Rafał SIKORSKI[1,2]

Warsaw University of Technology (1), Institute of Aviation (2)

doi:10.15199/48.2019.12.57

Flexible multi-agent system for mobile robot group control

Abstract. *Constant developments in robotics field lead to popularization of using mobile robots in different fields of application. Currently developed solutions concentrate mainly on local robotic systems control. Because of wide spread robot use aside from reactive control a robotic system must have social abilities (task planning and peer-to-peer negotiations) that are needed for efficient interacting of many heterogeneous units in one environment. This paper proposes using Multi-Agent systems (MAS) as main model for building software systems for mobile robot groups control. The approach limits basic shortcomings recognized in past applications such as fault tolerance, openness and scalability. Each autonomic agent controls one mobile robot communicating with other agents using ACL messages. The system was realized based on JADE, a multi agent Framework.*

Streszczenie.*Ciągły rozwój w dziedzinie robotyki doprowadził do popularyzacji wykorzystania mobilnych robotów w różnych dziedzinach zastosowań. Aktualnie opracowywane rozwiązania koncentrują się głównie na lokalnej kontroli robotów. Ze względu na szerokie zastosowanie robotów poza sterowaniem reaktywnym system robotyczny musi mieć umiejętności społeczne (planowanie zadań i negocjacje peer-to-peer), które są potrzebne do efektywnego współdziałania wielu heterogenicznych jednostek w tym samym otoczeniu. Praca proponuje zastosowanie systemów wieloagentowych (MAS) jako podstawy budowy systemu informatycznego przeznaczonego do kontroli grup robotów mobilnych. Podejście zapewnia odporność na błędy, otwartość i skalowalność. Każdy z autonomicznych agentów steruje jednym robotem mobilnym oraz komunikuje się z innymi agentami korzystając z wiadomości ACL. System zrealizowany został w oparciu o JADE, platformę Multi-Agentową.* ***(Elastyczny wieloagentowy system sterowania grupami robotów mobilnych).***

Keywords: Mobile robots, Multi Agents, Java Agent Development Framework
Słowa kluczowe: Roboty mobilne, Multi Agent, JADE

Introduction

Modern robotics is dynamically growing field. New findings help in mitigating hardware limitations like battery capacity, relatively small electric motors power comparing to their mass, cost of sensors just to name a few. Alongside of hardware new software control solutions are constantly developed. All those developments caused that mobile robots are more widely used and more often we must deal with not just one robot, but a group of devices that must work simultaneously, communicate and cooperate with each other.

Except of low-level control algorithms, most of the control applications have their beginnings in field not directly connected with robotics for example in desktop applications, cloud computing [1]. To meet growing demands on robots' control, there is a need for solutions integrating new programming trends into safe and reliant multi robot systems. Created solutions must solve problem of reactive control of single robot and long-term planning and coordinating tasks in multi robot environment. A lot of work in the field of robotics concentrates on developing low level drivers designed for specific hardware solutions largely ignoring the problem of managing large groups of autonomous robots.

The paper describes a solution, built using multi agent system (MAS) as high-level robot control system. System uses previously developed tools like local path planning based on Vector Field Plus (VHF+) [9], Adaptive Monte Carlo Localization algorithm (AMCL) [7, 8], wave front path planning [11] and specific hardware drivers. Proposed system incorporates them in top layer multi agent system for global intelligent task planning and robot interactions. Inherent strengths of MAS systems as decentralization and scalability may solve some of main problems of currently used monolithic solutions. Using social purpose-oriented agents, we can ensure solid, scalable and open architecture of mobile robot control platform that will keep key features of planning capability and quick reacting on local obstacles.

Mobile Agents

There is no one clear definition of software Agent. Generally, agent is described as special component of software that is autonomic, behaves as a human in servicing clients and seeks to realize its personal goals [2]. True agent is autonomic, has social capabilities, is reactive and pro-active [3]. Agent is autonomic in the sense that he must be able to act on its own without interacting with other agents or humans. That implies that any agent has some level of intelligence. Reactivity means that agent is a being localized in real or virtual environment and has ability of interacting with it. From computational perspective this agent feature has crucial importance, it helps in differentiating between agents and other kinds of software like modules and objects. Pro-active agent constantly observes the environment and based on that data tries to perform some tasks based on obtained information. In software development process that need means that there is a need for including of central agent control thread. Independently of used agent implementation method important thing is that agent isn't an object passively waiting for actions performed on it but actively seeks to achieve the main goal.

Developed system is based on popular Java Agent Development Framework [4]. The platform is fully FIPA [5] compliant system designed to be a middleware for multi-agent systems based on software agent paradigm. JADE supplies graphical tools that support implementation, debugging and deployment phases. It allows for building systems distributed across multitude of machines that ideally complies with requirement of system built from group of autonomous mobile robots. Because JADE is based on very popular Java language, creating new functionalities and integration of already developed drivers and methods is relatively easy.

The architecture of MAS

Instead of building one big specialized application that controls the whole system, approach proposed in the paper splits system functionality between number of smaller self-reliant software agents. Each agent assigned to one, particular robot, controls its local behavior and can communicate with rest of the robot population for cooperation purposes.

Each agent is fully autonomous, self-reliant and proactive. Enclosed in its structure, agent has implemented basic localization, path planning and decision-making

algorithms. That allows for individual robot functioning even without any other system components present. Agent uses multi robot platform Player/Stage [6]. The platform provides wide array of software drivers for popular mobile robots and sensing devices. Thanks to that, multi agent system is elastic and allows for easy adding of new types of robots to the environment without the need of making modifications to existing software agents.

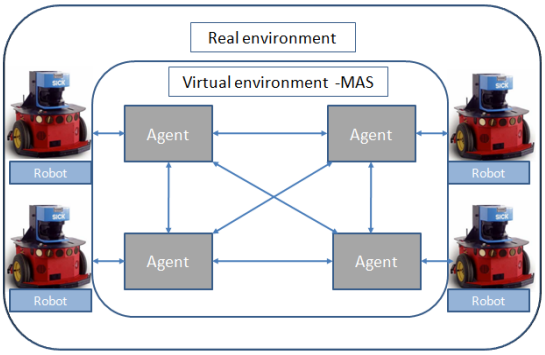

Fig. 1 Robot control MAS architecture

On top of that, each virtual agent has capability of communicating with agents assigned to other robots inside of the environment. The communication between agents is realized with asynchronous ACL messages. Thanks to that the system allows for high level task planning and group action coordination.

Robot low level control

Each software agent that is a part of proposed MAS solution contains set of drivers that are responsible for low level robot control. That solution ensures that even one robot controlled by assigned to it software agent is capable of autonomic functioning according to its paradigms. The most important low-level drivers used in the solution are shortly described below.

Adaptive Monte Carlo localization (AMCL) driver

Current position of the robot is obtained with use of Adaptive Monte Carlo localization method [7, 8]. AMCL driver represents probability distribution with help of particulate filter. Filter is adaptive, it dynamically adjusts particles number. When position of the robot is unknown the number of particles is increased, when position is well known the number of particles is decreased. Robot equipped with laser scanner, when moving supplies new data that helps in localization process.

Sample localization process is shown in figure 2.

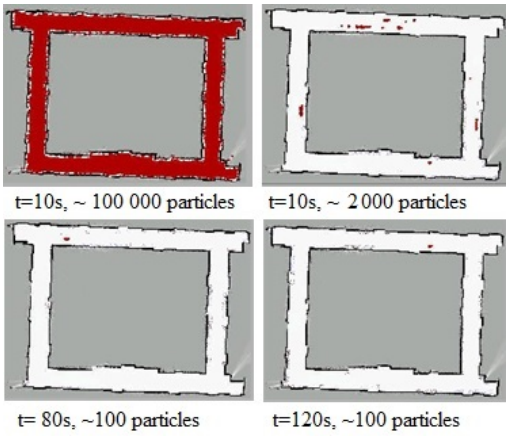

Fig. 2 AMCL localization process

Vector Field Histogram Plus (VHF+)

Collision avoidance was achieved using Vector Field Histogram Plus (VHF+) method [9], that is an evolution of VHF algorithm first presented by Borenstein and Korem [10]. Method allows robots to avoid obstacles in real time with keeping current movement direction. VHF+ algorithm builds a model of robot surroundings as two-dimensional Cartesian pole histogram. Model is constantly updated using sensors data. In VHF+ method we can distinguish three main stages. In first software builds two-dimensional pole histogram (fig. 3), it is created based on recorded distances to obstacles.

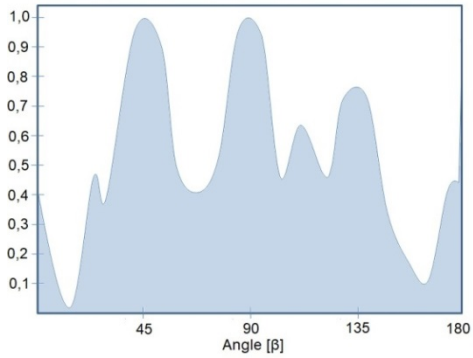

Fig. 3. Sample vector field histogram for 180 segments

X axis of histogram plot describes angle in degrees and Y gives sums of elements of vector describing segment of space. Histogram is built based on distance measurement. Algorithm splits observable area to number of equal segments. Next for each segment driver calculates angle value between the segment and center of active area that represents placement of the robot. The angle is calculated according to equation:

$$\beta_{i,j} = tan^{-1}\left(\frac{y_0 - y_j}{x_i - x_0}\right), \tag{1}$$

Where: x_i and y_j are placement of currently analyzed map segment, x_0 and $y_0$are coordinates of robot placement. Next step is to calculate value of vector element representing one segment of active area, chosen based on calculated angle. Value of vector is calculated according to equation:

$$m_{i,j} = c_{i,j}^2\left(a - bd_{i,j}^2\right), \tag{2}$$

where i and j mean currently considered map cell, c is the value inside cell, a and b are constants, and d is distance between current cell and center point of the robot.

Second main step of VHF+ algorithm creates binary polar histogram. It is achieved by setting threshold value, if pole histogram line falls below threshold value it has zero value on binary plot. If histogram value is greater than threshold, value of one is assigned to that line. Third step creates masked histogram plot taking into consideration robot kinematic. That solution allows for choosing optimal route for robots with constraints for example for robots not capable of performing sharp turns.

Lastly, algorithm identifies valleys in prepared binary histogram plot, that means lines of histogram with zero values. Angle values corresponding to valleys are identified as potential new optimal obstacle avoidance orientation for robot and further movement is changed appropriately.

Algorithm allows for including robot physical dimensions in calculations. It is achieved by extension of map cells including detected obstacle by radius of the circle corresponding with robot dimensions. That radius can be increased by set value that additionally accounts for robot

kinematic. That operation is needed for proper collisions avoiding.

Wave front driver

The Wave front driver realizes global path planning of mobile robot. After obtaining new destination for the robot, the path is planned from current position of the robot, that is determined with subordinate AMCL localization driver. The algorithm first creates configuration space dividing provided map into set of cells [11].

When planner gets new destination, the path is determined starting from the target destination map cell. Values are assigned first to cells adjacent to target and next to the cells with increasing distance from destination resembling the front of a wave on the water's surface. The following formula is used to assign value to each cell [12]:

(3)

$$map(i,j) = \begin{cases} min\big(neighbourhood(i,j)\big) + 1 & EmptyCell \\ NothingObstacleCell \end{cases}$$

Here i, j are coordinates of the cell. Neighborhood (i,j) represent the cell adjacent to the cell (i,j). When values are assigned to all cells, robot starts movement choosing the path consisting of cells with lowest cost values assigned to them until reaching the target.

Platform implementation

Combining those methods with available hardware drivers provided with Player software (driver p2os), single simulated robots is capable of most basic functioning of localizing its position, path planning and real-time collision avoidance. On top of that proposed system assigns software agent to each robot for high level reasoning, group coordination and task assignment. Simplified architecture of the system is presented on below figure:

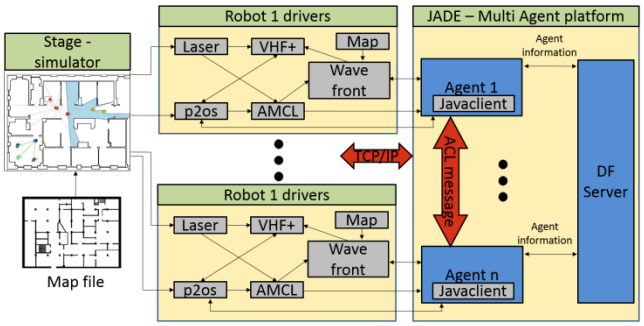

Fig. 4 Elastic multi-agent mobile robot platform architecture

Each mobile robot has one software agent that is assigned to it. Agent is constantly communicating with low level drivers allowing for real time control of the robot. Thanks to information procured from AMCL driver, Agent knows actual position of the robot in the environment and can reason based on that information. Data provided with use of Wave front method returns to agent information about estimated shortest route to assigned target position. With this information the system is capable of coordinating actions with rest of robot population and assigning the tasks to the robots that can perform them most efficiently. Agent can tell the robot to move to new target position. Then the system automatically calculates shortest possible route from current position to new target as a set of straight-line sections. When moving to the target, robot tries to follow planned route but if it comes across obstacle that is not part of provided map file it can correct its route to avoid it and still arrive to planned position. Additionally, Agent has full access to provided by Player platform robot hardware drivers. That allows direct access to all sensors and actuators integrated with the robot.

Platform is built in such a way that it allows for functioning of many heterogeneous robots in the same environment but is fully functioning also for single robot. To add new robot to the group we just create new agent instance to the JADE platform. When created, new agent automatically registers itself into JADE platform Directory Facilitator (DF) Server. From that point it is included in all platform wide actions, thanks to communicating with other agent present in the system with ACL messages.

Communication between platform Agents is based on FIPA-ContectNet protocol.

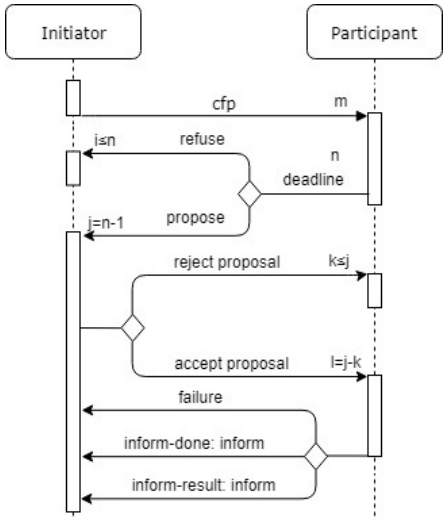

Fig. 5 FIPA-ContaractNet communication protocol [13]

Big advantage of built system is its openness, scalability and fault tolerance. Each agent populating the platform can have its own internal logic governing its actions. Each agent can be modified without impacting whole system functioning. As stated before system allows for easy integration of new robots or removing part of the population. Adding new mobile robot with prepared earlier function code can be done just by switching the robot on, creating new instance of the Agent to JADE platform. Newly started agent automatically registers itself into DF server and from that point actively participates in all social activities.

System is fault tolerant because each agent is separate entity and can function independently of whole system population. Whole system is distributed between mobile robots. Fault occurring on any of the system components may incapacitate single robot but will not impact functioning of rest of the robot population. The MAS is fully distributed system, there is no one crucial system element.

Simulation software

Testing of the functioning of the platform was done with use of robot simulation software Stage.

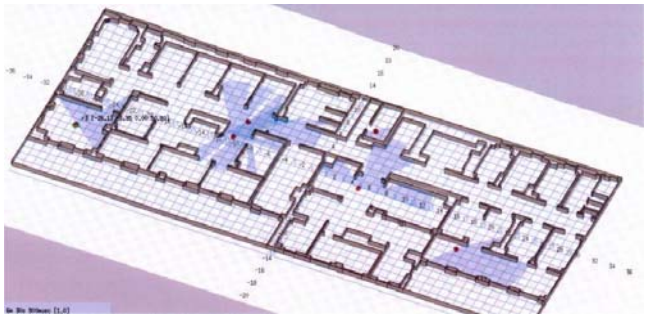

Fig. 6 Stage Multi robot simulation environment

Stage [5] is simulation software designed specifically for use in testing multi robot systems. It provides virtual environment for purpose of simulating robots, sensors and different objects that robots can sense and interact with them. It provides useful compromise between very elaborate, exact robot simulators and minimalistic simulations. Stage is realistic enough to allow for transferring developed control software directly to physical robots, additionally being fast enough to allow for testing of big populations of robots.

Fig. 7Pioneer 2DX with Sick LMS200 laser scanner

Tests were done simulating group of popular Pioneer 2DX mobile robots equipped with SICK LMS200 laser scanner sensor (fig. 7). The choice was dictated by good support of simulated hardware by both simulation software and Player platform hardware drivers support.

Functional Testing of implemented MAS

To check functioning of the proposed system, series of different simulation scenarios were performed. Firstly, testing had to prove that single robot controlled by software agent is capable of correct functioning inside simulated environment. Secondly, populating the simulation with group of autonomous robots, the capabilities of social interactions between them and coordinating their functioning had to be tested.

Single robot tests

Testing of single mobile robot controlled by software agent functioning inside simulated environment largely comes to proving, that it can move between assigned positions with avoiding of potential collisions with other objects on the way. To perform this task the robot must be able to correctly estimate its current position, plan the optimal route to assigned new position and when moving to the target avoid any dynamic obstacles not being part of predefined map.

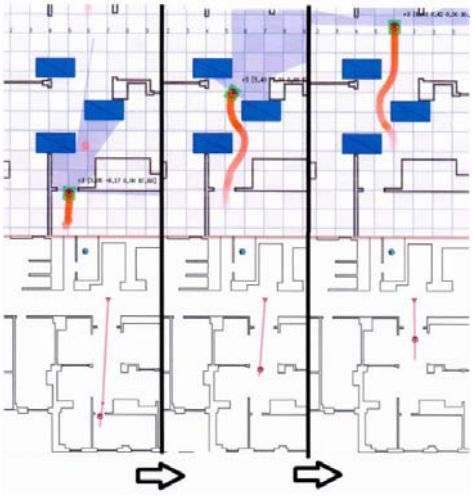

Fig. 8 Simulated robot dynamic obstacle avoidance

To check correctness and effectiveness of functioning of localization and path planning with local obstacle avoidance, several obstacles not being a part of environment map were introduced at random locations.

On figure 8 we can see example of path planning with obstacles on the way. After assigning new target position to the mobile robot waveform driver using current robot position estimation data provided by AMCL, establishes shortest route to the target and movement starts. The path is planned as set of straight-line segments that give shortest route to target based on the map layout. When obstacle, not being part of the map, is met on the route, robot actively avoids them using VHF+ method.

Bottom half of figure 8 shows robot position estimated based on sensor data and planned route. Top half of the figure shows actual position and route of robot inside simulation.

As tests shown, when robot senses dynamic obstacle on its planned route it will avoid it and reach target position. The system is built in a way that allows for avoiding dynamic obstacles including avoiding potential collisions with other robots, keeping target position in memory. Tests showed that robots being controlled by software agents possess enough ability to successfully navigate the environment.

Robot group testing

To check social capabilities and high-level task distribution functioning of the system a typical scenario of leader-follower was tested using many robots. Figure 9 shows sample simulation with 20 robots populating the environment.

Fig. 9 Simulation of 20 robots

Part of the robots populating simulation had “wanderer” type behavior. Their main task was, to randomly explore the environment actively avoiding obstacles. Rest of robots had a task of pursuing them. At each time only one “pursuer” robot was to be assigned to each Wanderer.

The robot pursuing each “wanderer” was chosen in process of open auction based on FIPA-ContaractNet communication standard. All “pursuer” robots using ACL messages, send their calculated route to all of the “wanderers” that can make decision basing on that information. Only the robot that has shortest route to catch up with target robot starts movement. The whole process is cyclical because of the dynamic character of the task.

"Wanderer" robots are in constant movement. Tests were done with big robot populations up to 20 simultaneously functioning robots. Figure 10 shows simplified system components interactions.

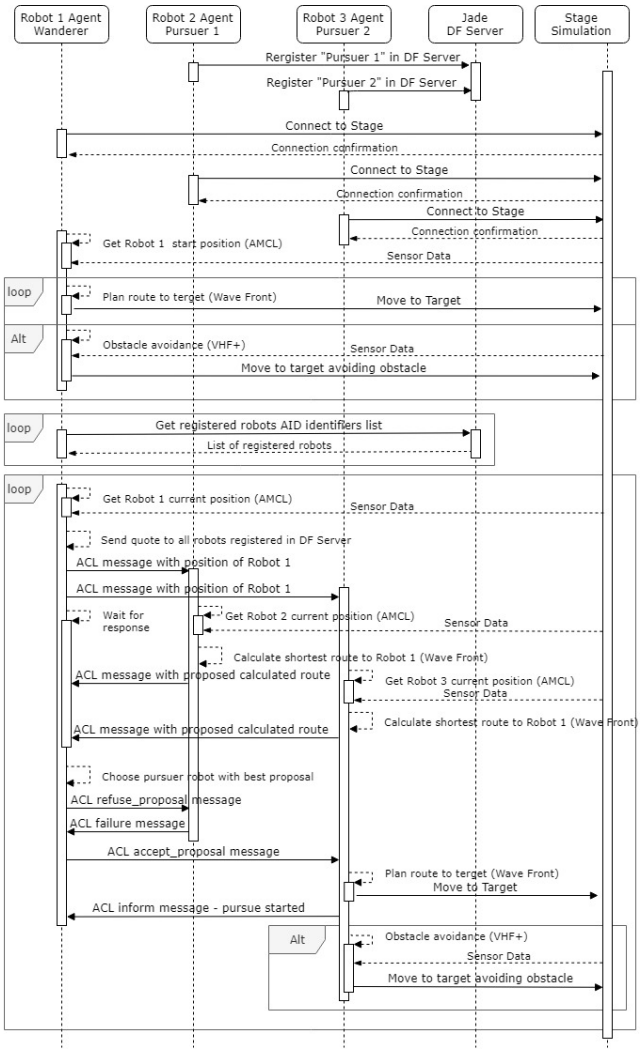

Fig. 10 UML diagram - system elements interaction

Tests proved that robots controlled by software agents can successfully communicate and coordinate their actions. Series of simulations with different number of different type agents were performed. In each case robots were able to correctly negotiate task assigning and efficiently performed programmed functions.

The big advantage of the MAS system is its openness and elasticity compared with other system approaches. Adding or removing new robots is relatively easy. New agent introduced to the simulation automatically registered itself in DF Server accessible by all other agents. From that point it started actively participating in all platform-wide actions. Similarly removing any of the robots' form environment, for example simulating fault, did not disturb functioning of the rest of MAS.

Performed tests included adding new robots and simulating critical failures of already registered robots without stopping of the simulations. As tests proved the system is fully distributed. There is no one crucial element of proposed MAS. Potential fault of any of the system components or introduction of totally new agent type does not disturb function of the system. Newly added robots, after introduction to simulation started performing their tasks and actively took part in task assigning process. Robots with simulated fault after stopping were simply excluded from task assigning auction and didn't negatively affect rest of population.

Conclusion

This paper proposes using MAS architecture as a basic framework for multi robot control systems. For that purpose, a complex MAS system responsible for local robot control, communication and high-level reasoning was created and tested. Software Agents populating the platform were able to efficiently control assigned robots and coordinate their functioning for performing platform wide tasks requiring task assigning and social interactions between robots. Performed tests confirmed assumptions about potential benefits of the MAS approach as scalability, openness and fault tolerance.

Agent oriented programming and MAS systems are the future of software engineering. Object-oriented programming does not provide all the benefits that are inherent to software agent paradigm. Relatively easy implementation of MAS systems and integrating them with now widely available and proven low-level mobile robots' drivers makes the proposed approach a very attractive alternative to currently used more rigid object-oriented systems.

Author: *Rafał Sikorski*
Institute of Theory of Electrical Engineering, Measurement and Information Systems, Faculty of Electrical Engineering, Warsaw University of Technology, Pl. Politechniki 1, 00-661 Warsaw
Lab Maintenance and Development, Institute of Aviation,
Al. Krakowska 110/114, 02-256 Warsaw
E-mail: rafal.sikorski.dokt@pw.edu.pl

REFERENCES

[1] Rogoza W., Zabłocki M. Grid computing and Cloud computing In scope of JADE and OWL based Semantic Agents – A Survey, *Przegląd Elektrotechniczny*, 90 (2014), nr 2, 93-96
[2] Wooldridge M., Jennings N.R.: Intelligent Agents: Theory and Practice, *The Knowledge Engineering Review*, 10(2) (1995), 115-152
[3] Bellifemine F., Caire C., Greenwood D.: Developing Multi-Agent Systems with JADE, John Wiley&Sons Ltd., 2007
[4] Java Agent Development Framework Home page. URL: *http://jade.tilab.com*
[5] The Foundation for Intelligent Physical Agents Home page. URL: *www.fipa.org*
[6] The Player Project Home page. URL: http://playerstage.sourceforge.net/
[7] Frank, D., Burgard, W., Dellaert, F., Thrun, S.: Monte carlo localization: efficient position estimation for mobile robots. In: Proceedings of the National Conference on Artificial Intelligence and the Innovative Applications of Artificial Intelligence Conference. AAAI '99/IAAI '99, American Association for Artificial Intelligence, pp. 343–349 (1999)
[8] Fox, D.: Adapting the sample size in particle filters through kld-sampling. I. J. Robotic Res. 22(12) (2003), 985–1004
[9] Jaskot, K., Knapik K.: Budowa mapy otoczenia z wykorzystaniem grupy robotów mobilnych, *Przegląd Elektrotechniczny*, 90 (2014), nr 12,30-39
[10] Borenstein J., Koren Y.: Histogramic in-motion mapping for mobile robot obstacle avoidance, IEEE Transaction on Robotics and Automation, vol. 7, nr. 4, (1991) 535–539.
[11] Pal, A., Tiwari, R., & Shukla, A.: A focused wave front algorithm for mobile robot pathplanning. In Hybrid Artificial Intelligent Systems, *Springer Berlin Heidelberg*. (2011) 190-197
[12] Ghai, Bhavya & Shukla, A. (2016). Wave Front Method Based Path Planning Algorithm for Mobile Robots. 10.1007/978-3-319-30927-9_28, 2016
[13] FIPA Contract Net Interaction Protocol Specification. URL: http://www.fipa.org/specs/fipa00029/SC00029H.