Mobile Robot Coordination Using Fear Modeling Algorithm

Michał Konarski, Szymon Szomiński, and Wojciech Turek
AGH University of Science and Technology, Krakow, Poland
Email: {szsz, wojciech.turek}@agh.edu.pl

***Abstract*—Large groups of animals or people are able to solve complex motion coordination problems without centralized planning or explicit communication. This observation forms the basis for a fully autonomous robots coordination method, which allows solving conflict situations involving groups of robots operating in complex environment. The method is based on modeling the most basic instinct of survival, which is fear. The fear model allows robots to determine priority in situations involving single robots and groups of robots. Disturbing the symmetry of the algorithm allows solving complex problems like deadlocks in narrow passages. The method presented in this paper has been tested in simulation and in reality, giving very promising results.**

***Index Terms*—robot coordination, fear model, mobile robots**

I. Introduction

The vision of utilizing groups of mobile robots to perform complex task fast and robustly has motivated researchers for several decades now. The most basic problem which must be solved to make such a system useful is motion coordination. Robots coexisting in the common environment must be able to avoid collisions and deadlocks and to achieve desired locations efficiently despite possible limitations in the available space.

Motion coordination is a very complex problem [1], even when many simplifications are made. Many of the existing solutions assume that robots' dynamics is meaningless and that robots move very slow compared to the time required for planning optimal actions. However, observed trends in hardware development suggest that the solutions to the problem of motion control and coordination should rather focus on handling very fast and possible inaccurate movements of robots as well as react adequately to unexpected situations. In real-life scenarios the guarantee of safety and robustness seems more important than global optimality of solutions.

Existing solutions to the problem are typically categorized according to the level of centralization [2] into fully centralized, decoupled and distributed. Centralized and decoupled planners require single entity to decide actions of particular robots. Distributed solutions typically require explicit communication protocol to agree on the plan.

While almost all the existing multi-robot motion coordination algorithms need some level of centralization and multi-entity planning, in the surrounding world many examples of fully autonomous motion coordination situations can be observed. In Fig. 1, hundreds of people share very limited environment, each heading into own destination. Humans create and adjust their motion plans autonomously, using knowledge and observation and applying common rules of behavior. In the presented pedestrian crossing no multi-entity planning takes place and explicit communication between entities is limited to minimum. Nevertheless the traffic flows smoothly and particular entities do not experience significant delays.

Figure 1. Fully autonomous motion coordination of hundreds of people at a pedestrian scramble in Tokyo[1]

The same phenomenons can be observed in different environments with different types of entities. Huge animal herds crossing a river in a narrow ford or drivers on an intersection without specified priority use fully autonomous planning and decision making process to determine the order of crossing. Application of common knowledge is in many cases sufficient to guarantee safe and robust motion.

These observations inspired us to design a fully autonomous algorithm for coordinating motion of mobile robots, which would use neither centralized planning nor explicit communication. The algorithm is based on autonomous path planning and real-time adapting to the situation using observation of other robots. The reactions for observed situations mimic selected behaviors of humans, like avoiding collisions with bigger or self-confident people and respect for organized groups. More complex situations are solved using commonly-known

Manuscript received August 11, 2015; revised December 17, 2015.

1 https://www.flickr.com/photos/aktugan

doi: 10.18178/ijmerr.5.2.96-102

rules or different phenomenons, like panicking in dangerous situations. The behaviors are inspired by one of the most basic instinct of survival, which is fear.

The details of the algorithm are presented in the third section, after the description of the existing solutions in the domain of mobile robots motion coordination. Following section presents the implementation of the algorithm and the results of the experiments conducted in simulation and with the use of a real multi-robot system.

II. Mobile Robot Motion Coordination

The problem of finding optimal trajectories for more than one mobile robot operating in a continuous space with continuous time is proven to be an NP-hard task [1]. Finding an optimal solution is possible only if strong assumptions are considered and is always a computationally complex task.

As in case of many other problems related to multi-robot systems, the motion coordination has been firstly addressed in fully centralized manner. Centralized planners have been developed since the eighties [3], when the priority-based approaches were first introduced. Trajectories for robots were calculated one after another, considering previously calculated ones as moving obstacles, which have to be avoided. The order of planning determined priorities of robots and could be the subject for optimization [2]. This kind of centralized planning algorithms assumed the planning takes place before the robots start to move and that the plan execution never fails.

Robots' dynamics constraints were rarely taken under consideration in the planning phase. In [4] authors proposed to represent a robots' trajectory as a parametric curve consisting of a sum of harmonics. The representation made it possible to use a numerical optimization. Another method was proposed in [5] where the A-star algorithm was used for constructing trajectories from small steps, which represented robots' dynamics capabilities.

All the centralized solutions with initial planning phase suffer from the same set of problems:

- Lack of resistance for plan execution failure,
- Inability of modifying parts of plan during execution,
- High computational cost,
- Single point of failure and need for collecting all necessary information in one place.

It seems that in real-life scenarios the centralized approach recalculating optimal solution misses the point. A solution calculated after minutes of processing will be in most cases far less valuable than robust, sub-optimal control method. Therefore more recent research in the area focuses on decentralized solutions.

A decentralized modification of the priority-based solution has been described in [6], [7]. Each robot plans its own path and detects possible conflicts. A communication protocol is used to determine which robot must yield and which can continue on the preferred trajectory.

In order to model limits in communication ranges the authors of [8] proposed introduction of a *coordinated network*, which is a set of robots located within specified range. The robots used a negotiation protocol to determine best possible solution of a conflict situation.

Decentralization of planning and control in multi-robot systems typically leads to better robustness, error resilience and ability to modify the conditions on runtime. In case of the motion coordination problem it also involves reduction of effectiveness because global optimal solutions cannot be found. What is more important distributed motion planning systems require exchanging significant amounts of information and strongly depend on the communication reliability.

Further increase of robots' autonomy in the process of coordinating motion can be achieved by removing explicit communication. Fully autonomous planning and collision avoidance algorithms do not suffer from any of the above-mentioned issues, however they can fail to generate efficient solution in particular situations. Basic method of this kind has been presented in [9]. It introduces a concept of *Velocity Obstacles* calculated by each robot using observation of other robots' movements. Generated *collision cones* are used for selecting safe vector of movements.

An extension of the algorithm, called *Reciprocal Velocity Obstacles* [10], introduces the before-mentioned concept of commonly-known rules, which are applied by all participants of motion. Using the assumption that all robots are executing the same algorithm made it possible to modify the algorithm itself, which resulted in smoother and shorter paths. Further extensions called *optimal reciprocal collision avoidance* [11] made it possible to solve motion coordination problems for hundreds of coexisting simulated agents. The algorithms are fully reactive, which guarantees robustness and flexibility. However, this simple solution cannot handle properly complex environments, like narrow door or long corridors with places for passing. More complex environments, which lead to conflict situations, require a method for disturbing the symmetry of the control algorithm - some robots must retreat to allow others to pass. Defining simple, reactive rules which would allow solving such situations has been the subject of work presented in [12]. The authors focused on the problem of conflict situations between two robots in narrow passages. The presented *aggressive competition* algorithm is based randomization of own behavior in order to select a robot which should pass first. After detecting a conflict situation both robots start to retreat till a random condition is met. The first to finish the retraction is considered more aggressive and passes first.

The *aggressive competition* algorithm proves that it is possible to solve complex motion coordination situations without explicit communication. Its greatest limitation is the inability to solve a conflict with more than two robots involved. The work presented in this paper proposes a more general algorithm which is focused on solving conflicts involving groups of robots in complex environments.

III. Fear Model

The designed method aims at providing a robust mobile robot motion coordination by using behavioral controllers executed autonomously by each of the involved robots. No explicit communication is used - all the decisions are based on autonomous observations only. The behaviors of robots are inspired by nature, mimicking the most basic instinct of survival, which is fear.

The proposed fear model is inspired by mechanisms observed among animals and people which allow solving complex coordination problems with simple behaviors of particular individuals. Let us consider the autonomous decision making process performed by a human being in the most basic coordination problem: decide whether to change speed or direction when another person is noticed on a collision path. It the other person looks self confident or aggressive typically one decides to perform actions in order to eliminate the risk of getting into close proximity. On the other hand, if the person looks uncertain or hesitant, one typically does not change planned actions assuming the other person will sidestep.

Another behavior which will be useful in the considered problem can be called panicking. When a cat chased by a noisy dog gets chucked into a blind alley, it turns around and becomes aggressive. Typically, when the dog notices the cat turning around and bristling its fur it immediately changes the judgment concerning its advantage and runs away.

People and animals often tend to form temporary groups when facing difficult situations. Fish form dense shoals to avoid being eaten and people follow big individuals while pushing through a crowd. People walking in a big group do not pay much attention for individuals who must stop or turn to let them pass. Groups of living creatures, while facing a danger, tend to compare the size of their own troop and the strength of its members with the enemies. Effects of that observation influence making the final decision - whether to run or attack.

More developed societies developed common rules, which are known and applied by all members. Every person is required to let other people get off a city bus before entering. What is more important, people leaving the bus can assume that people waiting to enter will let them out first.

Before selecting proper behaviors and designing the algorithm several assumptions and goals have been defined. The most obvious is safety - the algorithm cannot lead to any collisions. With the safety guaranteed, the main goal to achieve is robustness. Robots operate in unpredictable environment where unknown conditions and events can happen. Therefore they must be able to react dynamically to changing environment in order to complete given tasks. Second goal is to achieve effectiveness. Not only succeeding a job, but also doing it in the shortest possible time is needed. Because of the fact that many robots having different targets exist in same area, a possibility of deadlock cannot be overlooked. It means that they need to be equipped with appropriate counteraction behavior.

For the purposes of the presented algorithm we decided to implement selected behaviors and phenomenons, which allow solving most important problems of mobile robot coordination, namely:

- Autonomous path planning algorithm,
- Determining own and foreign fear factor for groups of robots,
- Reactive avoidance of more scary robots,
- The phenomenon of panic.

The path planning algorithm was based on the solution presented in [13] - each robot calculated the shortest path using graph model of the environment, which consisted of a list of checkpoints to reach.

We introduce an artificial feature used for determining which robot is scarier. It is represented by a single, floating-point value *k*, called *fear factor*, which is calculated as:

$$k_j = \sum_{i \in R} \left(\frac{D_{\max} - d_i}{D_{\max}} * \cos(\phi_{ij}) * f_i \right) \tag{1}$$

where:

$$\phi_{ij} = \begin{cases} \alpha_i - \alpha_j & \text{if } \left|\alpha_i - \alpha_j\right| \le \frac{\pi}{2} \\ \frac{\pi}{2} & \text{otherwise} \end{cases} \tag{2}$$

R is a set of robots, which are located within fixed range D_{max} from robot *j*, including robot *j*. d_i is the distance between robot *i* and *j*. f_i is the base fear factor of robot *i*, which can be observed by other robots. α_i and α_j represent the direction of robots *i* and *j*.

The base fear factor *f* is assigned during system initialization to each of the robots. It is selected randomly from the range of [1.0, 1.01], which means that it is almost equal for all robots. The small differences are crucial in solving 1-to-1 conflicts but become meaningless when more robots are involved. In further extensions of the algorithm it can reflect relative importance of particular robots.

The observed fear factor of every robot is the sum of fear factors of all robots forming a group. Robots are considered to form a group when the angle between their directions is less than 90 degrees and the distance between them is within particular range. Calculated fear factor is the basis for making autonomous decisions by all robots. The possible behaviors are:

- When the planned path is clear within the considered range, the robot moves directly to the next checkpoint.
- When a more scary robot is detected on the way, the best collision-avoiding maneuver is calculated.
- When a less scary robot (which was supposed to give way) is detected in direct proximity, a robot stops immediately to avoid collision.
- When the reason for emergency stopping does not disappear for specified amount of time, a deadlock situation is identified. In such case the robot

reduces its fear factor to become less scary than the obstacle, which forces it to retrieve from the deadlock situation.

The collision detection algorithm is based on observation of other robots' movements and prediction of their next states in particular moments in time. As shown in Fig. 2, future robots locations are estimated using constant time span. If the distance between the locations of two robots in the same future time is less than specified constant, the situation is considered dangerous and the robot with smaller fear factor must activate the collision avoidance behavior.

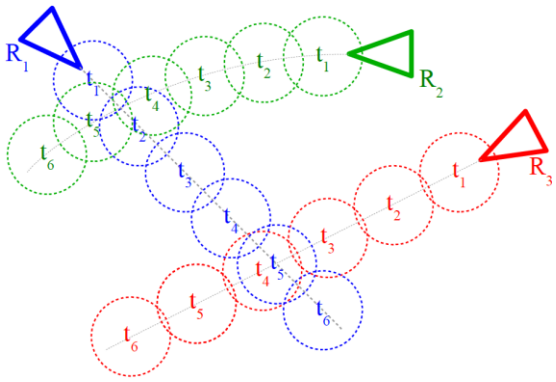

Figure 2. Visualization of the collision detection algorithm. Robots R_1 and R_3 can collide in time t_4, while the motion of robots R_1 and R_2 is safe.

The collision avoidance algorithm is based on random sampling of the control space. Random values of speed and turning angle are tested against the observed situation. The selected solution must not cause any collisions and should bring the robot closer to its next checkpoint. If none safe solution is found, the robot stops in current location. These simple behaviors turned out to be sufficient for solving complex robot coordination problems.

IV. Implementation and Experimental Results

A process of designing the algorithm was done in an iterative manner. Every new behavior and improvement had to be verified in testing environment to prove its correctness, find and fix any misbehavior and measure its effectiveness. In order to make the whole process as fluent as possible, a custom testing system had been designed and implemented. It was used to prepare two different experimenting environments. One was purely software simulation based, while the other used real hardware robots.

A. Software Architecture

The main difficulty in developing two-environment testing system was to provide an abstraction over two possible robot interfaces. An assumption has been made that a single program containing the whole algorithm had to be able to control both simulated and hardware robot without any modifications. According to that any change in algorithm's behavior could be immediately tested in two ways. In order to meet this requirement the testing system had been divided into set of layers as shown in Fig. 3. Modules common to both experimental environments could be therefore separated and reused.

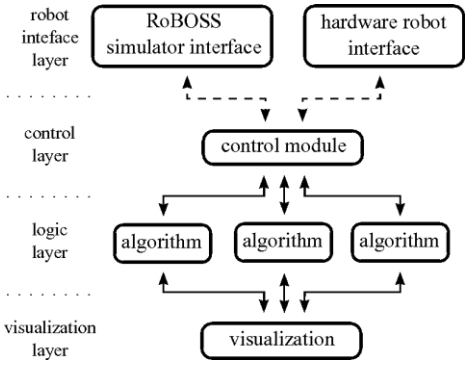

Figure 3. Software architecture diagram

Robot interface layer is responsible for communication with robot's actuators and sensors. This is the only part of the system that changes when environments are switched. There are two possible modules to be used here - RoBOSS simulator's interface [14] and hardware robot's interface. Both of them contain logic that translates and dispatches messages between a robot (real or simulated) and the control layer.

Control layer is used to spawn and control algorithm modules. It has only one module which supervises members of logic layer.

Logic layer contains one or more algorithm modules. Algorithm module is an operating system process that controls one robot - either real or simulated. In case of hardware-based environment only one process is spawned because each robot has a separate control system. In the second case, when simulated context is used, multiple instances are created - one for every robot present in the simulation.

Visualization layer contains a module used to monitor and visualize progress of an experiment and present current state of all robots.

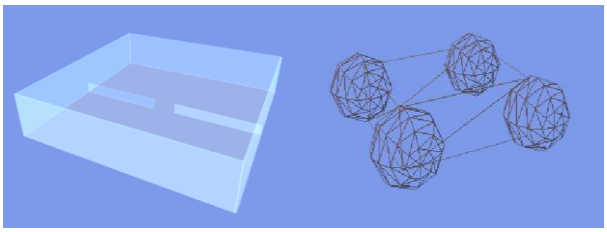

Figure 4. Visualization of an exemplary simulation environment and robot's model used in the experiments

B. Simulation Experiments

The main element of simulation environment is RoBOSS [14] simulator. It is an universal and advanced tool designed for simulating movement and interaction between mechanical structures made of rigid bodies. It offers three-dimensional simulation of kinematics and dynamic of robots and the possibility of creating control programs in convenient programming language.

An environment in which robots operate can be freely customized in XML files. Exemplary simulation's environment and robot's model are presented in Fig. 4.

Several series of experiments have been performed. In order to measure algorithm's effectiveness, comparisons with a second algorithm using constant priorities were made. In this reference algorithm, priorities for all robots are randomized at the beginning and kept for the whole run.

1) 3 vs 1 in open space: First experiment was used to verify the very basic concept of the algorithm - is a troop of robots able to obtain a right of way over a single individual. Each of four robots involved in this test case had to reach the opposite wall (Fig. 5). Since each member of the group of three robots maintained the same speed and direction, their fear factor reaches significantly higher level that the fear factor of the single robot. It means that the single robot should always give way to the group.

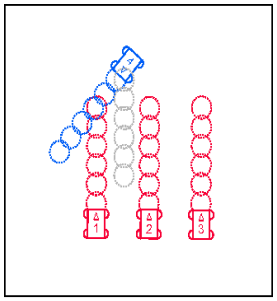

Figure 5. *3 vs 1 in open space:* one frame of the visualization

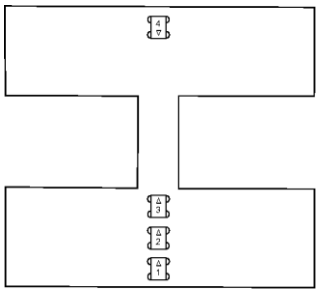

Figure 6. *3 vs 1 in a narrow corridor* test case map

TABLE I. BASIC RESULTS FOR *3 VS 1 IN OPEN SPACE* TEST CASE

	Random priorities	**Fear based**
Runs	100	100
Succeeded runs	100	100
Failed runs	0	0
Effectiveness	100%	100%
Average duration of succeeded run [s]	19.48	20.09

Observations of robots' movements confirmed the correctness of the basic fear concept. During each run the single robot decided to avoid the group, choosing turning either left or right. Since this is a really simple test case with no additional obstacles, results of measurements done for both algorithms are very similar, as shown in Table I. Fear-based approach in this case does not have any advantages in terms of performance.

2) 3vs1 in a narrow corridor: This test case contains more complicated environment as shown in Fig. 6. Now a group of three robots is aligned in line and its members move one after another. The fourth, single robot is placed at the opposite side. They all are supposed to travel via a narrow corridor located in the middle. Because the corridor is not wide enough for two robots to move side by side, a problem of effectively passing this obstacle arises. The best possible solution is when the group forces the single robot to wait for them to pass (Fig. 7 A). Moreover, there is a possibility of a deadlock when a robot with low priority is trapped between others (Fig. 7 B).

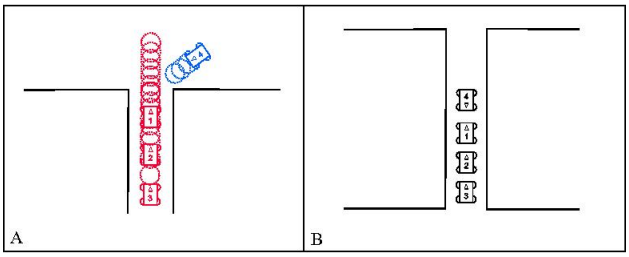

Figure 7. *3 vs 1 in a narrow corridor* test case situations

TABLE II. BASIC RESULTS FOR *NARROW CORRIDOR WITH A PASSING BAY* TEST CASE

	Random priorities	**Fear based**
Runs	100	100
Succeeded runs	59	100
Failed runs	41	0
Effectiveness	59%	100%
Average duration of succeeded run [s]	50.19	45.44

As shown in Table II, randomized algorithm did not succeed in all runs. As predicted earlier, nearly half of experiments failed because of a deadlock. On the other hand fear-based approach showed its advantage here, having all runs finished properly in around 45 seconds. Those are the cases in which a group had the right of way over the single robot. There is also a small set of runs that took roughly 65 seconds (Fig. 8). That happened when only two robots from the group went through the corridor, while the third one was forced to step back.

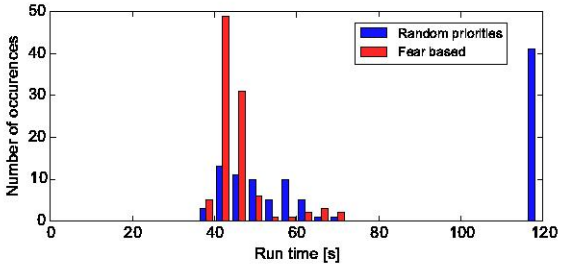

Figure 8. *3vs1 in a narrow corridor* test case time histogram

3) Narrow corridor with a passing bay: The aim of this experiment was to verify the algorithm's deadlock solving mechanism. Given environment has a form of a long narrow corridor with one passing bay located asymmetrically (Fig. 9).

There are only two robots moving in opposite directions. In case of using constant priorities algorithm,

success of a run depends on the initial randomization. When the right-hand robot gets higher priority, it is able to force its enemy to move into a passing bay (Fig. 9 A and 9 B). Otherwise the left-hand robot pushes the other to a dead end at the right side (Fig. 9 C), which leads to a deadlock (Fig. 9 D). Fear based algorithm should be able to solve this situation in a way shown in Fig. 9 E and Fig. 9 F.

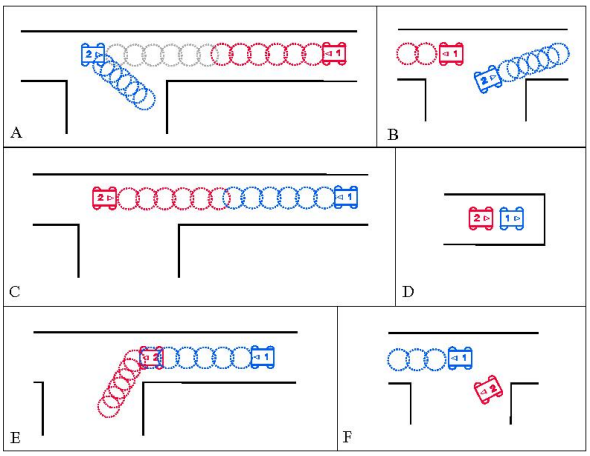

Figure 9. *Narrow corridor with a passing* bay test case situations

TABLE III. BASIC RESULTS FOR NARROW CORRIDOR WITH A PASSING BAY TEST CASE

	Random priorities	**Fear based**
Runs	100	100
Succeeded runs	50	98
Failed runs	50	2
Effectiveness	50%	98%
Average duration of succeeded run [s]	31.58	32.72

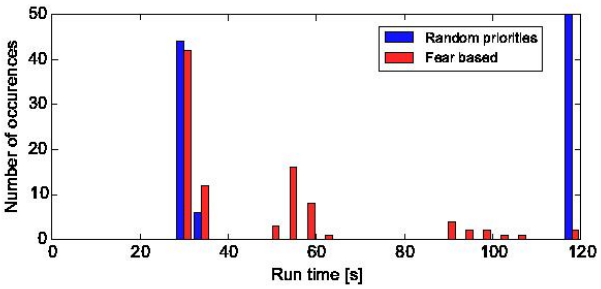

Figure 10. Narrow corridor with a passing bay test case time histogram

Results in Table III show that the randomizing algorithms succeeded in exactly 50% of runs, when the hand robot receives higher priority. Fear based equivalent shows its ability to solve deadlocks, which effects in nearly 100% success rate. Its results are grouped into several sets (Fig. 10). This is caused by a fact that the left-hand robot after decreasing its fear factor and moving back, does not always use the passing bay, which therefore resets the situation to the initial state.

Several other configurations have also been tested in the simulation showing very promising features of the approach.

C. *Experiments with Hardware Robots*

Preliminary experiments performed with hardware robots involved two scenarios: troop avoidance in the open space and narrow corridor with a passing bay (Fig. 11). We used several CAPO robots [15], which are equipped with powerful on-board computer and a laser scanner.

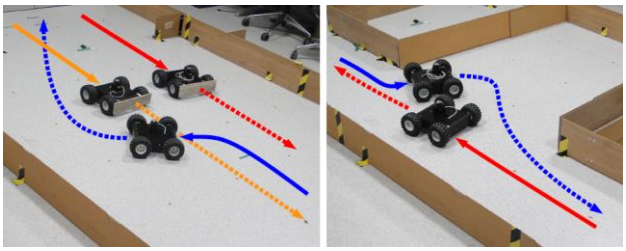

Figure 11. On the left: troop avoidance in the open space; on the right: Narrow corridor with a passing bay

The experiments did not include autonomous localization of other robots and identification of their fear factor - these information were given. With this simplification the system was able to execute the coordination algorithms similarly as in case of simulation. Further work is required for making the control algorithms fully autonomous.

V. CONCLUSIONS AND FURTHER WORK

Presented results show that it is possible to solve coordination problems involving groups of robots in complex environments without centralized planning and without explicit communication. Mimicking the behaviors of living creatures can lead to robust and efficient coordination algorithms.

Further research in this area will include widening the range of possible behaviors and representing common, social knowledge in order to solve more complicated scenarios.

ACKNOWLEDGMENT

The research leading to these results has received founding from the Polish National Science Centre under the grant no. 2011/01/D/ST6/06146.

REFERENCES

[1] P. Surynek, "An optimization variant of multi-robot path planning is intractable," in *Proc. Twenty-Fourth AAAI Conference on Artificial Intelligence*, 2010, pp. 1261–1263.
[2] M. Bennewitz, W. Burgard, and S. Thrun, "Finding and optimizing solvable priority schemes for decoupled path planning techniques for teams of mobile robots," *Robotics and Autonomous Systems*, vol. 41, no. 2, pp. 89–99, 2002.
[3] M. Erdmann and T. Lozano-Perez, "On multiple moving objects," in *Proc. IEEE International Conference on Robotics and Automation. Proceedings*, Apr 1986, vol. 3, pp. 1419–1424.
[4] P. Gallina and A. Gasparetto, "A technique to analytically formulate and to solve the 2-dimensional constrained trajectory planning problem for a mobile robot," *Journal of Intelligent and Robotic Systems*, vol. 27, no. 3, pp. 237–262, 2000.
[5] W. Turek, "Motion coordination method for numerous groups of fast mobile robots," *Recent Advances in Intelligent Information Systems*, pp. 721–731, 2009.

K. Azarm and G. Schmidt, "Conflict-free motion of multiple mobile robots based on decentralized motion planning and negotiation," in *Proc. IEEE International Conference on Robotics and Automation*, Apr 1997, vol. 4, pp. 3526–3533.
[6] Y. Guo and L. Parker, "A distributed and optimal motion planning approach for multiple mobile robots," in *Proc. IEEE International Conference on Robotics and Automation*, 2002, vol. 3, pp. 2612–2619.
[7] C. Wei, K. Hindriks, and C. Jonker, "Multi-robot cooperative pathfinding: A decentralized approach," in *Modern Advances in Applied Intelligence*, Springer, 2014, vol. 8481, pp. 21–31.
[8] R. A. Brooks, "A robust layered control system for a mobile robot," *IEEE Journal on Robotics & Automation*, vol. 2, no. 1, pp. 14-23 1985.
[9] J. van den Berg, M. Lin, and D. Manocha, "Reciprocal velocity obstacles for real-time multi-agent navigation," in *Proc. IEEE International Conference on Robotics and Automation*, 2008, pp. 1928–1935.
[10] J. van den Berg, S. Guy, M. Lin, and D. Manocha, "Reciprocal nbody collision avoidance," in *Robotics Research, ser. Springer Tracts in Advanced Robotics*, Springer, 2011, vol. 70, pp. 3–19.
[11] R. Vaughan, K. Støy, G. Sukhatme, and M. Mataric, "Go ahead, make my day: robot conflict resolution by aggressive competition," in *Proc. Intl. Conf. on Simulation of Adaptive Behavior*, 2000, pp. 491–500.
[12] W. Turek, K. Cetnarowicz, and W. Zaborowski, "Software agent systems for improving performance of multi-robot groups," *Fundam. Inf.*, vol. 112, no. 1, pp. 103–117, Jan. 2011.
[13] W. Turek, R. Marcjan, and K. Cetnarowicz, "A universal tool for multirobot system simulation," in *Knowledge-Driven Computing. Springer Berlin Heidelberg*, 2008, vol. 102, pp. 289–303.
[14] S. Szominski, K. Gadek, M. Konarski, B. Blaszczyk, P. Anielski, and W. Turek, "Development of a cyber-physical system for mobile robot control using erlang," in *Proc. Federated Conference on Computer Science and Information Systems*, Sept. 2013, pp. 1441–1448.

Michał Konarski studied computer science at the AGH University of Science and Technology in Krakow. He received his MSc degree in 2014 with a thesis about behavioral algorithm for mobile robot motion coordination. His main interests are software engineering, mobile robots and distributed systems.

Szymon Szomiński studied computer science at the Cracow University of Technology and obtained his degree in 2010. Currently, he study computer science at the AGH University of Science and Technology.

Wojciech Turek was born in Krakow, Poland in 1981. He received his Ph.D. at the AGH University of Science and Technology in Krakow in 2010. He has a significant expertise in the field of robotics, multi-agent systems and Erlang technology applications. He has published more than 20 papers presenting applications of Multi-Agent Systems, mostly in the field of mobile robotics. He has been involved in several national and international projects as investigator and coordinator. His researches focus on applications of the Erlang technology for creating robot control and management systems using the agent paradigm.