IJWA

Cooperation of Autonomous Mobile Robots Using an Intelligent Integrated Approach

Khalil Shihab
School of Engineering, Computing & Science
Swinburne University of Technology – Sarawak Campus
Malaysia
kshihab@swinburne.edu.my

ABSTRACT: *In this work, we developed a hybrid-learning based model for intelligent control of autonomous mobile robots. In the proposed model, we used the Q-learning technique for navigation and the inter-process communication (IPC) for message passing between the robots. First, we developed a simulation program based on multi-agent paradigm that adopts these techniques for controlling the movements of the mobile robot and for collision avoidance. Second, we conducted experiments on three robots from the Robix Company and results show a good performance of the proposed method. At the motion control level, the Q-Learning technique is trained to perform motor control that moves one of the robots from the resource point to the destination point and back. The communication between the robots is carried out by using the inter-process communication technique (IPC).*

Keywords: Robotics, Q-Learning, Interprocess Communication, Intelligent Systems.

Received: 11 June 2009, Revised 18 July 2009, Accepted 30 July 2009

1. Introduction

Cooperative robot systems are emerging as a new frontier in robotics research, posing new challenges and offering new solutions to old problems [1]. They are usually employed to solve tasks that are impossible, or very difficult, for just one robot; a single robot may fail to perform a task in a desired manner and time. Another benefit of using this approach is that there can be a major reduction in the number of robots deployed to accomplish the assigned tasks that can be distributed efficiently. That also leads to a reduction in floor space requirements. This approach is, for example, being used in some DaimlerChrysler's Mercedes production operations in Germany, where it reduces production space requirements by up to 20% and manufacturing system investment by 5%. In the DaimlerChrysler deployment, as many as 15 robots are used in what are called "RoboTeams" for machining or assembly operations [2]. Each of the robots has its own controller and collaborates with other robots by applying shared information, transmitted via a communication network. One of the robots is the leader and the others follow; this helps facilitate the programming effort.

It is, however, apparent that this new generations of robots is difficult to control in order to carry out tasks that require timely, reliable cooperation [3]. Robots and agents, in agent-based systems, acting in dynamic domains still seem to exhibit strange behavior, such as walking into walls and using items inefficiently. Even though, to a certain degree, agents currently seem to act in an intelligent way and make intelligent decisions, there is still something lacking in their behavior. Their actions although intelligent still seem quite primitive or require the processing of large data sets in a limited time. Therefore, we used message passing and Q-learning techniques to address these drawbacks.

The efficiency of the cooperative robots system depends upon two main factors: the communication among the robots and the movement of the robots [3 and 4]. In this work, we used inter-process communication sockets to allow the robots to synchronize their actions by sending and receiving messages via the communication network. For the movement of our robots, we used the Q-Learning technique to train the robots in a series of training sessions. Many researchers used neural networks and other AI techniques for some of the functionalities of the robots. However, only a few of them used these techniques to adapt the robot controller [5].

Although the final aim is to apply our integrated approach to real robots, we decided to design and build a simulation program prior to investigations with real robots. The simulation program allowed us the flexibility to build a controlled robotic environment and to verify this approach before the real application [6].

2. Related Works

2.1 Reinforcement Learning (RL)

The Reinforcement Learning (RL) is an area of machine learning that is well-suited for solving robot navigation problems [7]. Its underlying strategy is to learn a mapping from situations to actions by trial-and-error interactions with a dynamic environment that is described by a set of states, see Figure 1. For a comprehensive coverage, see the book by Sutton and Barto [8] or the survey by Kaelbling, L. et al [9].

The simplest and the most widely used Reinforcement Learning algorithm is known as Q-Learning technique, see section 3 for detailed description.

Smart and Kaelbling [7] used Q-Learning to learn control policies for two simple robot tasks, corridor following and obstacle avoidance.

Qiao et al [10] Presented an automation learning and navigation strategy based on dynamical structure neural network and reinforcement learning. Results show that the robot can learn the correct action and finish the navigation task without people's guidance.

The work of Su Mu Chun et al [11] was based on using reinforcement learning and a neural network, but with Gaussian operators to robot navigation. However, it is not clear how these two techniques are integrated to allow the robot avoiding obstacles in a calculated time.

Saad et al [12] developed a small and simple team of mobile robots with limited capabilities in terms of memory and computational power. A set of robots (targets) are freely wandering in predefined area. Another set of robots (chasers) tries to track all target robots by following them. Both sets exhibit collection of behaviors (avoidance, follow).

Yoshikawa [13] proposed a Q-learning algorithm based on Genetic Algorithm and has a hierarchical evolutionary mechanism. The proposed learning algorithm introduces new adaptive action value tables and it enables sharing knowledge among agents effectively. However, this work suffers from its complexity to be implemented on small mobile robots.

Gu and Hu [14] used Q-learning to state-based planning of behavior-based walking robots, see also [15]. After the accumulation of some prior knowledge, the learning is switched to the autonomous learning stage to let the robot explore the solution space. They conducted their experiments in the RoboCup domain.

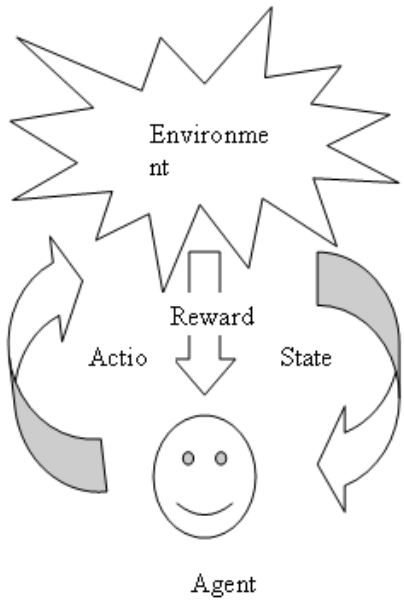

Figure 1. Reinforcement strategy

The most related work to the theme of this paper includes that of Asadpour [16]. A compact Q-learning algorithm with limited memory and processing power needs was developed. This compact Q-learning algorithm was applied to the task of robot safe wandering. However, the work of this paper addresses the problems of excessive training effects, environment of training and the false training due to moment of inertia of the robot body.

2.2 Interprocess Communication (IPC)

The first IPC system developed for robotics is IPT [17]. It was immediately followed by CMU-IPC system [18], which is based on message passing techniques. It was designed to facilitate communication between heterogeneous control processes in a large networked system and to provide sufficient functionality and flexibility in a real-time robotics system. For a thorough survey of IPC in the context of distributed robotics, see Gauthier el al [19].

Real-Time Communications (RTC) was designed as IPC middleware to provide real-time capabilities specifically for robotic applications. RTC is also based on message passing techniques.

Ko Nak et al [20] developed a simulator for multiple robot system using IPC. The simulator consists of the four units: communication, robot control, kinematic design, and driver for real robot operation and used for verification of collision avoidance algorithms in an environment with multiple heterogeneous robots.

Jianqiang Jia , Weidong Chen and Yugeng Xi [21] applied IPC to their robot design to carry out several tasks such as navigation and robot soccer.

3. Simulation

For our integrated approach to work and mimic realistic human behavior, we developed an experimental model. We wanted to find out how real life robots reacted in certain situations and what processes were required to drive and control their movements.

3.1 Experimental game

In this experiment we used three agents, namely A1, A2, and A3 that simulate the human reasoning process. To do this, agents maintain a list of cases (shared information) establishing points of view of other agents and use these cases to take future actions. These three agents work in a cooperative environment in which the agent A1 should wait for a message from the agent A2 in order to start its work. The agent A3 should also wait for a message from A1. However, A2 does not need to wait for either of these agents to ensure that the system works always in a safe state, i.e. it does not enter into a deadlock state. It generates messages at predetermined time intervals. We used inter-process communication (IPC) socket that establishes bidirectional communication between a server program and one or more client programs [22]; see Figure 2 and section 3 for more details.

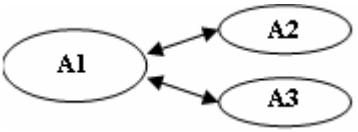

Figure 2. Multithreaded Client/Server Architecture

The game environment for the agent A3 is a simple evacuation of an agent from any room in the building, see Figure 3. At the start of the game, the agent is allocated to Room C (initial state) and we want the agent to learn to go outside the house (F). At each landmark (initial state, obstacle and destination state), A3 should send a message to A1 and updates the shared information.

We consider each room (including outside the building) as a state. The agent's movement from one room to another room is called an action. Figure 4 shows that states are represented by nodes in the state diagram while actions are represented by the arrows.

From state C, the agent can go to state D because state C is connected to D but with reward zero because D is not the goal state. From state C, however, the agent cannot directly go to state B because there is no direct door connecting room B and C (thus, no arrow). From state D, the agent can go either to state B or state E or back to state C. If the agent is in state E, then

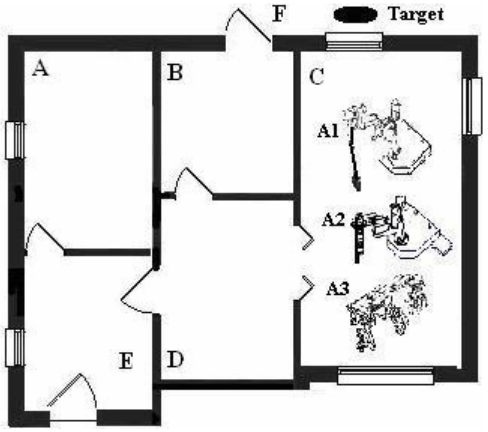

Figure 3. A simple agent's environment

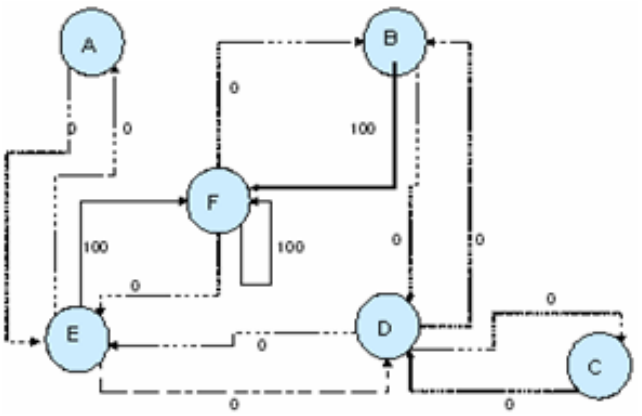

Figure 4. The state diagram

three possible actions are to go to state A with reward zero, or state F with reward 100 (because F is the goal state) or state D. If the agent is in state B, it can go either to state F or state D. From state A, it can only go back to state E.

The agents described in this experiment are able to participate in a multi-stage game in which one agent (A3) should learn through experience without a teacher by applying the Q-learning technique, which is a reinforcement learning technique that bridges the gap between supervised and unsupervised learning categories. At the start, the agent can pass from one room to another but has no knowledge of the environment. After a sequence of training sessions, the agent should be aware of the environment and the location of the target and other significant points. A map like for all of these pieces of information should be produced and saved in a sharable location to allow both robots easy access, see Figure 3 and Table 1.

Q-learning requires a similar matrix name Q in the brain of our agent that will represent the memory of what the agent has learned through many experiences. The rows of matrix Q represent the current state of the agent, the columns of matrix Q point to the action to go to the next state. As mentioned above, the agent starts without knowledge of the environment, thus we put Q as a zero matrix.

The transition rule of this Q learning is the following simple formula

$$Q(s_i, a) = R(s, a) + \alpha.Max[Q(s_j, \text{all actions})] \quad (1)$$

Where,

Q is the transition matrix; rows represent states and columns represent actions; s_i is the current state; s_j is the next state; **R** is the instant reward matrix; and **α** is the learning parameter.

Action to go to state						
Agent in state	**A**	**B**	**C**	**D**	**E**	**F**
A	-	-	-	-	0	-
B	-	-	-	0	-	100
C	-	-	-	0	-	-
D	-	0	0	-	0	-
E	0	-	-	0	-	100
F	-	0	-	-	0	100

Table 1. State reward values (R)

	B	**C**	**D**	**E**	**F**
A	20/E	40/E	20/SE	20/S	40/NE
B		20/E	30/S	50/SW	20/N
C			20/SW	60/SW	50/NE
D				20/SW	70/N
E					50/NE

Table 2. Distances between landmarks, where N=North, S=South, E=East, W=West, NE=North East, SW=South West, SE=South East

3.2 Q-Learning Technique

Given: State diagram with a goal state (represented by matrix R, see Table 1)
Find: Minimum path from any initial state to the goal state (represented by matrix Q)

Q Learning Algorithm goes as follow:

1. Set parameter **α**, and environment reward matrix **R.**
2. Initialise matrix **Q** to zero
3. For each episode:
   - Select random initial state
   - Do while the goal state not reached
     - *Select one among all possible actions for the current state*
     - *Using this possible action, consider to go to the next state*
     - *Get maximum Q value of this next state based on all possible actions using formula (1)*
     - *Set the next state as the current state*
     - *End Do*

The above algorithm is used by our agent (A3) to learn from experience or training. It was proposed for Markovian's problems decision, with discrete states and actions spaces. The Q-Learning is, therefore, well suited for on-line applications that are characterized by discrete states, and generally performs well in practice. Each episode is equivalent to one training session. In each training session, the agent explores the environment (represented by Matrix R), gets a reward (or nothing) until it reaches the goal state. The purpose of the training is to enhance the 'brain' of our agent that is represented by the Q matrix. The Q-learning algorithm is guaranteed to converge to Q*(s, a), the optimal action value function, with probability 1 as long as each state-action pair is continually updated [23]. The Q-learning algorithm learns an action value function, which is the expected sum of discounted future rewards for taking action a in state s and behaving optimally thereafter. This action value function is often represented in a Q-table of the form Q(state, action), which is updated during the learning process, see Table 1. More training will produce a better Q matrix that can be used by the agent to move in an optimal way. In this case, if the Q matrix has been enhanced, instead of exploring around and going back and forth to the same room, the agent will find the fastest route to the goal state.

Parameter α is in the range 0 to 1($0< \alpha<1$). If α is closer to zero, the agent will tend to consider only an immediate reward. If α is closer to one, the agent will consider a future reward with greater weight, and be willing to delay the receipt of a reward.

To use the Q matrix, the agent traces the sequence of states, from the initial state to the goal state, hence producing a trajectory of its path. At each step of the sequence, the agent should find an action that maximizes Q(state, action) for the current state. The mobile agent (robot) was trained in the above mentioned environment, see Figure 4.

Since the search space is low-dimensional and discrete for the task at hand, the convergence of the Q-learning to the optimal state is ensured [24].

4. Using Inter-Process Communication Sockets

Sockets are the most popular form of the Inter Process Communication (IPC) protocols. Network applications use sockets to communicate over a TCP/IP networks. A socket is one end-point of a two-way communication link between two programs running on the network. Because a socket is bidirectional, data can be sent as well as received through it.

Our robots send and receive messages from each other and these messages should match with the time calculated by each robot. This time is based on the distance and the time taken to finish the job. Each robot will send its position at constant intervals of time and at landmark points; these are the point of origin (the initial state), the vertices, and the target. Also, messages should be sent critical events. These events are:

- task start (task_id, type)
- task completion (task_id, type)
- task failure (task_id, error message)

If for some reason a message is not received by a waiting robot in a calculated time, this robot should send a warning message, and if again no response is received, the waiting robot will send a task failure message and will issue a call for an immediate halt to the system.

Once the mobile robot is trained, using the Q-Learning technique and the time intervals are recorded at each landmark point,

the task environment becomes known. Each robot is capable of detecting the current position of the other robot and the position of the point of origin (the initial state), and the target, thus allowing the robots to maintain collision avoidance while they are moving, see for example [2, 23].

5. Experimentation with Robix Rascal Robots

The Rascal RCS-6 robot from Robix [25] is a complete hardware and software robotics product, suitable for both the beginner and the professional. It comes with all the parts needed to build robots with grippers, sensors, and up to six joints. Robix recently adds Usbor hostware that supports easy distributed control of multiple robots and it has WiFi communication ability. Rascal's scripts and macros can be combined with any high level programming code that can link to C-style libraries included with the RCS-6.

At present Robix products do not have a mobile platform for the Rascal set. However, it allows the construction of a three-legged robot. Therefore, we assembled a modified version of the three-legged robot to carry out the experiment. It has two feet on each leg and does a shuffling walk forward and backward, turning as it goes if desired. The experiment was carried out by two groups with two students in each group and they received prestigious awards.

5.1 The Mobile Robot Environment

Once the mobile robot is operational, it should work on a flat, open navigation space of 2m x 2m search area that consists of 6 white (high intensity) spots (obstacles) and a black (low intensity) colored target.

The mobile robot is assisted by a sensor and a distance map (Table 2). As stated before, the movement of the mobile robot is known to other robots through the shared information that produced by the IPC mechanism. Also, the estimated time required for the mobile robot to move from one landmark (a spot or a target) to another is calculated and updated to reveal the current position and the estimated time required to reach to the next landmark, i.e. the mobile robot is required to send a message at each landmark. If for some reason that the mobile robot does not reach to the next landmark or a message is not received according to the anticipated time, a halt command (task failure) is issued to stop the process.

5.2 Light Sensor

The mobile robot was assembled to carry a light sensor that was adjusted to read data only at a distance of less than 10cm. The sensor was also sensitive enough to differentiate between the black (low intensity light) object, which is the target object and the obstacles that were colored white (high intensity light).

The Usbor software, provided by Robix, consists of two main programs, the Nexus and the Nexway which communicate via TCP/IP and allow remote operations. Therefore, our program gains control of the robots by loading the Nexway and using it to relay commands and receive sensor information.

5.3 Customizing the Rascal Script

The Rascal scripting language is designed for programming motion sequences and setting motion parameters. Although the script can be used to write many pieces of program code to carry out many functions, it lacks general programming constructs including variables, conditional statements, loop constructs, function calls, system calls, etc. Therefore, many functions and programming codes are added to the script in order to accomplish the experiment. Some of these functions are as follows:

Synchronize function: It is used to synchronize time and organize the movement of the robots. If one of the stationary robots is busy with a task, the other robot should wait until the task is finished. The pseudocode of this function is as follows:

```
Timer() = current time or clock time
Time_needed= the time needed to carry out the task at hand

endTime = Time() + Time_needed
do while endTime>Time()
loop
```

Winsock (Window Socket) Function: It defines a network programming interface for Microsoft Windows which is based on the "socket" paradigm popularized in BSD Unix. It encompasses both the familiar Berkeley socket style routines and

a set of Windows-specific extensions. It is used here to allow smooth communications between programs that control the robots using the server/client paradigm.

Start Function: It reads a user option. It first checks the status of the robots and issues a function call to the synchronize function if the required robot is not either in the busy or the wait state. After assigning the task to a robot, it should send a message to all. The pseudocode is as follows:

```
Check if a robot x = true and connection = true;
Then if user option = true;
  Then if user selection = true;
 Then call the synchronize
      function;
    Call WinSocket();
    Send a message to the robots;
       Else send error-message "Error user selection";
    End;
  Else send error-message "Error
    user option";
   End;
   Else send error-massage "No connection";
End;
```

In order to activate a robot to carry out any action, the movements of the robot should be carefully controlled to ensure the accuracy of its movements that lead it to the precise position. In this regard, the Rascal programming script is very helpful and using it with Java, C++ or Visual Basic, we can produce programs for controlling the robots with high accuracy.

5.4 Cooperative Robots in Action

One robot has a small plastic spoon attached to its end. It dips into tilted containers of cement, sand, gravel and water. It carries spoons of each material and pours them in the container. The mix ratio is either entered by the user or the robot uses its own ratio of 1 spoon cement: 2 spoons sand: 3 spoons gravel: 9 spoons water. Armed with a strong pen shaped metal, the second robot mixes the materials together for a predetermined time, see Figure 5.

Once the mix is ready, the mobile robot is supposed to use the wrist and gripper assembly to pick up and carry the container to the destination (target). However, this task is difficult for our robot because of the current limitations of

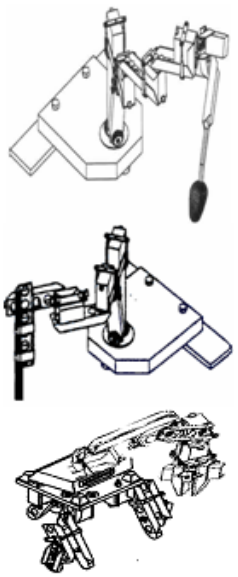

Figure 5. The assembled robots A1, A2, and A3

Robix Rascal set. Therefore, the robot is made to pick up a small black object, a relatively small. The robot then uses the light sensor and the distance map for navigation of the search space, and drops the object at the destination next to the empty object that should be picked up and moved to the initial state. This process is repeated until the robots receive a stop command.

The experiment was repeated many times in order to have all the robots carrying out the task with high accuracy. For each repetition we had to adjust the time, the positions, the distances, and the locations of the materials used in the experiment. It required extra effort to adjust the movements of the mobile robot, especially backward from the target to the initial state.

6. Conclusions

In this paper, we have attempted to address the possibility of designing and building a program that enables a number of robots working cooperatively to carry out an assigned task. It begins by developing a simulation model for multiple agents that learn to accomplish a task by applying the Q-Learning and IPC algorithms. The work also shows that the model could be used to control the actions and movements of multiple robots by applying these techniques.

We conclude that using the integrated technique is helpful because it provides more accurate control on the movements of multiple robots that working in a cooperative environment. Furthermore, we determined that the best method for robot navigation is to use a map of the environment that guides the robot. With this map and good communication between the robots, the mobile robot could very accurately carry out its tasks.

As an extension to this work, the application of the Q-Learning algorithm could be compared with both RRT (Rapidly-exploring Random Tree) and nonholonomic algorithms.

References

[1] Dubey, V. N., Dai, J. S. (2006). A packaging robot for complex cartons, Industrial Robot, V. 33 (2) p. 82-87.

[2] KUKA Roboter GmbH. (2008). Production robots are becoming versatile team players, Augsburg/Stuttgart, Germany, Category: Industry, Real Estate & Construction.

[3] Hidenori Kawamura, et al. (2009). Cooperative control of multiple neural networks for an indoor blimp robot, *Artif Life Robotics*, 13, p. 504-507.

[4] Shihab, K., Chalabi, N. (2009). Bringing Emotion to Computer Games, *Opportunities and Challenges for Next-Generation Applied Intelligence*, Been-Chian Chien and Tzung-Pei Hong (eds.), Springer, p. 193-198.

[5] Lehman A. et al. (2008). Surgery with cooperative robots, Computer Aided Surgery, V.13, Issue 2, p. 95-105.

[6] Leesa, M. B. L., Theodoropoulos, G. K. (2006). Agents, games and HLA, *Simulation Modelling Practice and Theory*, V. 14, Issue 6, p.752-767.

[7] Smart, W. D., Kaelbling, L. P. (2002). Effective Reinforcement Learning for Mobile Robots, *In: Proc. of the IEEE International Conference on Robotics and Automation*, p. 3404- 3410.

[8] Sutton Richard, S., Andrew G. Barto. (1998). Reinforcement Learning: An Introduction, MIT Press, Cambridge, MA.

[9] Kaelbling, Leslie P. et al. (1996). Reinforcement Learning: A Survey, *Journal of Artificial Intelligence Research*, 4: p. 237-285.

[10] Qiao J. et al. (2009). Q-Learning Based on Dynamical Structure Neural Network for Robot Navigation in Unknown Environment, Lecture Notes in Computer Science, Springer Berlin / Heidelberg, p. 1611-3349.

[11] Su M. et al. (2004). Reinforcement-Learning Approach to Robot Navigation, *IEEE International Conference on Networking, Sensing, and Control*, p. 665-669, Taiwan.

[12] Saad E. M. et al. (2008). Multi-Target Tracking using Human Teachable Robots, *International Journal of Computers,* Issue 4, V. 2, p. 479-488.

[13] Yoshikawa Masaya., Takeshi Kihira., Hidekazu Terai. (2008). *Q-learning based on hierarchical evolutionary mechanism, WSEAS Transactions on Systems and Control.* V. 3, Issue 3, p. 219-228.

[14] Gu, D., Hu, H. (2005). Teaching robots to plan through Q-learning, Robotica, V. 23 , Issue 2, p. 139- 147.

[15] Thomaz., Andrea, L., Cynthia Breazeal. (2008). Teachable robots: Understanding human teaching behavior to build more effective robot learners, *Artificial Intelligence*, V. 172, p. 716-737.

[16] Asadpour, M., Roland Siegwart. (2004). Compact Q-learning optimized for micro-robots with processing and memory constraints, *Robotics and Autonomous Systems* , 48, p. 49-61.
[17] Gowdy. J. (1996). An object oriented toolkit for interprocess communication, Technical Report, CMU-RI-TR-96-07, Robotics Institute, Carnegie Mellon University, Pittsburgh, PA.
[18] Pedersen, J. (1998). Robust communications for high bandwidth real-time systems, Technical Report, CMU-RI-TR-98-13, Carnegie Mellon University.
[19] Gauthier, D. et al. (1987). Interprocess communication for distributed robotics, Robotics and Automation, IEEE Journal of Robotics and Automation, V. 3, Issue 6, p. 493-504.
[20] Ko Nak Y. et al. (2007). 3-Dimensional simulator for multiple robot system using TCP-IP. *In:* Proceedings of the 25th conference on IASTED International Multi-Conference: *Software Engineering*, p. 130-135.
[21] Jianqiang Jia., Weidong Chen., Yugeng Xi (2004). Design and Implementation of an Open Autonomous Mobile Robot System. *In: Proceedings IEEE International Conference on Robotics & Automation,* p. 1726-1731*,*
[22] Kim K. H. (2006). A Non-Blocking Buffer Mechanism for Real-Time Event Message Communication, Real-Time Systems, V. 32, p. 197–211.
[23] Simeon T. et al. (2002). Path Coordination for Multiple Mobile Robots: A Resolution-Complete Algorithm, *IEEE Transactions on Robotics and Automation,* V. 18 (1) p. 42-49.
[24] Watkins, C. J. C. H. (1989). Learning from Delayed Rewards, PhD thesis, Cambridge University, Cambridge, England.
[25] Robix™ Robotics. (2008). http://www.robix.com/default.html, last accessed April.