2010 8th IEEE International Conference on
Control and Automation
Xiamen, China, June 9-11, 2010

The Bubble Rebound Obstacle Avoidance Algorithm for Mobile Robots

I. Susnea, A. Filipescu, G. Vasiliu, *Member, IEEE*, G. Coman, A. Radaschin

Abstract—**This paper proposes a simple, reactive algorithm for real time obstacle avoidance, aimed to be implemented on low cost, low power microcontrollers. According to this algorithm, the robot reacts to obstacles detected within an area called "sensitivity bubble", whose shape and size are dynamically adjusted, depending on the kinematics of the robot. Upon detection of an obstacle, the robot "rebounds" in a direction having the lowest density of obstacles, and continues its motion in this direction until the goal becomes visible, or a new obstacle is encountered. Based on experimental results with simulators and real robots, the paper describes the performances and drawbacks of the algorithm, in comparison with other, simple obstacle avoidance algorithms.**

I. Introduction

THE ability to detect and avoid obstacles in real time is crucial for any implementation of the control system for autonomous vehicles. Therefore, a significant number of solutions have been proposed for this problem. Unfortunately, most of these solutions demand a heavy computational load, which makes them difficult, if not impossible, to implement on low cost, microcontroller based, control structures.

This paper presents the results of a research aimed to develop a new algorithm for obstacle avoidance relying on low cost ultrasonic, or infrared sensors, and involving a reasonable level of calculations, so that it can be easily used in real time control applications with microcontrollers.

Besides this introduction, the structure of the present paper is as follows:

- Section II contains a brief overview of the existing solutions. This section was introduced in order to facilitate the understanding of the proposed algorithm. However, only reactive algorithms, of comparable complexity are considered in this analysis.
- Section III contains a description of the proposed solution, and notes on the actual implementation.
- Section IV presents the experimental results used for evaluating the algorithm.
- Section V is reserved for conclusions and discussion.

This work was supported by CNCSIS –UEFISCSU, project number PNII – IDEI, ID_641/2007

Ioan Susnea is lecturer at the Department of Automation and Industrial Informatics, Faculty of Computer Science, University "Dunarea de Jos" of Galati, Domneasca, 47, Galati, 800008, Romania. e-mail: isusnea@yahoo.com, ioan.susnea@ugal.ro

Adrian Filipescu is professor at the Department of Autoamation and Industrial Informatics, Faculty of Computer Science, University "Dunarea de Jos" of Galati, Romania., corresponding author, phone: +40 724 537594; fax: +40 236 460182, e-mail: Adrian.Filipescu@ugal.ro, Adrian_Filipescu@yahoo.com ;

Grigore Vasiliu is lecturer at the Department of Automation and Industrial Informatics, Faculty of Computer Science, University "Dunarea de Jos" of Galati, Romania, e-mail: vasiliugrigore3@yahoo.com ;

George Coman is Ph.D. candidate at the Department of Automation and Industrial Informatics, Faculty of Computer Science, University "Dunarea de Jos" of Galati, Romania, e-mail: georgecoman2002@yahoo.com

Adrian Radaschin is Ph.D. candidate at the Department of Automation and Industrial Informatics, Faculty of Computer Science, University "Dunarea de Jos" of Galati, Romania, e-mail: radaschinadrian@yahoo.com

II. Background. Brief Overview of the Existing Obstacle Avoidance Algorithms

A. *The Bug Algorithms*

The simplest obstacle avoidance algorithm ever described is called "the bug algorithm" [1]. According to it, when an obstacle is encountered, the robot fully circles the object in order to find the point with the shortest distance to the goal, then leaves the boundary of the obstacle from this point (see figure 1).

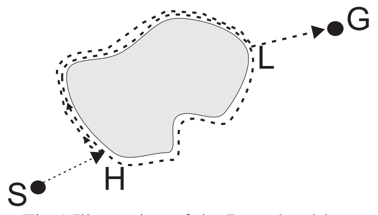

Fig.1 Illustration of the Bug algorithm

This algorithm is obviously very inefficient, and therefore several improvements have been proposed ([2],[3]).

In the 'bug2" algorithm (figure 2), the robot starts following the boundary of the obstacle, but leaves it as soon as it intersects the line segment that connects the start point and the goal.

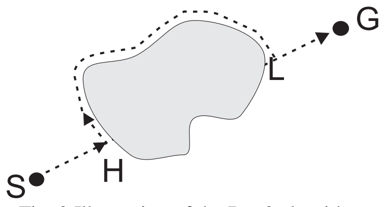

Fig. 2 Illustration of the Bug2 algorithm

Although their simplicity is a major advantage, the bug-type algorithms have some significant shortcomings:

- they do not consider the actual kinematics of the robot, which is important with non-holonomic robots,
- they consider only the most recent sensor readings, and therefore sensor noise seriously affects the overall performance of the robot.
- they are slow

B. The Potential Field Algorithm

While the bug-type algorithms are based on a purely reactive approach, the following algorithms tend to view the obstacle avoidance as a sub-task of the path planning, in a deliberative approach.

The potential field algorithm, described in [4], and [5], assumes that the robot is driven by virtual forces that attract it towards the goal, or reject it away from the obstacles. The actual path is determined by the resultant of these virtual forces (see figure 3).

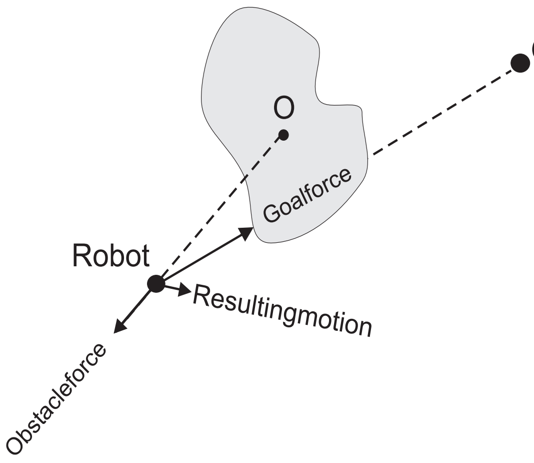

Fig. 3 Illustration of the potential field algorithm

Despite its elegance, this algorithm still does not solve all the drawbacks of the bug algorithms, performs poorly on narrow passages, and is more difficult to use in real time applications.

C. The Vector Field Histogram (VFH) Algorithm

Described for the first time in [6], and later improved in [7] and [8] by Borenstein et al., the Vector Field Histogram, or VFH algorithm overcomes the problem of the sensors noise by creating a polar histogram of *several* recent sensor readings, like the one depicted in figure 4.

In figure 4, the x-axis represents the angles associated with sonar readings, and the y-axis represents the probability P that there really is an obstacle in that direction.

The probabilities are computed by creating a local occupancy grid map of the environment around the robot.

The polar histogram is used to identify all the passages large enough to allow the robot to pass-through. The selection of the particular path to be followed by the robot is based on the evaluation of a cost function, defined for each passage. This depends on the alignment of the robot's path with the goal, and on the difference between the current wheel orientation and the new direction. The passage with the minimum cost is selected.

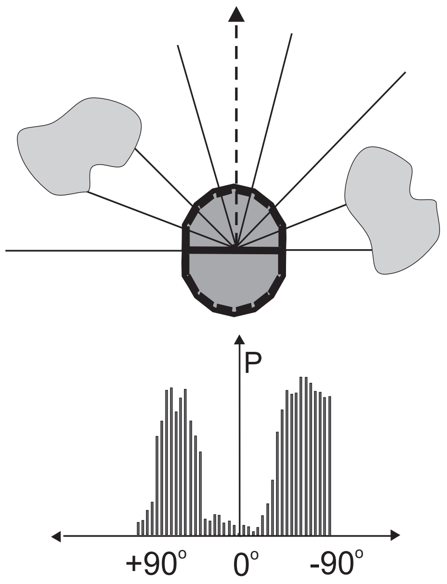

Fig. 4 The polar histogram used in VFH algorithm

This algorithm offers better robustness to sensor noise, and takes into account the kinematics of the robot, but still involves a considerable computation load, which makes it difficult to implement on embedded systems.

D. The Bubble Band Technique

Proposed by Khatib and Quinlan in [9], this method defines a "bubble" containing the maximum available free space around the robot, which can be traveled in any direction without collision. The shape and size of the bubble are determined by a simplified model of the robot's geometry, and by the range information provided by the sensors (see figure 5).

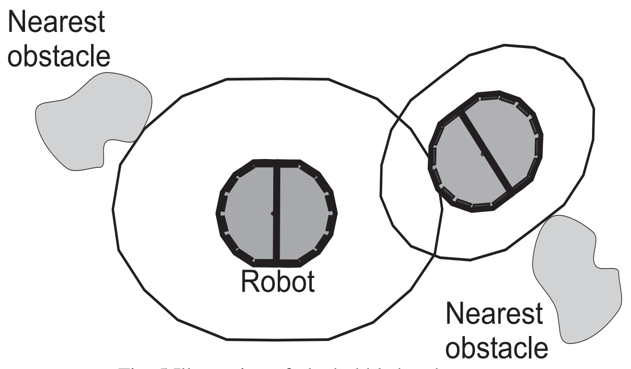

Fig. 5 Illustration of the bubble band concept

With this concept, a *band* of such bubbles can be used to plan a path between a starting point and a goal. Obviously, this technique is more a problem of offline path planning than one of obstacle avoidance, but we have included it in this brief presentation, because the idea of a bubble, seen as a subset of free space around the robot has some similarity with the solution proposed in this paper.

E. *Other Obstacle Avoidance Algorithms*

There are, of course, several other interesting algorithms for obstacle avoidance. However, relatively few of them are suitable for real-time, embedded applications, and will not be discussed here. Among them, fuzzy logic solutions, like those presented in [10], and [11] can be integrated as a natural extension of the fuzzy path following problem, described in [12].

III. Description of the Proposed Algorithm

A. *Detection of Obstacles*

Consider a vehicle, having a ring of equidistant ultrasonic sensors, covering an angle of 180 degrees, as depicted in figure 6.

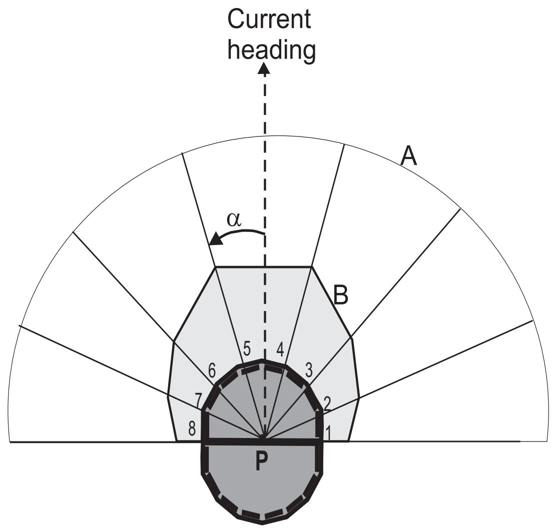

Fig. 6. Robot, ultrasonic sensors, and sensitivity bubble

If N is the number of sonar sensors, the following code defines the sensitivity bubble:

```
unsigned int sonar_readings[N];
unsigned int bubble_boundary[N];
bubble_boundary[i]=Ki*V*delta_t;
int check_for_obstacles(void)
{
for(i=0;i<N;i++)
  {
  if(sonar_readings[i]<=bubble_boundary[i]
      return(1);
  else return(0);
  }
}
```

where, V is the translation velocity of the robot, *delta_t* is the time interval between successive evaluations of sensor data, and Ki are scaling constants, used for tuning. In our experiment, $Ki \in [0.2-3]$.

Note that the shape and size of the sensitivity bubble (curve B in figure 6) defined like this is dynamically adjusted, depending on the distance that can be traveled through by the robot, during the time interval delta_t, provided that the bubble does not exceed the range of the sonar sensors (curve A in figure 6).

Since the ultrasonic sensors are uniformly distributed, covering an arc of 180 degrees, the sonar readings can be represented in a polar diagram, as shown in figure 7.

Note that, in the absence of obstacles, the readings of the ultrasonic sensor are equal to a maximum value, RANGE (5000mm in our experiment). Also note, that the readings of the sensors represent the distance between the actual position of the sensor (which is on the boundary of the vehicle) and the obstacle. Therefore, the above definition of the sensitivity bubble indirectly includes information on the geometric shape of the robot.

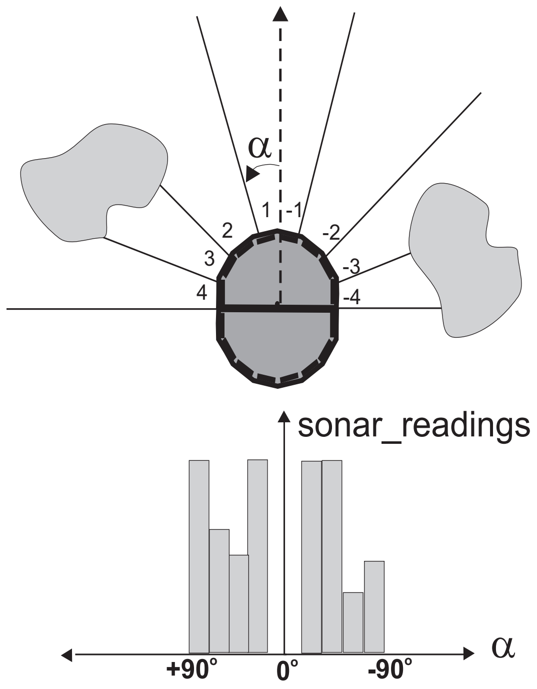

Fig. 7 Polar diagram of the sonar readings

B. *Description of the Algorithm*

Initially, the robot moves straight towards the goal. If an obstacle is detected within the sensitivity bubble, the robots "rebounds" in a direction found as having the lowest density of obstacles, and continues its motion in this new direction until the goal becomes visible (i.e. no obstacle within the visibility range of the sonar in that direction), or until a new obstacle is encountered. The whole process is summarized in the flowchart presented in figure 8.

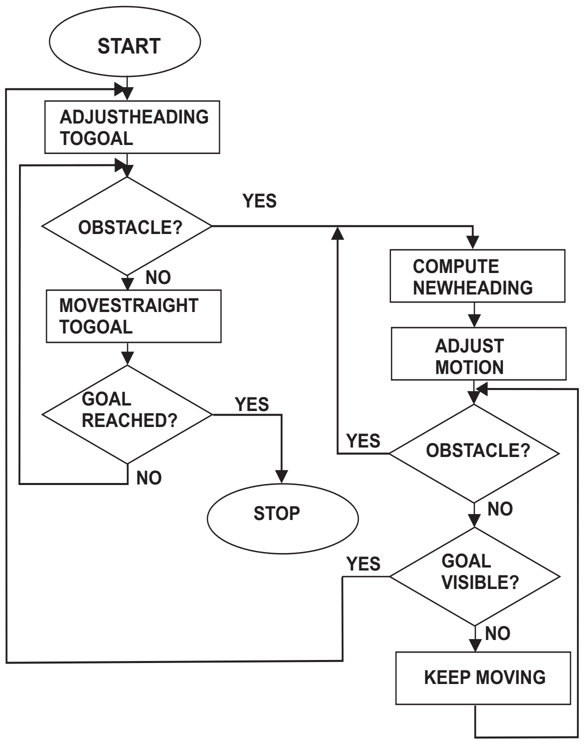

Fig. 8 Flowchart of the obstacle avoidance

Figure 9 presents an illustration of the rebound mechanism. In this image, H is the "hit-point" – the location of the robot at the moment of the detection of an obstacle, and V is the point where the robot regains visibility of the goal.

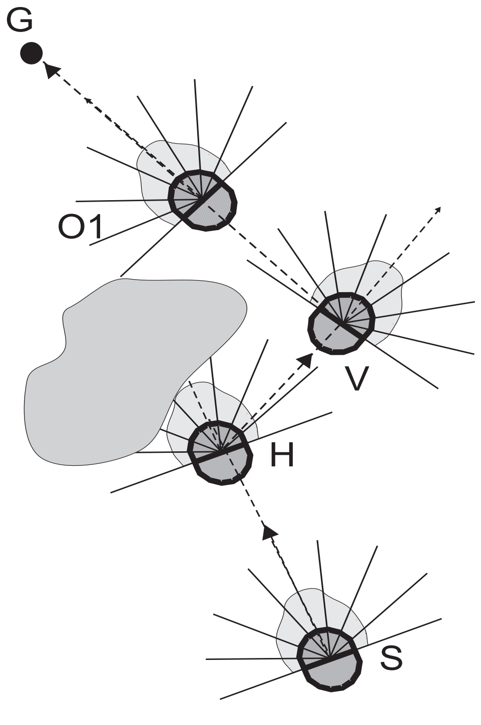

Fig. 9 An example of the rebound process

C. Computing the Rebound Angle

Considering the fact that the sonar cells are uniformly distributed, at an angular pace:

$$\alpha_0 = \frac{\pi}{N} \tag{1}$$

then, the sonar index, i, contains angular information:

$$\alpha_i = i\alpha_0$$
$$\alpha_i \in \left[-\frac{N}{2}, \frac{N}{2}\right] \tag{2}$$

where N is the total number of sonar cells.

With these notations, the simplest way to compute the rebound angle is:

$$\alpha_R = \frac{\sum_{i=-\frac{N}{2}}^{\frac{N}{2}} \alpha_i D_i}{\sum_{i=-\frac{N}{2}}^{\frac{N}{2}} D_i} \tag{3}$$

where D_i is the value reported by the sonar cell i.

Note that this method of computing the rebound angle has no connection with the concept of elastic collision from physics. See figure 10 for an example of how the rebound angle is computed.

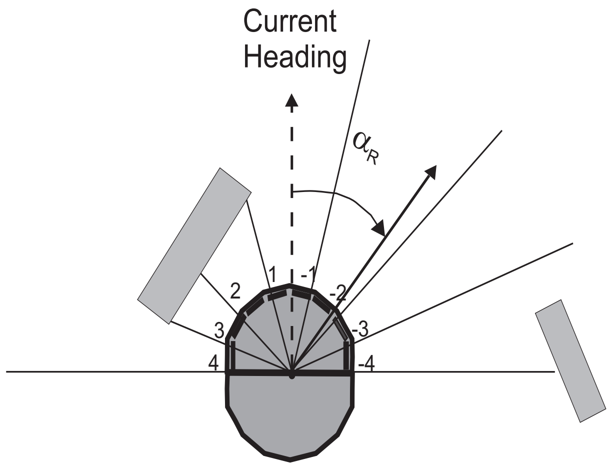

Fig. 10 Computing the rebound angle

This method of weighting the average of sonar position angles by the distances reported by each cell, though very simple and effective, has a drawback. There are situations, when the distribution of obstacles is symmetrical relative to the current heading of the robot, like in figure 11. In these cases, the value computed for the variable *alpha_R* is zero (or PI).

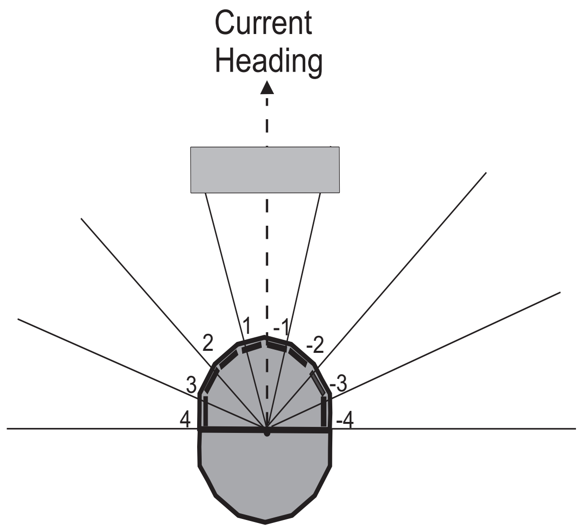

Fig. 11 An example of miscalculation of the rebound angle

A simple workaround is to introduce an additional condition when calculating the rebound angle:

```
if(alpha_R==0) alpha_R=PI/2;
```

IV. Experimental Results

A. General Conditions

This experiment is part of a more comprehensive research, aimed to develop cost effective solutions for real time control of mobile robots, based on embedded systems.

The experiment was designed to work with the robots Pioneer3-DX and PeopleBot, manufactured by MobileRobots Inc. ([13]).

During the simulation phase of the experiment, we have used the robot simulator software *MobileSim*, offered by MobileRobots Inc., specifically designed for the Pioneer3 robots. Various maps of the environment, with multiple distributions of the obstacles were created with the software application *Mapper3*, also from MobileRobots.

The algorithm was also tested using real robots, with satisfactory results.

The actual implementation used a low-cost, 8-bit microcontroller, and was written in C. Besides the obstacle avoidance task, the microcontroller was executing additional tasks for fuzzy path following, and communication, as described in [12].

The average speed of the robots in the experiments was set to 0.4m/s – a value reasonable for a wheelchair moving indoors.

B. Experimental Results

Figure 12 is a snapshot generated with MobileSim, to illustrate the basic obstacle avoidance behavior. Figure 13 shows an example of corridor navigation, and figure 14 shows the trail of the robot while performing a more difficult task, in a maze-type environment, assuming that a higher level global planner, have defined a set of intermediate goals for each path.

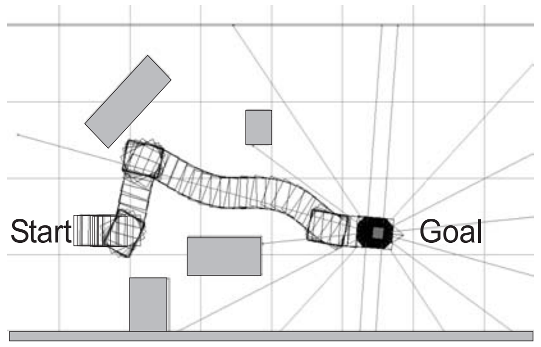

Fig. 12 Basic obstacle avoidance with simulated robot

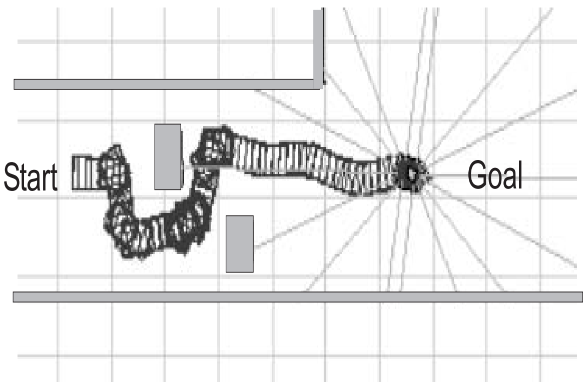

Fig. 13 An example of corridor navigation

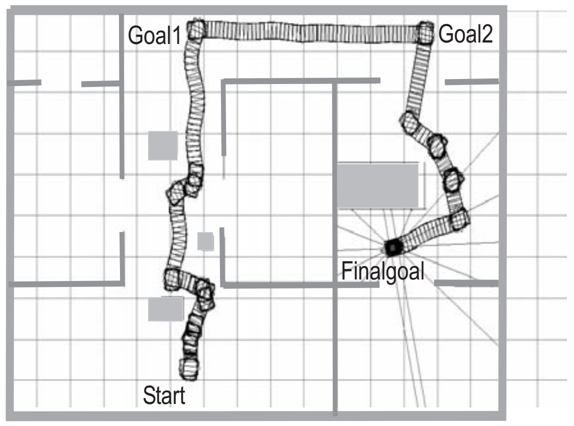

Fig. 14 An example of navigation in maze-type environment, with intermediary goals

Finally, the algorithm was tested with two real robots, namely the models Pioneer3 and PeopleBot of MobileRobots, having the same kinematics model, (see figure 15) in a typical office environment containing a variety of static obstacles, plus human operators moving permanently within the visibility range of the sonars.

The control structure used in the experiment was that described in [14].

The overall performance of the robots in these conditions was satisfactory.

Most of the failures recorded were due to cumulative odometric errors.

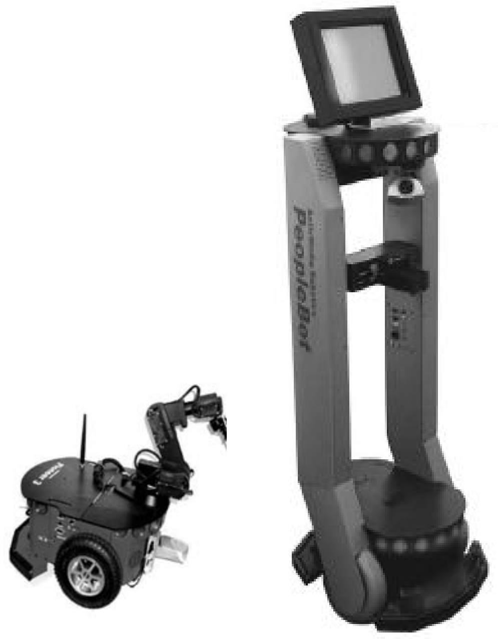

Fig. 15. The robots used in the experiment

V. Discussion

Among the advantages of the proposed solution, we notice:

- It demands very low computational load, and can be implemented on low-cost microcontrollers
- It is capable to avoid any kind of static obstacles, and even some moving obstacles, like walking humans
- It requires low cost sensors
- It can be easily adapted for other sensors, like rotating laser rangers
- It performs very well on narrow corridors
- It compensates, in part, the sensor noise by considering the information provided by a whole set of sensors.

And the major weaknesses are those common for the majority of purely reactive algorithms:

- It is far from being optimal
- Like most purely reactive algorithms, it requires a higher level path planner to perform reasonably well in maze-type environments. For example, in figure 14, the algorithm is incapable to conduct the robot from Start to Final goal, unless a planner can define intermediary goals Goal1 and Goal2.
- The motion is not smooth
- Failure is possible even when a valid path to goal exists
- Motion is attempted even if there is no path to goal.
- It is vulnerable to sensor noise.

Despite the above mentioned drawbacks, the proposed solution may be of interest in applications where a higher level path planner is available, and when the cost criterion is particularly important.

Further work is required to improve the overall smoothness of the motion. This is possible, because at this time, the robot stops and adjusts its heading each time an obstacle is detected. Most of the avoidance maneuvers can actually be executed on the fly, without the need to stop the robot.

The experiments with the simulator, and with real robots suggest that such simple algorithms can be used in the implementation of low-cost embedded control devices for intelligent wheelchairs, and other service robots, contributing to a drastic cost reduction of these equipments.

References

[1] Lumelsky, V., Skewis, T., "Incorporating Range Sensing in the Robot Navigation Function." *IEEE Transactions on Systems, Man, and Cybernetics*, 20:1990, pp. 1058–1068..

[2] Lumelsky, V., Stepanov, A., "Path-Planning Strategies for a Point Mobile Automaton Moving Amidst Unknown Obstacles of Arbitrary Shape," in *Autonomous Robot Vehicles*. New York, Spinger-Verlag, 1990

[3] Kamon, I., Rivlin, E., Rimon, E., "A New Range-Sensor Based Globally Convergent Navigation Algorithm for Mobile Robots," *in Proceedings of the IEEE International Conference on Robotics and Automation*, Minneapolis, April 1996.

[4] Khatib, O., 1985, "Real-Time Obstacle Avoidance for Manipulators and Mobile Robots." *1985 IEEE International Conference on Robotics and Automation*, March 25-28, St. Louis, pp. 500-505..

[5] Koren, Y., Borenstein, J., "High-Speed Obstacle Avoidance for Mobile Robotics," *in Proceedings of the IEEE Symposium on Intelligent Control*, Arlington, VA, August 1988, pp. 382-384.

[6] Borenstein, J., Koren, Y., "The Vector Field Histogram – Fast Obstacle Avoidance for Mobile Robots." *IEEE Journal of Robotics and Automation*, 7, 278–288, 1991.

[7] Ulrich, I., Borenstein, J., "VFH+: Reliable Obstacle Avoidance for Fast Mobile Robots," *in Proceedings of the International Conference on Robotics and Automation (ICRA'98)*, Leuven, Belgium, May 1998.

[8] Ulrich, I., Borenstein, J., "VFH*: Local Obstacle Avoidance with Look-Ahead Verification," *in Proceedings of the IEEE International Conference on Robotics and Automation*, San Francisco, May 24–28, 2000.

[9] Khatib, O., Quinlan, S., "Elastic Bands: Connecting, Path Planning and Control," *in Proceedings of IEEE International Conference on Robotics and Automation*, Atlanta, GA, May 1993

[10] Kim J.H.,Park J.B., Yang H. "Implementation of the Avoidance Algorithm for Autonomous Mobile Robots Using Fuzzy Rules" in *Fuzzy Systems and Knowledge Discovery,* Springer 2006.

[11] Tzafestas S.G. and Zavlangas P. Industrial and Mobile Robot Collision–Free Motion Planning Using Fuzzy Logic Algorithms, Industrial-Robotics-Theory-Modelling-Control, ARS/plV, Germany, 2006, pp. 964, 995

[12] Susnea I, Vasiliu G, Filipescu A**,** Real-Time, Embedded Fuzzy Control of the Pioneer3-DX Robot for Path Following, Proceedings of 12th WSEAS International Conference on SYSTEMS, Heraklion, Greece, July 22-24, 2008, pp.334-338,

[13] www.mobilerobots.com

[14] Susnea I **.** Vasiliu G, Filipescu A, Coman G., On the Implementation of a Robotic Assistant for the Elderly. A Novel Approach, 7th WSEAS Int. Conf. on Computational, Iintelligence Man-machine Systems and Cybernatics (CIMMACS '08)**,** Cairo, Egipt, December 29-31, 2008, pp.215-220