots and about 20 tasks. Task-rating is done in double-values (4 Bytes). Thus, without header, we send 400 Bytes after each scheduling step. Including latency of 30ms for each packet, transfer takes less than 130ms. This enables more than seven updates per second.

A second approach is to use a single master to perform the arbitration. This would increase the update rate as only the master has to broadcast its scheduling table. In our case, an update rate of more than 30 Hz could be reached. On the other hand, this approach suffers from reduced robustness, as the master is a single-point-of-failure. This problem can be solved by a logical master which requires additional software and communication protocols.

In our case, the first approach works fine, since it is easy to implement and we do not have to care about availability of robots. As each task-rating packet contains a time-stamp, the arbiter is able to give priority to the youngest subscriber (LIFO). This makes it possible to handle communication failures or synchronization problems, because the important tasks are implicitly scheduled to robots which were known to be reachable a short time before.

If some robots are unavailable, the remaining robots perform the high weighted tasks in the according configuration. In case of a complete communication breakdown, each robot just executes the task with the highest product of priority by ability $v_i^r \cdot w_j^r$.

5. Results

In 2004, we equipped our robot team with the suggested control-structure. We observed no communication problems during all championships. Further, it was clearly observable, that the robots localized significantly better than 2003 due to the fusion of sensory information. Finally, our novel task distribution took us to the semi-finals at the German Open 2004 and at the US Open 2004. We placed third at the Australian Open and won against the last years world-champion UNSW [14]. Finally, at the world championship, we provided a demonstration of our software for the RoboCup Open Challenge and got honored by the first place.

The suggested approach seems to be a useful solution for decentrally controlled robotic systems. It offers a robust communication platform, an efficient data distribution, and an automatic task scheduling. The algorithmic complexity is moderate - our system works in real-time. In our the exemplary application of a Sony Aibo team, the processing-power consuming image processing and analysis works in parallel to the control application at

the same processor (QED RM5232 at 567 MHz) and provides the full rate of 30 frames per second.

6. Conclusion

As next steps, we will open the framework to other applications, such as surveying and mapping by a swarm of robots. We will investigate in more transparent ways to prioritize tasks and a fully-automatic situation classification. Thereto, we will use an adaptive classification with automatic confidence estimation [15]. Then, we expect the framework to be useful for a wider range of applications.

7. Acknowledgments

This work was supported by Microsoft Deutschland GmbH and the DFG under the program of emphasis SPP 1125. We thank our colleague Jens Ziegler and our students Thomas Kindler and Jörn Hamerla for their valuable work at this project in the past.

8. References

[1] M. Mataric and G. Sukhatme, "Task-allocation and coordination of multiple robots for planetary exploration," 2001.

[2] T. Balch, "Grid-based navigation for mobile robots," 1996.

[3] T. Vu, J. Go, G. Kaminka, M. Veloso, and B. Browning, "Monad: A flexible architecture for multi-agent control," 2003.

[4] L. Parker, "Alliance: An architecture for fault-tolerant multi-robot cooperation," 1998.

[5] Sony Legged League Participants, "Sony four legged robot football league rule book," tech. rep., Sony Legged League, 2002.

[6] M. Juengel, M. Loetzsch, R. Brunn, M. Kallnik, N. Kuntze, M. Kunz, M. Risler, T. Laue, T. Roefer, I. Dahm, M. Hebbel, M. Wachter, A. Osterhues, and J. Ziegler, "German Team 2003," in *RoboCup 2003*, Lecture Notes in Artificial Intelligence. Springer, 2003.

[7] S. Deutsch, T. Dickhöfer, W. Ding, K. Engel, P. Kudlacik, A. Osterhues, J. Prünte, A. Reiß, S. Schmidt, C. Thiel, and M. Wachter, "*Sony Legged League: Entwicklung von verteilten Algorithmen zur effizienten Kontrolle von autonomen Fußballrobotern.*"

[8] U. S. C.Bitten, J. Gehring and R. Yahyapour, "The NRW-Metacomputer. Building Blocks for A Worldwide Computational Grid," in *International Parallel and Distributed Processing Symposium 2000*, 2000.

[9] U. Schwiegelshohn and R. Yahyapour, "Resource Allocation and Scheduling in Metasystems," in *Proceedings of the Distributed Computing and Metacomputing Workshop at HPCN Europe* (P. Sloot, M. Bibak, A. Hoekstra, and B. Hertzberger, eds.), pp. 851–860, Springer–Verlag, Lecture Notes in Computer Science LNCS 1593, April 1999.

[10] I. Dahm and J. Ziegler, "Adaptive methods to improve self-localization in robot soccer," in *RoboCup Symposium Fukuoka*, 2002.

[11] A. H. Timmer and J. A. G. Jess, "Exact scheduling strategies based on bipartite graph matching," pp. 42–47.

[12] M. Munkres, "Algorithms for the assignment and transportation problems," *Journal of the Society of Industrial and Applied Mathematics*, vol. 5, no. 1, pp. 32–38, March 1957.

[13] E. H.-M. Sha and K. Steiglitz, "Maintaining bipartite matchings in the presence of failures," in *International Parallel Processing Symposium*, pp. 57–64, 1993.

[14] A. Olave, D. Wang, J. Wong, T. Tam, B. Leung, M. S. Kim, J. Brooks, A. Chang, N. V. Huben, C. Sammut, and B. Hengst, "The UNSW RoboCup 2002 Legged League Team," in *The First RoboCup Australian Open 2003 (AORC-2003)*, 2003.

[15] I.Dahm, "Neural networks with on-the-fly confidence-estimation," in *IEEE International Conference on Signal Processing (ICSP)*, 2004.

View publication stats