Annals of the University of Craiova, Math. Comp. Sci. Ser.
Volume 35, 2008, Pages 139–148
ISSN: 1223-6934

A Mobile Agent Virtual Reality Modeling of Searching Robots

Claudiu Popirlan and Mihai Dupac

Abstract. This paper presents a mobile agents approach for multiple searching robots in a virtual reality environment. The robots are programmed to perform objects searching in a real house starting from a base point. A robot is defined as an autonomous ground vehicle and is equipped with software (mobile agent) and sensors. When a robot found the required object, the mobile agent announce the end of the searching, and the robots return to the base. The equations of motion (kinematics and dynamics) were presented, and the motion planning with obstacle avoidance (collision free) implemented. The efficacy of algorithms are evaluated based on mobile agents communication.

2000 Mathematics Subject Classification. 68T99; 65D17.
Key words and phrases. Web-based Modeling, Simulation, Animation, Virtual Reality, mobile agents, dynamic modeling, visualization, object oriented implementation.

1. Introduction

The concept of mobile agent is defined in [1], [2], [5]. They are autonomous objects that can migrate from node to node on behalf of the user who have executed them and make use of the databases or computation resources from clients connected by the network. In order for a mobile agent to be able to migrate, there must be a virtual place, the so-called mobile agent system, that supports mobility. To facilitate effective communication ([3], [4]), the agents has to include information about the possible states of knowledge, abilities and preferences of the other agent(s) present in the environment.

Research on robots has attracted attention in the last years since [14]. Most of the research have been directed to the use of kinematic models of the mobile robots to achieve and accomplished the motion control [9, 8, 11]. Later on, the research has been focused on robots with additional sensory system to develop autonomous guidance path planning systems [12]. Sophisticated sensory systems has been used in [13], helping the software to learn about the operating environment and to evaluate path constraints for a good path planning programming.

Path planning consists of determining a route joining two input configurations, i.e., from one coordinate location to another location. Path planning algorithms are determinant for the mobile agent behavior, including collision avoidance. Modelling and simulation of a group of mobile robots, the controllability issue and path planning was studied in [15]. Different other approaches including rigid formations, potential field methods, and neural have been studied in [14]. The motion planning for mobile robots when using a dynamic environment and moving obstacles was studied in [16].

The computational geometry algorithms for path planning use a polygonal description of the scene and operate on the vertices visibility graph of the obstacles (for more details see [17]). For the work presented here this type of planning has been adopted.

Received: 11 December 2008.

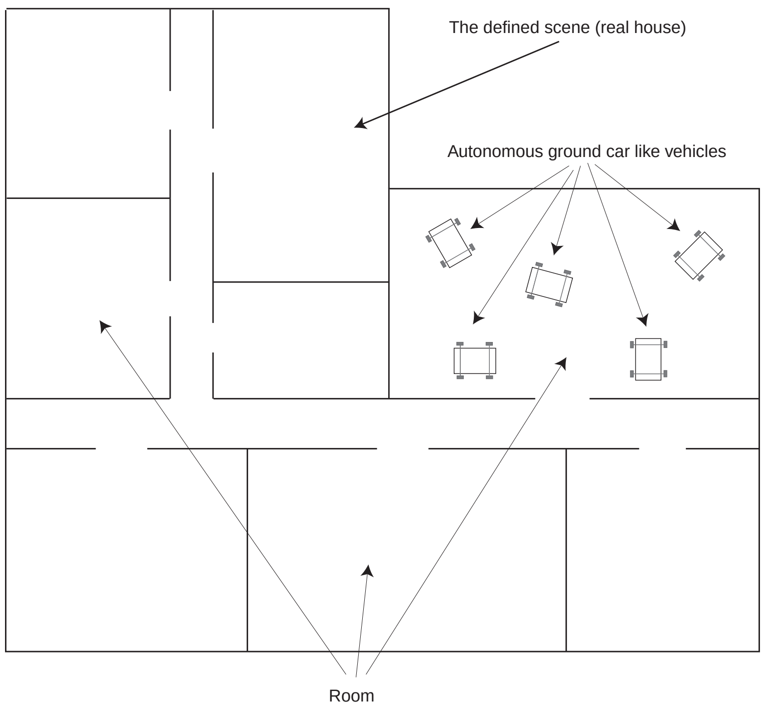
The defined scene (real house)
Autonomous ground car like vehicles
Room

Figure 1. The Virtual Reality Defined Scene (Real House) with Five Searching Robots

In this paper searching robots defined as an autonomous ground car like vehicles and equipped with software (mobile agent) and sensors, in a virtual reality environment is presented (see Fig 1). The robots are programmed to perform objects searching in a defined scene (real house) starting from a base point, and to return to the base when the required object is found. The return to the base signal is established by the mobile agent announcement. The equations of motion (kinematics and dynamics) are presented, and the motion planning with obstacle avoidance (collision free) described. The efficacy of algorithms are evaluated based on the vehicle design, sensors performance, and the artificial intelligence of the associated mobile agents.

2. Mathematical Model

2.1. Kinematics. To study the robot kinematics one can consider x and y the x-coordinate and y-coordinate of the robot center of the mass (relative to the origin) in a Cartesian reference frame xOy, and $\alpha(t)$ the robot orientation, i.e., the angle between the robot direction and the Ox axis. The robot velocity in the x direction (longitudinal speed) is given by its first derivative $u = \dot{x}$, the robot velocity in the y direction (lateral speed) is given by $v = \dot{y}$, and the angular speed ω is given by $\dot{\alpha}$.

The longitudinal speed u, the lateral speed v and the angular speeds $\dot{\alpha}$ can be determined using the angular speeds on the left and right drive wheels of the robot (it is considered that the left angular speed and the right angular speed of the wheels are the same). Let consider ω_{left} the angular speed of the left drive wheel and ω_{right}

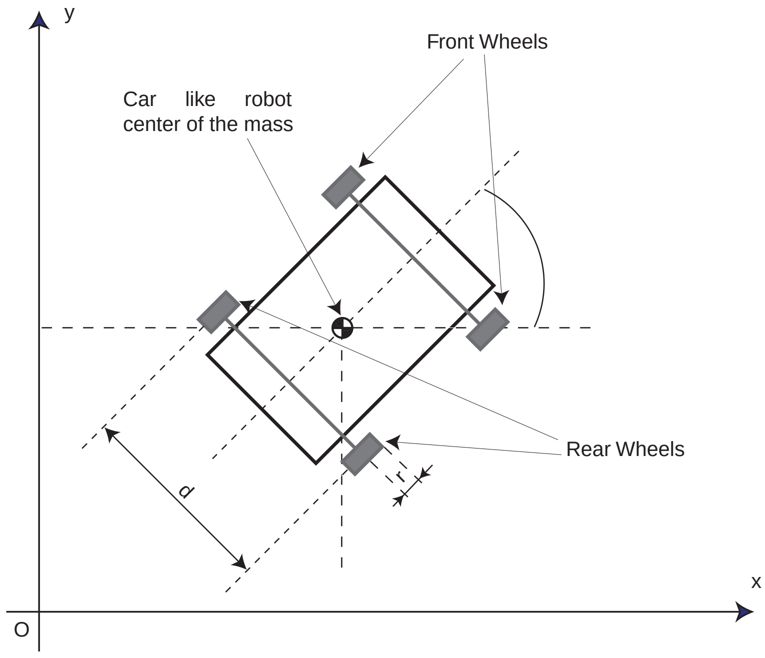

FIGURE 2. The Autonomous Ground Car like Searching Robot

the angular speed of the right drive wheel. One can write,

$$\begin{aligned} w &= \frac{r(\omega_{left} + \omega_{right})}{2} \\ \dot{\alpha} &= \frac{r(\omega_{left} - \omega_{right})}{d} \end{aligned} \tag{1}$$

where r is the wheel radius and d is the distance between the wells as shown in Fig 2. The longitudinal speed $u = \dot{x}$ and the lateral speed $v = \dot{y}$, can be calculated using

$$\begin{aligned} \dot{x} &= w\cos(\alpha) \\ \dot{y} &= w\sin(\alpha) \end{aligned} \tag{2}$$

From Eqs. (1) and (2) one can write

$$\begin{aligned} \dot{x} &= \frac{r\cos(\alpha)(\omega_{left} + \omega_{right})}{2} \\ \dot{y} &= \frac{r\sin(\alpha)(\omega_{left} + \omega_{right})}{2} \\ \dot{\alpha} &= \frac{r(\omega_{left} - \omega_{right})}{d} \end{aligned}$$

or equivalent

$$\begin{aligned} u &= \frac{r\cos(\alpha)(\omega_{left} + \omega_{right})}{2} \\ v &= \frac{r\sin(\alpha)(\omega_{left} + \omega_{right})}{2} \\ \omega &= \frac{r(\omega_{left} - \omega_{right})}{d} \end{aligned}$$

If the motion is nonholonomic, the non slip condition can be written as,

$$\dot{x}\sin(\alpha) \quad = \quad \dot{y}\cos(\alpha)$$

2.2. Dynamics. One can write the nonholonomic equations of motion for the mobile robot based on the Euler Lagrange formulation. In a matrix form one can write

$$\begin{bmatrix} m & 0 & 0 \\ 0 & m & 0 \\ 0 & 0 & I \end{bmatrix} \begin{bmatrix} \ddot{x} \\ \ddot{y} \\ \ddot{\alpha} \end{bmatrix} = \frac{1}{r} \begin{bmatrix} \cos(\alpha) & \cos(\alpha) \\ \sin(\alpha) & \sin(\alpha) \\ -d & d \end{bmatrix} \begin{bmatrix} \tau_l \\ \tau_r \end{bmatrix} + \begin{bmatrix} \sin(\alpha) \\ \cos(\alpha) \\ 0 \end{bmatrix} \lambda \tag{3}$$

where τ_l is the torque of the left wheel, τ_r is the torque of the right wheel, m is the mass of the motor, I is the robot mass and inertia, and λ the Lagrange multipliers of constrained forces. The friction force acting on a wheel is F. Two DC motors are assumed to propel the robot. Each DC motor is connected to the drive wheel on each side and generate torque. For the sake of briefness, the discussion of the DC motor model is thus ommitted here, eventhough the model is included in the simulation. Considering are τ_{linear} and $\tau_{angular}$ the linear and angular torques, one can write,

$$\begin{aligned} \tau_{linear} &= \frac{\tau_l + \tau_r}{r} \\ \tau_{angular} &= \frac{d(\tau_l + \tau_r)}{r} \end{aligned}$$

From the previous two equations one can deduce

$$\begin{aligned} \ddot{x} &= \frac{\tau_{linear}}{m}\cos(\alpha) + \frac{\lambda}{m}\sin(\alpha) \\ \ddot{y} &= \frac{\tau_{linear}}{m}\sin(\alpha) - \frac{\lambda}{m}\cos(\alpha) \\ \ddot{\alpha} &= \frac{\tau_{angular}}{I} \end{aligned} \tag{4}$$

Applying the second derivative to the Eq. (2) one can obtain

$$\begin{aligned} \ddot{x} &= -w\dot{\alpha}\sin(\alpha) + \dot{w}\cos(\alpha) \\ \ddot{y} &= -w\dot{\alpha}\cos(\alpha) + w\sin(\alpha) \\ \ddot{\alpha} &= \dot{\omega} \end{aligned} \tag{5}$$

Finally, from Eqs. (4) and (5) one can deduce

$$\begin{aligned} \frac{\tau_{linear}}{m}\cos(\alpha) + \frac{\lambda}{m}\sin(\alpha) &= -w\dot{\alpha}\sin(\alpha) + \dot{w}\cos(\alpha) \\ \frac{\tau_{linear}}{m}\sin(\alpha) - \frac{\lambda}{m}\cos(\alpha) &= -w\dot{\alpha}\cos(\alpha) + w\sin(\alpha) \\ \frac{\tau_{angular}}{I} &= \dot{\omega} \end{aligned}$$

2.3. Obstacle Avoidance and Sensors. Obstacle avoidance was integrated on the mobile agent software. The software adjust the direction of the robot based on any obstacles in its path. While the robot executes the searching, if any obstacles is found on the robot path, the direction is adjusted to run tangent to the obstacle, and so, the old path is replaced with a new path to avoid the obstacle. Since the mobile agent continuously update the path based on the sensors information, the obstacle avoidance analysis and the direction path is constantly replanning. This continuous

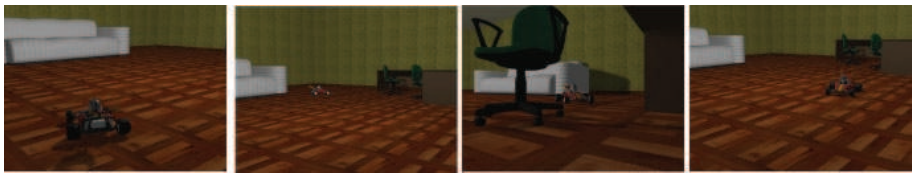

Figure 3. An Example of a Searching Robot in a Room: Starting from the Base Point, Searching with Obstacles (Chair Legs) Avoidance, and Return to Base

replanning allows the searching robot to handle a highly dynamic environment and to take advantage of any path free opportunity (the visibility graph).

The sensors installed on the robot such as stereo vision sensors (two CCD cameras fixed on the left and right side of the searching robot) and laser scanning system allows the robot to navigate in the defined environment without hitting the walls, and with obstacle avoidance (collision free). The collision free is implemented for the static motion and intended for the dynamic motion. The laser scanning system allows the agent to plan the path and trajectory based on the 3D created map of the environment. The two CCD cameras synchronously capture the image and combine it with the 3D created map for the searching object recognition purpose. For the present described system (car like robot) the obstacles are assumed to be convex polygons.

3. Mobile Agent System Description

The system must be able to simulate objects finding using the knowledges in virtual reality. In this case, the mobile agent visit, one after another, all or a part of the virtual reality (using sensor) to whom they ask for certain information. An example of five searching robots with mobile agents and sensors as shown in Fig 1. When the agent gathers all the knowledge specified by its user it return to the base. The system firstly obtains the knowledges that are currently in the virtual reality(is send by the user) and gives that knowledges to the agent. The mobile agent migrates through the virtual reality having a list of knowledges, set by the user on behalf of whom it is working. Once the agent gets all the solutions to its problems it can return to the base.

Let's imagine a representation of the knowledge in the form of *K x*, where x is a positive number that uniquely identifies a desire (for agent). The mobile agent may have as its Wishes List (WL):

$$K\ 3\ ,\ K\ 18\ ,\ K\ 1\ ,\ K\ 10$$

A sensor might respond to the following knowledge, *K 18*, with::

$$K\ 20\ ,\ K\ 13\ ,\ K\ 7$$

We suppose that the sensor return the following new WL:

$$K\ 3\ ,\ K\ 1\ ,\ K\ 10\ ,\ K\ 20\ ,\ K\ 7\ ,\ K\ 13,$$

obtained by replacing knowledge *K 18* with new knowledge which will become new desires for agent. These new desires, along with those which does not yet have an answer will be the next place to visit. As it is natural, understanding of a certain knowledge does not imply an understanding of another adjacent, related knowledge. An example of such knowledge is the fundamental geometrical notion of "point". From here arises the need to define knowledge in primary and complex. Therefore, an agent may receive from the sensor, in response to its demands whether new extended knowledge that will be treated as new desires, or basic knowledge that will not require further answers. The mission of the mobile agent is considered to be finished when it has only primary knowledge, and its list of desires is empty.

3.1. The System Communication. The system must be able to make possible the agents communication, including the communication protocols. The communication protocols should allow message understanding and wireless transmission. Message understanding implies decision acceptance or rejection based on the collected sensors information and other mobile agents information. Mobile agents communication is based on sending (action) and receiving (perception) messages.

The mobile agents will communicate to each other to achieve the defined goals. The degree of coherence and coordination comes from the extent to which the system avoid redundant actions, competition on resources, bottlenecks and the unsafe operating conditions. The goal is to maintain an overall coherence, without having always a global control in place. The coordination between mobile agents not entering the competition is based on cooperation. For the agents entering in the competition, or those having reciprocal dependence, the coordination is based on negotiation.

In our system, the communication language has two levels:

- Level 1: communication between agents and sensors;
- Level 2: communication between agents.

Communication between agents and sensors (level 1) is in one sense: from sensor to agent. The agent receive informations from sensor and make decision.

Communication between agents (level 2) is done via messages. An agent can communicate with other agent by message passing. An agents that wants to communicate with another agent first has to create a message object, then send it to the target agent. A message object has a kind and an optional argument object. The receiver agent determine what to do by checking the kind of received message and get parameters as the argument object. For system implementation one can use a subset of a standard indicators ([18]) of the Knowledge Query and Manipulation Language (KQML):

tell
$: content < expression >$
$: language < word >$
$: ontology < word >$
$: in - reply - to < expression >$
$: force < word >$
$: sender < word >$
$: receiver < word >$

This indicators can be used when an mobile agent found an object and announce its finding. The message agents of our system act somewhat like the active packets in the capsule tradition of active networking research; they are minimal programs that carry a payload of communications data across the network. They make their decisions about where they need to go based on information that routing agents, with which they share the network, accumulate and cache.

3.2. An algorithm for the mobile agents planning. Mobile agents are capable of traveling collision-free from a start point (S) to a destination point (D) without the intervention of a human as shown in Fig 3. We cover low level path planning in static environments context. A static environment consists of stationary obstacles.

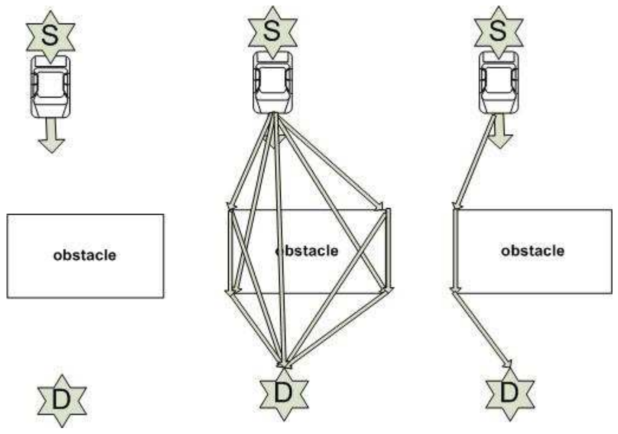

FIGURE 4. The visibility graph

Here determining a shortest, collision free path from S to D requires knowledge of all pairs of points visible from each other (without an intervening obstacle), so a visibility graph is created. Since the obstacles are in motion, the beginning path may no longer be collision free after a certain period of time. This problem is addressed by using an accessibility graph to determine whether the current path is still valid or must be changed. This paper concentrates on techniques we have implemented for finding the object in the static case.

Obstacles are assumed to be polygons. Our approach assumes that all the polygons are convex, and this must be verified at input. A polygon is simple if there is no pair of non consecutive edges sharing a point. A convex polygon is a simple polygon with a convex set, that is to say, any two points in the polygon can be connected without going outside the polygon([17]). This validation is achieved by testing each polygon's convexity. The convex hull algorithm implemented is Grahams scan. (Ironically, Graham is the name of the Computer Science Department building at A&T.) It works in three phases:

(1) Find an extreme point. This point will be the pivot and is guaranteed to be on the hull. It is chosen to be the point with largest y coordinate.
(2) Sort the points in order of increasing angle about the pivot. We end up with a star-shaped polygon (one in which one special point, in this case the pivot, can "see" the whole polygon).
(3) Build the hull, by marching around the star-shaped polygon, adding edges

The visibility graph (Fig. 4) is created by:

- Reading in S, D, and the obstacles.
- Generating all paths possible from S to D.
- Removing those paths obstructed by obstacles.
- Determining the shortest path

Once the Line array only contains those lines that are visible from one another, we can determine the shortest path from S to D. An adjacency structure contains Points that are used to represent the visibility graph. We implemented Dijkstras algorithm to find the shortest path. To create the visibility graph, all of the points need to be connected together. This is done with a brute force method that starts with S and connects it with each point; we continue, connecting each pair of points. Given the

Algorithm 1 Priority queue and inserts/deletes from the binary search tree

```
procedure insert()
$j := 1$;
while $j \leq N$ do
   $y1 := lines[j].p1.y$;
   $y2 := lines[j].p2.y$;
   pqinsert(j,y1);
   if $y1 \neq y2$ then
      pqinsert(j, y2);
      $j := j + 1$;
   end if
end while

procedure scan()
while $pqsize \neq 0$ do
   $j := pqremove()$;
   $x1 := lines[j.i].p1.x$;
   $y1 := lines[j.i].p1.y$;
   $x2 := lines[j.i].p2.x$;
   $y2 := lines[j.i].p2.y$;
   pqinsert(j, y1);
   if $j.c = y1$ then
      bstinsert(j.i, x1);
   end if
   if $j.c = y2$ then
      bstinsert(j.i, x2);
   end if
end while
```

{Remark: Since our binary search tree is based on line orientation, we use a counterclockwise function to determine if a line should be in the left subtree or right subtree of another line in the tree. While this function can determine which subtree of another line a given line belongs to, it also detects if the two lines intersect.}

number of points, N, connecting the graph is done in $O(N^2)$ time. Special care must be taken to ensure that no lines lie within polygons. Once we have an initial graph, we must remove any lines that intersect with a polygon. We accomplish this by scanning through the graph in a bottom-to-top method. In order to simulate scanning in lines, we use a priority queue, binary search tree and linked-list. The binary search tree is used to insert/delete the points of the lines in a bottom-to-top fashion, while the linked-list stores intersections found within the tree. By using a binary search tree, as opposed to nested for loops, our search algorithm runs in $O(Nlog_2N)$ versus $O(N^2)$ time. To ensure that the lowest point is used for insertion/deletion, a priority queue is used. Inserting all the ordinates of the lines generates the priority queue (which is ordered in nondecreasing order of the ordinates). By removing all the points from the priority queue and inserting/deleting from our binary search tree accordingly, all intersections are found. Insertion into the binary search tree is done when the lowest point of a line is inserted. Once the line's other endpoint is found, it is deleted from the tree (see Algorithm 1).

4. Conclusions and Future Work

In this paper a group of searching robots based on mobile agents for motion planning and obstacle avoidance in a virtual reality environment was presented. The robots are defined as an autonomous ground car like vehicles and programmed to perform objects searching in a defined environment. The robots are equipped with software (mobile agent) and sensors, and moves based on the described equations of motion (the kinematics and dynamics of the robots is included).

In the future, new mobile agents searching robots to maneuver in a dynamic environment will be proposed. This requires additional functions (methods) and abstractions (classes) that will work with obstacles in motion.

References

[1] P. Braun , W. Rossak, *Mobile Agents: Concepts, Mobility Models, & the Tracy Toolkit*, Elsevier Inc.(USA) and dpunkt.verlag(Germany), 2005

[2] S. Russell and P. Norvig *Artificial Intelligence: A Modern Approach*, Prentice Hall, 1995

[3] Claudiu Popîrlan, Cristina Popîrlan, Algorithms for Mobile Agents in Network using Tracy(Mobile Agent Toolkit), *Research Notes in Artificial Intelligence and Digital Communications*, Vol.106, Thessaloniki, Greece, p.31-38, (2006).

[4] C.I. Popîrlan, & C. Popîrlan, Mobile Agents communication for knowledge representation, *11-th World Multiconference on Systemics, Cybernetics and Informatics (WMSCI 2007)*, Orlando, USA, July 8-11, *1*, 92-96, (2007).

[5] J. Baumann, *Mobile Agents: Control Algorithms*, Lecture Notes in Computer Science, Springer, 2000.

[6] B. Eckel, *Thinking in Java*, Prentice Hall (4-th Edition), 2006.

[7] *The Sun Developer Network (SDN) Web page*, (`http://java.sun.com/`).

[8] G. Mester, Motion Control of Wheeled Mobile Robots, *4th Serbian-Hungarian Joint Symposium on Intelligent Systems, SISY*, 119-130, (2006).

[9] W. Dong, Y. Guo, Dynamic tracking control of uncertain nonholonomic mobile robots, *Intelligent Robots and Systems, (IROS 2005), IEEE/RSJ*, 2774 - 2779, (2005).

[10] P. Vleugels, Exact motion planning for tractor-trailer robots Svestka, *J. Robotics and Automation, Proceedings of IEEE International Conference*, Volume 3, 2445 - 2450, (1995).

[11] A. De Luca, G. Oriolo, C. Samson, *Robot Motion Planning and Control: Feedback control of a nonholonomic car-like robot*, Springer Verlag, 2006.

[12] X. Xiang, J. Zhang, C. He, An Efficient System for Nonholonomic Mobile Robot-Path Planning, *Advances in Intelligent Systems Research, International Conference on Intelligent Systems and Knowledge Engineering (ISKE 2007)*, (2007).

[13] T. R. Wan, H. Chen, R.A. Earnshaw, A Motion Constrained Dynamic Path Planning Algorithm for Multi-Agent Simulations, *The 13th International Conference in Central Europe on Computer Graphics, Visualization and Computer Vision'2005*, 211-218, (2005).

[14] R. Gayle, A. Sud, M. C. Lin, D. Manocha, Reactive Deformation Roadmaps: Motion Planning of Multiple Robots in Dynamic Environments, *Intelligent Robots and Systems, IROS 2007. IEEE/RSJ International Conference*, 3777-3783, (2007).

[15] G. Klancar, B. Zupancic, R. Karba, Modelling and simulation of a group of mobile robots, *Simulation Modelling Practice and Theory*, Volume 15, 647658, (2007).

[16] T.-J. Pan, R.C. Luo, Motion planning for mobile robots in a dynamic environment with moving obstacles, *Robotics and Automation, IEEE International Conference Proceedings*, Volume 1, 578 - 583, (1990).

[17] F. Preparata, M. Shamos, *Computational Geometry: An Introduction*, Springer-Verlag, New York, 1985.

[18] T. Finin, Y. Labrou, J. Mayfield, *Software Agents: KQML as an agent communication language, invited chapter in Jeff Bradshaw (Ed.)*, MIT Press, Cambridge, 1997.

(Claudiu Popirlan) University of Craiova Faculty of Mathematics and Computer Science, Department of Computer Science 13 Alexandru Ioan Cuza Street, Craiova, 200585, Romania
E-mail address: `popirlan@inf.ucv.ro`

(Mihai Dupac) University of Craiova Faculty of Mathematics and Computer Science, Department of Computer Science 13 Alexandru Ioan Cuza Street, Craiova, 200585, Romania
E-mail address: `mihai.dupac@inf.ucv.ro`