An Interface System for Real-Time Mobile Robot Environment Mapping using Sonar Sensors

J. Aekaterinidis, K. Kostoulakis, L. Doitsidis, K. P. Valavanis[1], N. C.Tsourveloudis

Abstract— **A user friendly graphical tool has been designed to provide in real-time detailed 2D graphical representations of a mobile robot's workspace environment and trajectory. Sonar sensors are used to collect information related to the robot's environment, including obstacle identification. Given the obstacles' distance from the robot, their coordinates are calculated progressively, as the robot moves, and subsequently the whole environment is mapped in real time with minimum time-delays.**

Index Terms — **Skid steering mobile robot, sonar sensor, environment mapping.**

I. INTRODUCTION

The central objective of this paper is to develop an efficient user friendly mobile robot – computer interface system, capable of constructing and duplicating in real-time with minimum time-delays the actual robot trajectory and workspace environment map, relying only on fused sonar sensor data. It is considered that the mobile robot operates in an unknown and possibly hostile (as well as generally remote), obstacle filled environment that needs be explored / investigated and/or studied through the "eyes" of a domain expert who either cannot reach it, or it is too dangerous to work in it. As a first step, it is considered that the mobile robot operates indoors; a 2-D environment map interface is built as the best and most realistic compromise between speed and quality of displayed information.

There has been considerable research in the area of robot environment mapping, but only research very closely related to the proposed approach is reviewed here. In [2] the mobile robot map construction uses data from a sonar-based range sensor, which is based on a 2-D grid consisting of a matrix of cells each containing an occupancy value and a certainty value.

These values are used by the occupancy and localization algorithms, respectively, to construct the map. Preprocessing is required, which delays the whole map building process. In [3] cognitive maps are used to build a map for a navigating robot equipped with sonar sensors. A key issue is the fact that a representation is computed for each local space the robot visits. Representations, connected in the way they are experienced, form the robot's cognitive map. Both approaches, although technically sound, appear quite complex and computationally demanding for real-time implementation. Alternative approaches to map building based on VRML include [4] and [5], however the objectives were the robot environment model building and communication protocol issues, and not actual real-time implementation.

The proposed approach differs from related work in that it derives a very simple, easily implemented in real-time, geometry-based algorithmic process using at each time-step the robot's coordinates (***x, y***), heading angle (***θ***) and the sonar sensors detected minimum obstacle distance to build the workspace environment map including static obstacles location, tracing dynamic obstacle movement and monitoring continuously potential collisions in four main directions, ***front***, ***back***, ***left*** and ***right***, as well as mobile robot rotational and translational speed. The actual real-time sonar sensor based mobile robot controller is a two-layer fuzzy logic controller that combines robot planned, re-planned, reflective and reactive behaviour in dynamic environments. Mobile robot goal-directed behaviour is controlled through controlling robot steering, while reaction-directed behaviour is monitored and controlled through analysis of potential collisions [1]. No assumptions are made on the environment a-priori information, on the shape of obstacles and their velocities. The environment map is constructed and it is viewed "on the fly" as the robot moves within the workspace environment. Therefore, the user, domain expert, observing the environment map through the interface, may command, in real-time, the robot to further investigate/explore a certain area of interest within its workspace. The interface system may operate in two modes, constructing a workspace environment map: i) off-line with user defined information related to robot trajectory, obstacle location, collision possibilities, robot speed, and, ii) on-line where all related information is provided by the actual robot at every time instant. It is obvious that map building may result from a combination of the above two modes, where some information may be a-priori defined (for example, static obstacle locations if known) while the rest is obtained in real-time.

Although the proposed technique may be applied to a variety of mobile robots, the specific robot type used is the ATRV–mini skid steering robot manufactured by *iRobot* (previously known as *RWI*: Real World Interface).

The rest of the paper is organized as follows: Section II

The authors are with the Department of Production Engineering and Management, Intelligent Systems and Robotics Laboratory, Technical University of Crete, Chania, Crete, Greece, GR – 73100.

[1] To whom all correspondence should be addressed. E-mail: k.valavanis@ieee.org

summarizes ATRV-mini characteristics, constraints and presents briefly the designed fuzzy-logic controller. Section III presents in detail the proposed approach while Section IV discusses the interface platform. Results are provided in Section V, while Section VI concludes the paper.

II. Vehicle Configuration and Real-Time Control

The mobile robot ATRV-mini used for experimentation is shown in Figure 1. The vehicle has a ring of ***24*** sonar sensors that are placed around the vehicle and grouped in pairs A_i, $i=1,...,12$, as shown in Figure 2, each with a (manufacturer claimed) maximum measurement range of ***4m***, returning data readings every ***0.1s*** [1]. Exhaustive experimentation and real-time testing has revealed that the sonar sensor maximum effective and reliable measurement range is approximately ***2m***, thus this is the effective sensor range considered in all implementations. The minimum of each sonar sensor pair readings is considered as a distance measure from (potential) obstacles.

Figure 1: DAMON: The ATRV - Mini mobile robot of the Laboratory for Intelligent Systems and Robotics at the Technical University of Crete.

For this robot system, a two-layer Mamdani-type controller has been designed and implemented [1], as shown in Figure 3. In the first layer, there are four fuzzy logic controllers responsible for obstacle detection and collision possibility calculation in the four main directions, ***front***, ***back***, ***left***, ***right***. The four controllers receive as inputs sonar sensor data and return as output the respective direction collision possibility. Data from group sensors A_1, A_2,..., A_5 (5 inputs) and group sensors A_7, A_8,..., A_{11} (5 inputs) serve as inputs to the individual controllers responsible for the calculation of the ***front*** and ***back*** collision possibilities, respectively. Data from group sensors A_5, A_6, A_7 (3 inputs) and group sensors A_{11}, A_{12}, A_1 (3 inputs) serve as inputs to calculate the ***left*** and ***right*** possibilities, respectively. The individual fuzzy controllers utilize the same membership functions to calculate the collision possibilities. The possibilities calculated in the first layer are the input to the second layer along with the angle error (the difference between the robot heading angle and the desired target angle taking values ranging from -180^0 to 180^0); the output is the updated vehicle's translational and rotational speed. The second layer fuzzy controller receives as inputs the four collision possibilities and the angle error, and outputs the translational velocity (responsible for moving the vehicle backward or forward) and the rotational speed (responsible for the vehicle rotation). Design and implementation details may be found in [1].

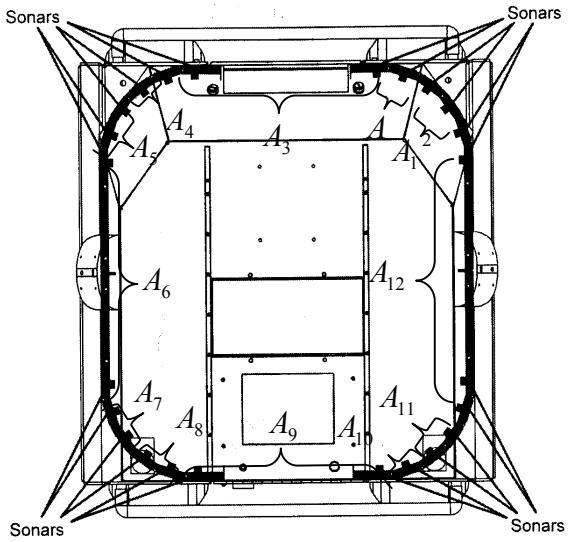

Figure 2: Sonar sensor grouping

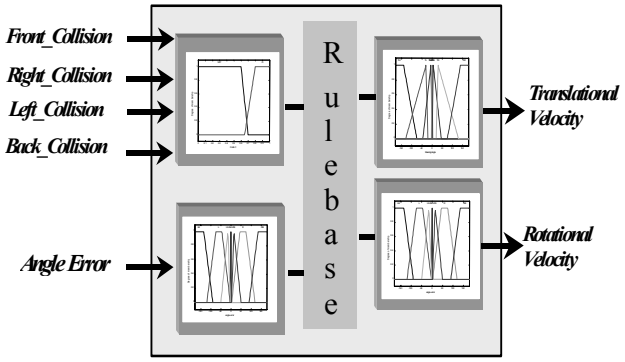

Figure 3: Block diagram of the fuzzy logic controller

III. THE PROPOSED APPROACH

As previously stated, the main objective is to develop a graphic interface system by deriving a simple and efficient algorithmic process that builds the robot's workspace environment map in real-time with minimum time delays. The single source of information is the data provided by the robot ring of sonar sensors.

In principle, considering sonar sensor data readings that correspond to measurements of the minimum obstacle distance they detect within a given range, and knowing or calculating the robot's coordinates at every time instant (in this case every ***0.1s***) one may compute obstacles' coordinates and plot the corresponding points at every time step. This

process, when repeated for the duration of the robot's movement, results in a map of the robot workspace environment. Stated differently, knowing a point's distance from given sonar and the coordinates of that sonar, the point's coordinates may be computed by simple geometric calculations.

The ATRV-mini sonar sensor ring configuration is shown in Figure 4. Each sonar ***i*** forms an angle φ_i (from φ_0 το φ_{23}) with the robot vertical axis; R_i is the distance between sonar ***i*** and the vehicle's center.

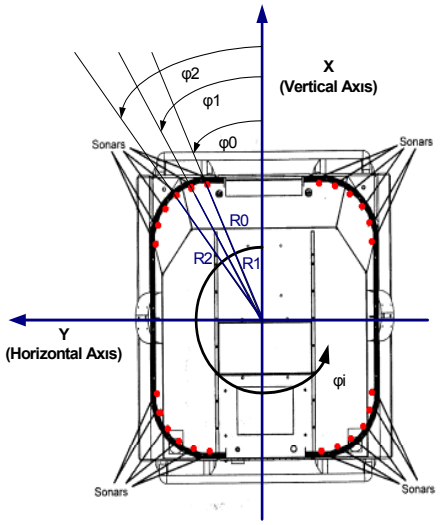

Figure 4: Sonar ring arrangement

Each sonar sensor has a *viewing cone* within which it "searches" for an obstacle. Further, the energy radiated by a sonar is not focused in a straight line. Instead it is distributed within the viewing cone, its size measured by the beam-width angle which, in this case is 30^o as shown in Figure 5. This means that although each sonar is capable of detecting obstacles within this cone, this capability results in a large angular uncertainty of the range reading. To overcome this uncertainty, it is assumed for simplicity purposes that when a sonar "detects" an obstacle, the latter lies in the main direction of the cone. This assumption is valid and realistic since the effective sonar range is small (***2m***) and the corresponding viewing cone arcs are small, too. For example, if the sonar is pointing at 90^o the actual reflection might be coming from an obstacle somewhere between 75^o and 105^o. The previous assumption dictates that the obstacle is detected at the 90^o direction.

The geometric configuration of a skid steering mobile vehicle in the ***X-Y*** plane is shown in Figure 6, where θ is the vehicle rotational angle, φ_i is the angle of sonar ***i*** from the vertical axis of the vehicle and R_i is the distance between the sonar and the vehicle center. Assume the vehicle starts from position $(x, y) = (0, 0)$ and $\theta=0$, and after a short period of time its position becomes $x=x_1$, $y=y_1$ and $\theta=\theta_1$. The obstacle coordinates (x_{ob}, y_{ob}) detected by sonar ***i*** may be calculated as:

$$x_{ob} = x_1 + (d_3+R_3)\cos(\varphi_3+\theta_1)$$
$$y_{ob} = y_1 + (d_3+R_3)\sin(\varphi_3+\theta_1)$$

In general, given the vehicle position as (x, y, θ) the obstacle coordinates detected by sonar ***i*** are calculated by the equations:

$$x_{ob} = x + (d_i+R_i)\cos(\varphi_i+\theta)$$
$$y_{ob} = y + (d_i+R_i)\sin(\varphi_i+\theta)$$

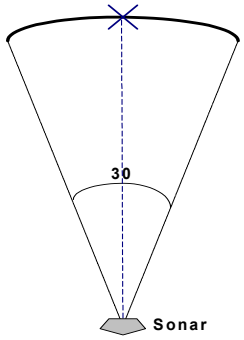

Figure 5: Sonar sensor viewing cone

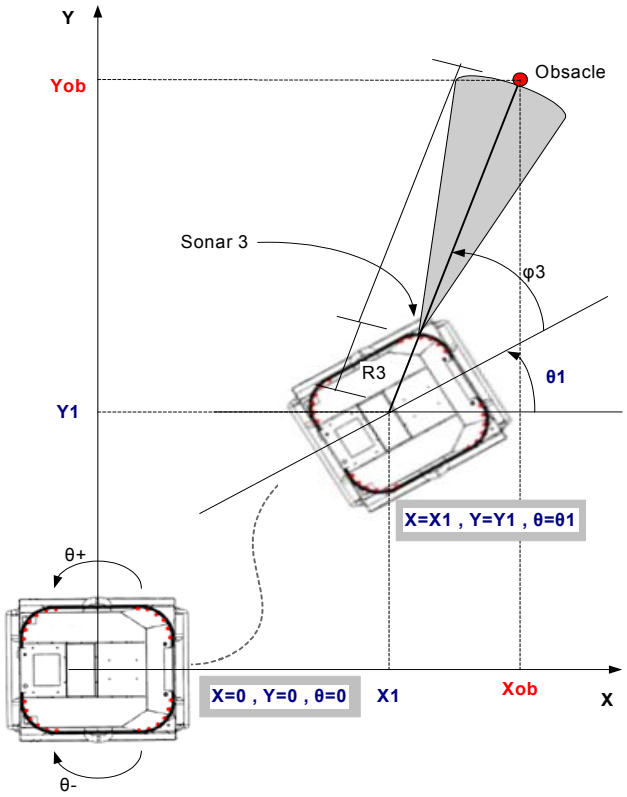

Figure 6: Example of coordinate's computation.

IV. THE INTERFACE PLATFORM

The **R**obot **E**nvironment **M**apping **U**sing **S**onars (REMUS) application has been developed using the ***RedHat 7.2 Linux*** operating system. The software has been written in ***C***++ along with the ***g2*** 2-D graphics library [7]. REMUS is equipped with an easy to use graphical user interface, developed with the ***Qt*** toolkit. If and when necessary, the ***Mobotsim*** simulator [8] was used for simulation purposes and testing of the REMUS system.

A. Description

The main task of the REMUS application is to represent the environment around the moving vehicle based on the values returned from sonars. Thus the main input data are the coordinates x and y of the vehicle, the rotational angle θ and the 24 distance values returned from the vehicle's sonars.

Nevertheless, extra functionality was added to REMUS. Thus the user can also view a plot of the collision probabilities (at left, right, front and back directions) computed by the fuzzy logic algorithm for the navigation of the vehicle as described in [1], as well as the angular and linear velocity. Last but not least, the user can supply the coordinates of known obstacles in order for them to be displayed in the case where the main concern is the vehicle movement and not the environment mapping. With this option the user can also verify whether the mapping results by the robot agree with the original configuration of the obstacles.

B. Application's GUI

The application's interface consists of three main areas (tabs). The first one (***Input Files Tab***), Figure 7, deals with the input files supplied from the user. These are:

i. The *Position Input File,* which contains the ***x, y*** coordinates and the rotational angle of the vehicle at each instance.
ii. The *Sensor Input File,* which contains the 24 distance values returned from the sonars at each instance.
iii. The *Velocities / Probabilities Input File,* which contains the left, right, back and front collision probabilities as well as the linear and angular velocities of the vehicle, at each instance.
iv. The *Predefined Objects' Coordinates Input File,* which contains the coordinates of known rectangular obstacles lying around the vehicle.

It should be made clear that the only required field is the *Position Input File.* The other three provide extra functionality and are optional, so the user may not supply the corresponding filenames for them if he/she so wishes.

The second tab (***Visual Options Tab***), Figure 8, consists of three checkboxes, which relate to the visual preferences of the user. More precisely the user may choose or not, to view the vehicle's trajectory, the known obstacles around the vehicle, and the velocity/collision probabilities plots.

The third tab (***Preferences Tab***), Figure 9, consists of 7 input fields where the user can define the vehicle's dimensions, the floor dimensions, the maximum viewing distance of the sonar, the sample rate with which the vehicle transmits data and the sonar angle file. The maximum viewing distance of the sonar determines the threshold where distances with greater value are ignored (see section III.A for more details). The sonar angle file contains the angle that each sonar forms from the vertical axes (φ_i in Figure 4). In the case where the application is used to display the environment around an ATRV-Mini robot, the user can use the sensor angle file corresponding to this robot by selecting the appropriate robot type. In other cases the user should define the sensor angle file, which corresponds to the robot used for his/her experiments.

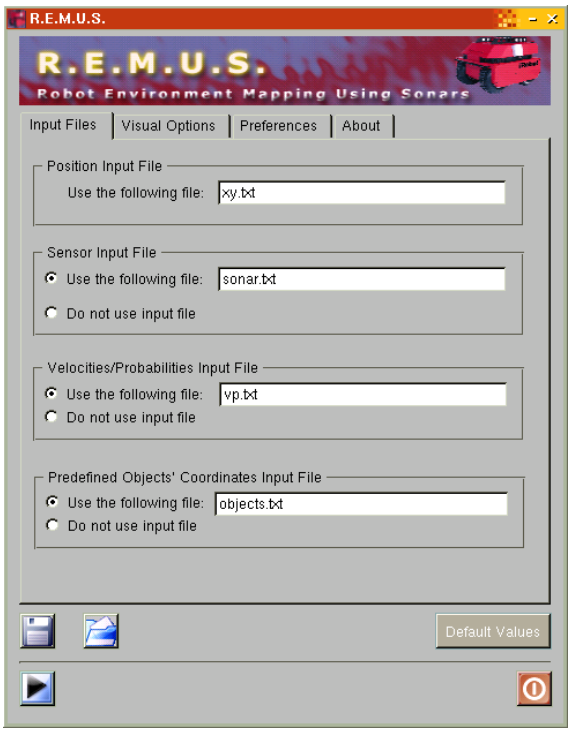

Figure 7. The *Input Files* Tab

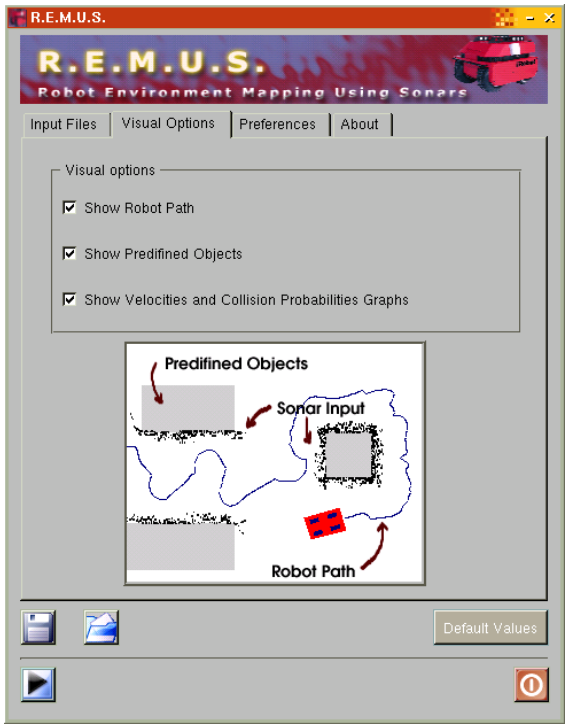

Figure 8. The *Visual Options* Tab

After defining the appropriate input files and application's options, the user can press the button at the bottom left corner in order to start the mapping of the environment around the vehicle. It should be noted that the user has the ability to save or load his/her preferences, as well as loading default values corresponding to the ATRV-Mini configuration.

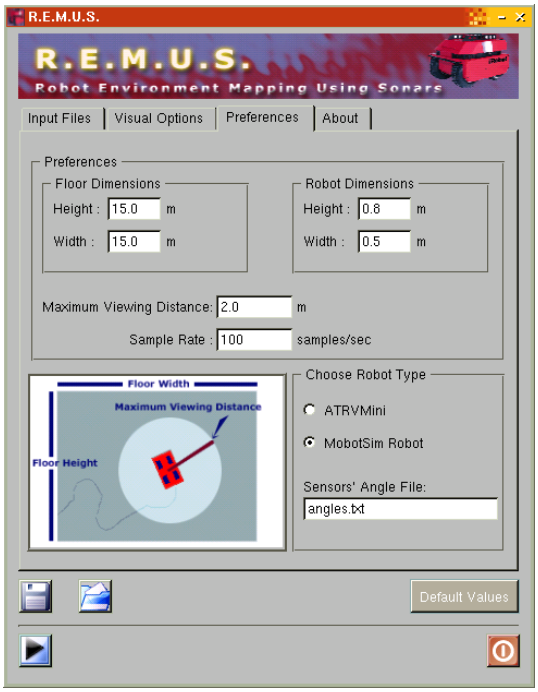

Figure 9. The *Preferences* Tab.

C. *Main Application's Window*

The main window of the application consists of three parts as seen in Figure 10.

At the left column three plots are visible: the first one corresponds to the linear and angular velocities of the vehicle, the second, to the left and right collision probabilities and the third, to the front and back collision probabilities.

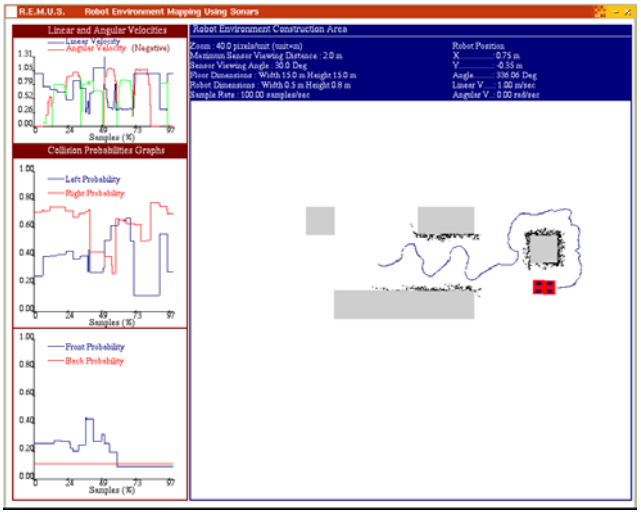

Figure 10. The main window of the REMUS application

At the top blue row the user can be informed about some essential parameters of the application. These are input parameters such as: the sensors' maximum viewing distance, the floor dimensions, the robot dimensions and the sample rate. Apart from these values the user can also be informed about the x, y coordinates of the vehicle, the rotational angle θ and the linear/angular velocities.

Figure 11. Obstacles at known positions.

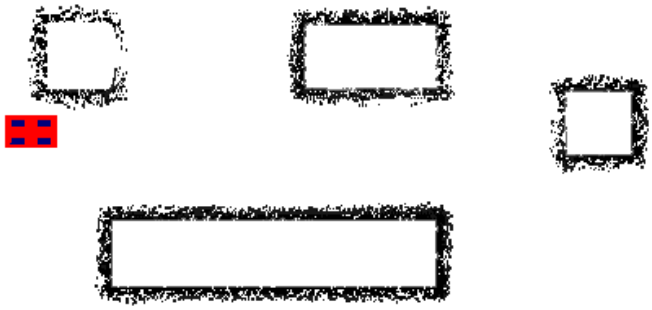

Figure 12. Vehicle environment mapping.

V. RESULTS

A. *Test Cases of Environment Mapping*

In Figures 11 and 12, a test case of the environment mapping can be seen. The input data for this experiment was collected using the Mobotsim 1.0 mobile robot simulator [8]. The difference between Figures 11 and 12 is the visibility of the known obstacles (gray rectangles).

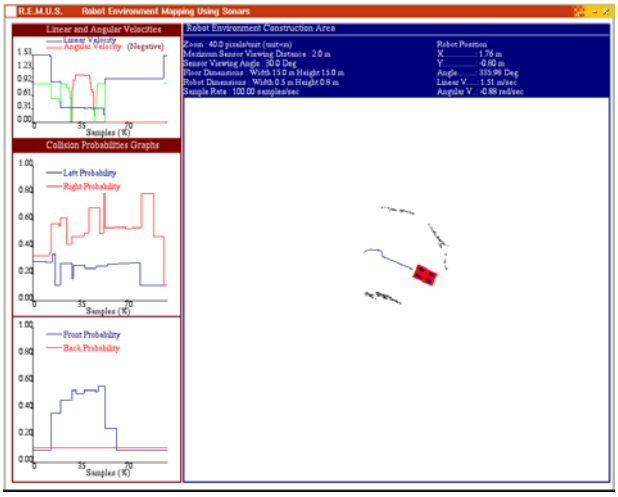

Figure 13. Environment mapping with the ATRV-Mini robot. The maximum sonar viewing distance value was set to 3.0m

As noted earlier (in section III) our approach is simple enough as we draw a black dot at the point where we believe that the sonar beam stroke. It is clear in Figure 12 that the

environment mapping is accurate enough and gives us a clear picture of the obstacles around the vehicle.

Several additional experiments have also been performed, based on input data derived from the ATRV-Mini vehicle sonars. Results agree with those obtained using the Mobotsim simulator. In Figure 13 the environment mapping of the ATRV-Mini robot with 3 obstacles around it is presented. One may observe at the left side of the figure the plots of the linear and angular velocity and the left, right, back, and front collision probabilities. Due to the inaccuracy of the sonars it was also observed that "small objects", not supposed to be there, have been mapped – illustrated with arrows in Figure 14. However, having in mind that, as the maximum viewing distance threshold increases the sonar error increases too, one may avoid this phenomenon by adjusting appropriately this threshold. The threshold value used at the environment mapping in Figure 13 was ***2.0m*** while in Figure 14 it was ***3.0m.***

B. Application's Running Modes

The REMUS application can be used either in real time or by supplying the required input files. In the real time mode the application is connected directly to the ATRV-Mini robot through UNIX sockets. Thus the user is able to view in real time the mapping of the environment around the vehicle. In the non-real time mode the user must supply basically the *position input file*.

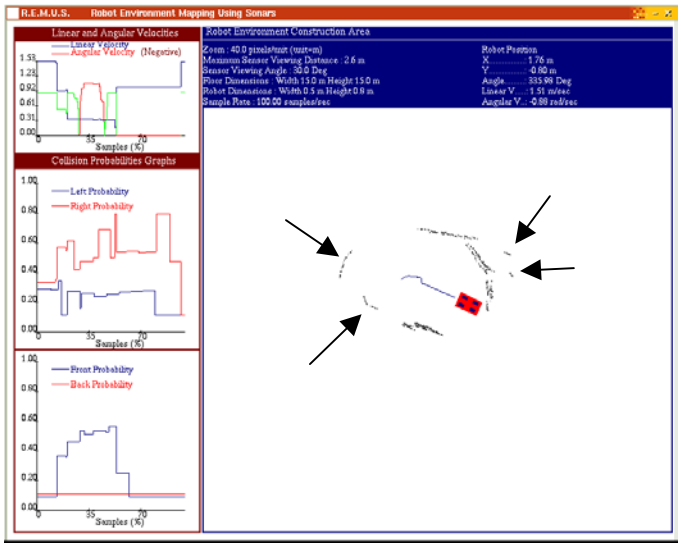

Figure 14. Environment mapping with the ATRV-Mini robot. The maximum sonar viewing distance value was set to 3.0m.

VI. Conclusion

Although this application can already be used on its own for experimenting and testing in robot simulations, its combination with more sophisticated mapping algorithms would yield better results. However, the issue of complexity is still a factor that should be considered when we are talking about real time results. Some future improvements include a better user interface, more extensive tests with real robots and more accurate sonar functionality.

References

[1] L. Doitsidis, K. P. Valavanis, N. C. Tsourveloudis, "Fuzzy Logic Based Autonomous Skid Steering Vehicle Navigation", *Proceedings of the 2002 IEEE International Conference on Robotics & Automation*, Washington DC, USA, pp. 2171-2177, 2002.

[2] V. Varveropoulos, "Robot Localization and Map Construction Using Sonar Data", *The Rossum Project* – http: //rossum.sourceforge.net.

[3] M. E. Jefferies, W. K. Yeap, L. Smith, D. Ferguson, "Building a Map for Robot Navigation Using a Theory of Cognitive Maps", *Proceedings of the IASTED International Conference on Artificial Intelligence and Application,* Marbella, Spain, pp. 348-353, 2001.

[4] J. Howell, B. R. Donald, "Practical Mobile Robot Self-Localization", *Proceedings of the International Conference on Robotics and Automation*, San Francisco, CA, April 24-28, pp. 3485-3492, 2000.

[5] M. Matijasevic, K. P. Valavanis, D. Gracanin, I. Lovrek, "Application of a Multi-User Distributed Virtual Environment Framework to Mobile Robot Teleoperation over the Internet," *Machine Intelligence & Robotic Control*," vol. 1, no.1, pp. 11–26, 1999.

[6] M. Matijasevic, D. Gracanin, K. P. Valavanis, I. Lovrek, "A Framework for Multi-user Distributed Virtual Environments", *IEEE Transactions on Systems, Man, And Cybernetics—Part B: Cybernetics*, vol. 32, no. 4, pp. 416–429, 2002.

[7] G2 2D graphics library. Available: http: //g2.sourceforge.net/

[8] Mobotsim 1.0, Mobile Robot Simulator, http: //www.mobotsim.com