Brazil-VR 09 Team Description Paper

Esther Luna Colombini[1], Alexandre da Silva Simões[2], Thiago de Freitas Oliveira Araújo[3], Aristóteles Terceiro Neto[3], Walter Guerra[3], Gustavo Prado[2], Jackson Paul Matsuura[1], and Antônio Marcus Nogueira Lima[3]

[1] Technological Institute of Aeronautics (ITA) - Brazil
[2] São Paulo State University (UNESP) - Brazil
[3] Federal University of Campina Grande (UFCG) - Brazil
contact:esther@ita.br

Abstract. This paper details the Brazil-VR team designed to accomplish the task proposed by the Virtual Robots Rescue Simulation Competition 2009. In previous Suzhou-08 competition, our team started the development of a completely new controller, which was not based on any other team's architecture. In this way, a development from the lowest to the highest controller level started. Based on the first experiences with the new controller in Suzhou-08 and RoboCup Brazil-08 competitions, we projected some important advances that are now being implemented in Brazil-VR 2009 team. First of all, all low level layer software was rebuild using the python language. The Graphical User Interface (GUI) based on the cross-platform WxWidgets was completely redesigned in order to improve usability and allow an easier user control while running robots. A plugin-based architecture was developed, allowing basic algorithms (like movement, victims detection, SLAM and so on) to be switched on-line. Some navigation algorithms like a neuro-fuzzy algorithm were implemented. Finally, a more powerful SLAM algorithm able to significantly reduce the robot error estimation was developed. This paper presents the Brazil-VR 09 team controller overview and also discuss some of the first results allowed by this approach.

1 Introduction

In order to deploy robots in a wide variety of applications, a way to program them efficiently has to be found. The possibility of having them working in an unsupervised way in complex, harsh or dangerous tasks that need interaction with an unstructured environment has increased the interest for their use.

Urban search and rescue (USAR) is a field where robust, easy to deploy robotic systems can be applied to face emergency situations. In this context, RoboCup competition has provided many benefits to USAR. More specifically, the Virtual Robots Rescue League was developed allowing pos-catastrophe situations to be held by simulated real autonomous robots.

The simulator used in this competition, USARSim, allows high fidelity simulations of multi-robot systems. It currently offers the possibility to simulate commercial as well as self-developed robot platforms. In this situation, a team of agents is responsible for mapping the unknown environment, searching for possible victims.

Because of the complexity of the problem proposed by the Virtual Robots Rescue task, many aspects of robotics such as mapping, localization, vision, sensor fusion and multi-agent coordination, have to be treated properly [1].

The Virtual Robots Rescue Task is a category in Robocup that intends to promote a closer interaction between real and simulated leagues by emulating real comercial/non-commercial robots platforms that should perform a rescue task in a pre-defined high-fidelity simulated environment built based on a real one. In this scenario, a team of heterogenous robots are asked to look for victims in this unstructured environment, to map it and to provide victims localization and health status. In this task, teams are encouraged to develop multi-agent groups that are allowed to exchange as much information as they want via TCP protocol, in a simulated wireless.

This paper presents the Brazil-VR 09 team controller overview designed to the Virtual Robot taks and also discuss some of the first results allowed by this approach. The paper is organized as follows: section 2 presents the proposed controller architecture. Section 3 details the strategy adopted in our approach to accomplish the Virtual Robots Competition task, focusing on the SLAM algorithm. Finally, section 4 presents our validation tests carried out. Section 5 concludes present work.

2 Brazil-VR Controller Platform

2.1 Development guidelines and history

USARSim is as a high fidelity simulator of robots, sensors and environments that can be used as a research tool for the study of Human Robot Interaction (HRI) and multi-robot coordination [2]. Brazil-VR team has as it main objective to develop a mobile robot research platform able to spread the use of the USARSim simulator around the country as a research platform on Artificial Intelligence and Robotics through the use of the controller built over the simulator. The main controller guidelines are in order to reach: *i)* a robust system mainly designed to the autonomous robots operation and research, allowing – but not dependent of – the guiding of a human operator; *ii)* a fully windows/linux/macos open-source cross-platform system; *iii)* an architecture that allow the on-line switching from a bank of control and strategy algorithms. Understanding that available controllers did not reach these characteristics, our team decided to development a completely new controller platform to the VR task, instead of adopting other team's architectures. In this way, a development from the lowest to the highest controller level started in 2008.

In the first version of this control system [3] a basic architecture – described in next section – was proposed. The system was completely developed in C/C++ language. The Graphical User Interface was based on the open-source cross-platform WxWidgets. Based on the first experiences with the new controller in

Suzhou-08 and RoboCup Brazil-08 competitions, we projected some important advances that are now implemented in Brazil-VR 2009 team. First of all, all low level layer software was rebuild using the python language. The Graphical User Interface (GUI) based on the WxWidgets technology was completely redesigned in order to improve usability and allow a easier user control while running robots. A plugin-based architecture was developed, allowing basic algorithms (like movement, victims detection, SLAM and so on) to be switched on-line. Finally, a more powerful SLAM algorithm was projected and is able to significantly reduce the robot error estimation. The system with the new improvements is described in next sections.

2.2 Architecture overview

The Brazil-VR Controller architecture is depicted in figure 1.

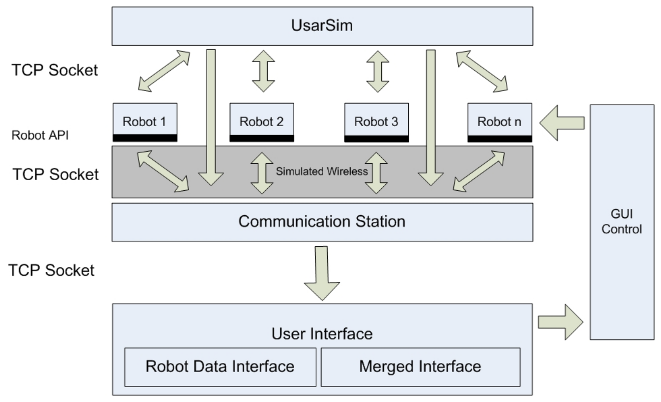


Fig. 1. Brasil-VR Controller Architecture

In the controller, each robot runs its own control and has its own visualization tools. They also communicate among each other and to the system general visualization tool (a merged interface) through the communication station, that simulates a wireless network in the environment.

In 2009 version, all our low-level code – including sockets, threads and so on – was implemented in python language. This approach shows to be more suitable and easier to this task then our previous pure C++ approach. In the controller core we decided to keep the original C/C++ approach. However, we adopted a plugin-based architecture in order to allow the on-line switching of all kind of control algorithms. Under this new architecture, all moving, SLAM and victims detection algorithms, for example, can now be written under any language, and loaded as a plugin in our environment. In order to support these changes and to allow a better usability to the possible human operator, the

Graphical User Interface based on WxWidgets libraries was completely rebuild. The machine-human communication was designed to be based on icons allowing a quick inspection of the robots situation. Monitoring tools were designed to alert the operator under specific user-defined situations or to execute specific algorithms under the same situation. All setup can be changed on-line, which is an important improvement.

The class structure for the controller built is basically the same previously proposed and is shown in Figure 2. In this class diagram the Robot class can interact to the manual or the automatic control. The inherited robot classes that represent specific robot models maintain relationships with every single sensor it is composed of and with an individual visualization tool. This tool maintain all the data regarding the robot sensors status in the environment. Figure 3 shows samples of the robot individual controller, the controller plugins algorithms selector and the robots sonars and laser viewers.

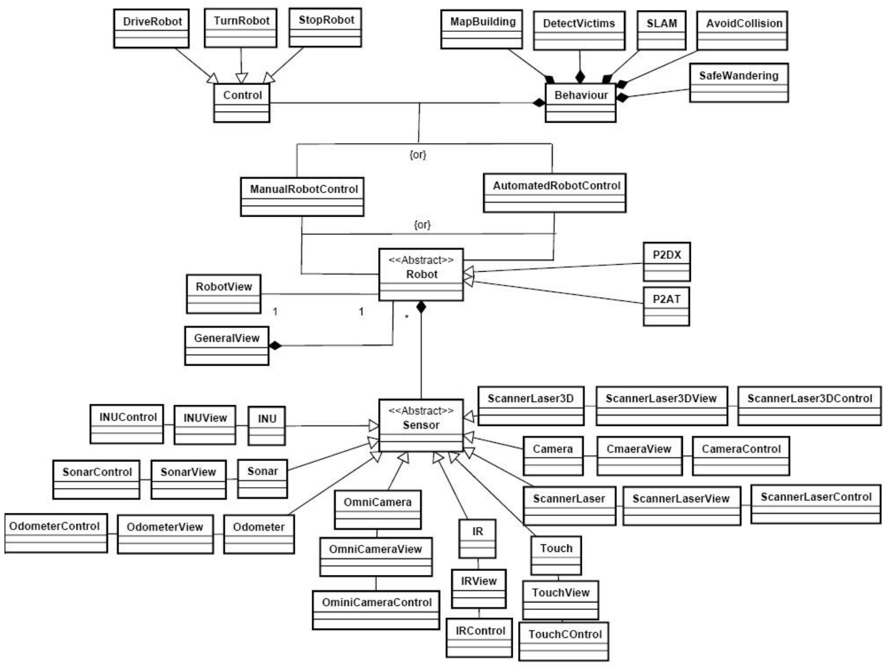
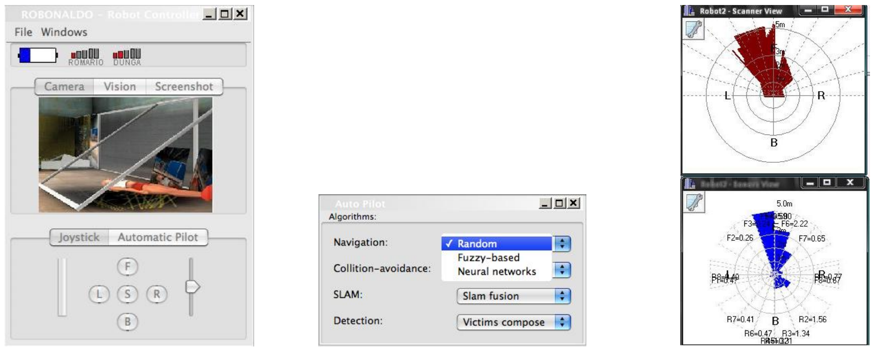


Fig. 2. Brasil-VR Controller Basic Class Diagram

In a general visualization tool – that concentrates information coming from all robots in the environment – it is possible to keep track of the world map generated, the robot current poses and battery levels and the identified victims



Algorithms:

Navigation: ✓ Random / Fuzzy-based / Neural networks

Collition-avoidance:

SLAM: Slam fusion

Detection: Victims compose

(a) (b) (c)

Fig. 3. Screenshots of the control system: a) The robot individual controller: icons allow easy and quick identification of battery and communication level with other robots; b) Algorithms loaded as plugins can be selected by operator in running time; c) laser scanner and sonnar viewers.

with their estimated positions. It is also possible to save the actual map into specific required formats, to control the simulation elapsed time as well as the communication levels.

2.3 Robots

Because the robots are asked to run both in indoor and outdoor environments, two robot platforms were chosen to composed Brazil-VR team. The first, more indicated to indoor environments is P2DX, a 2-wheel drive pioneer robot from ActivMedia Robotics [4]. The second robot model, the P2AT, is a 4-wheel drive outdoor all terrain robot vehicle also developed by ActiveMedia. Both robots are equipped with a variety of sensors listed in the Virtual Robots Rules for the 2009 competition. The team is mostly heterogeneous, both in robot models and used sensors. Aerial robots are also under implementation.

3 Proposed Control Strategies

The architecture of the control system proposed can be seen is Figure 4. It describes a multi-agent system where the robots should exchange information to decide what to do next. It can be noticed that, for each robot, the sensors data and the information provided by the other agents are used to decide which behavior (action) should be executed. The behaviors work in a multi-threading way. It means that any of them can listen to the sensors at the same time, although there is a priority structure to define which one is to be executed under some circumstances.

Depending on the chosen behavior, the robot can update its map, the victims list or its own localization in the world. These new information are passed to the other robots. This kind of cooperation is important to avoid multiple robots exploring closing areas.

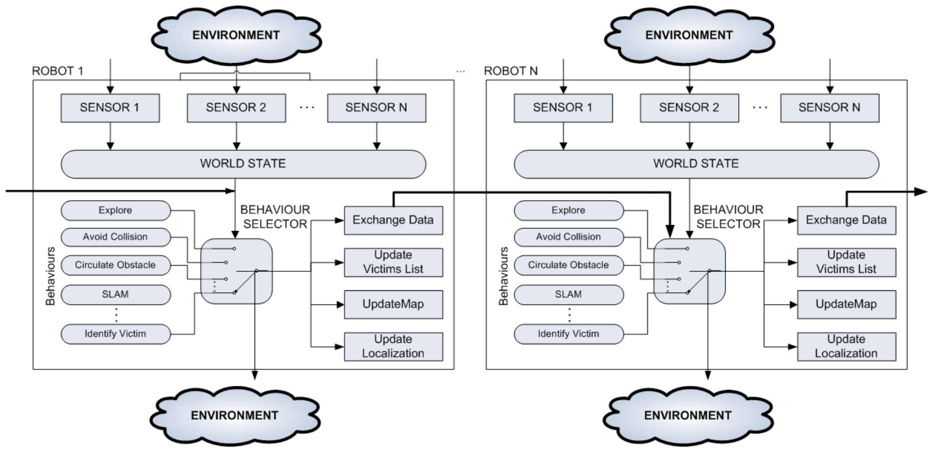


Fig. 4. Proposed System

Executing the controllers can cause specific effects in the robot situation such as: affect the map that is under construction, change the robot localization or even influence other robot decisions.

3.1 Autonomous Navigation

Although the SLAM module is responsible for mapping the environment and for recovering the robot pose in it, the autonomous navigation of each robot uses behaviors such as Explore, AvoidCollision, CircleObject and CVM [5] to navigate safely and efficiently. Some of these controllers parameters where adjusted by hand and others where adjusted using traditional MLPs [6] or fuzzy control [7, 8]. A neuro-fuzzy control was particularly proposed as the robot main navigation algorithm.

3.2 Visual Victims Detection

One of the most important goals of the Virtual Robots task is to properly detect victims and to establish their localization in the world. To accomplish this task an online visual detection algorithm was implemented.

Many approaches were evaluated. Among them are a multi-layer perceptron trained by the backpropagation algorithm [6], the threshold method, the neuro-fuzzy technique proposed in [9]; the spiking neural network with radial basis function used by [10]. The obtained results from each technique were compared to define the best compromise between execution time and classification efficiency. Tests involving environment lighting differences, victims skin colors, victims skin exposition and victims positions were carried out.

Some of the results can be seen in Figure 5. The (a) image shows the original image whereas the (b) image indicates the successful classification.

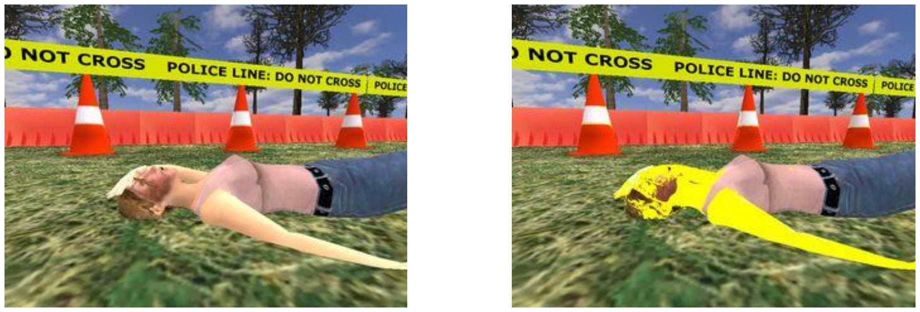


Fig. 5. a) Victim screenshot b) Victim resultant identification

3.3 Simultaneous Localization and Mapping

We know that one of the major concerns in robot navigation is the simultaneous localization and mapping. Currently many techniques are found to solve this problem. Errors on the association of the Map References are catastrophic, being one of the major issues faced by our team on the competitions in 2008. Our actual SLAM is based on modifications on the EKF-SLAM, that consists on estimating the robot and landmarks pose using an extended Kalman Filter, where the robot position and the landmarks are correlated using a covariance matrix. We use two versions of the EKF, one based on robot kinematics model and the other based on dead-reckoning [11].

The strategy is to use sensor fusion and parallel-running of the two versions to minimize the covariance of the error on robot pose. The careful sensor association is of great importance as it permits to eliminate accumulated errors. Moreover, the certainty of the data is guaranteed for map construction and characteristics detection. Our version currently works quite reasonably on outdoor and indoor environments, just needing some changes on the system's input.

It is evident that any of the Kalman filter based Slams requires knowledge about the noise characteristics, and also that in reality the covariances may not always be available correctly. Because of that, and although the values are usually known on the maps provided, based on [12], we are adapting an auto tuning algorithm for the Virtual Robots scenario and incorporating it to our Slam algorithm in such a way that our Slam may be able to adapt its characteristics according to the environment's response trying to maximize the performance of the algorithm.

A currently on-going implementation is the well-known FastSlam, combining the Rao–Blackwellized particle filter (RBPF) with the improved parameters of our EKF implementation. Also, the plan to the competition is to achieve the implementation of an Unscented FastSLAM implementation, now using the UKF to remove some filter inconsistencies [13], and a concurrent study is also being conducted on SVD based Slams for analyzing which algorithm is the best suitable for our needs [14].

The SLAM model adopted in our approach is detailed bellow:

$$\mathbf{u}\left(k\right) = \begin{bmatrix} V\left(k\right) \\ Ve\left(k\right) \end{bmatrix}$$

$$\mathbf{x}_v\left(k\right) = \begin{bmatrix} x_v\left(k\right) \\ y_v\left(k\right) \\ \phi_v\left(k\right) \\ xi(k) \\ yi(k) \end{bmatrix}$$

$$\mathbf{x}_v\left(k+1\right) = \mathbf{f}\left[\mathbf{x}_v\left(k\right), \mathbf{u}\left(k\right)\right]$$

$$\mathbf{f}\left[\mathbf{x}_v\left(k\right), \mathbf{u}\left(k\right)\right] = \mathbf{x}_v\left(k\right) + T\begin{bmatrix} \frac{(V_d(k)+V_e(k))}{2}\cos\left(\phi_v\left(k\right)\right) \\ \frac{(V_d(k)+V_e(k))}{2}\sin\left(\phi_v\left(k\right)\right) \\ \frac{(V_d(k)-V_e(k))}{B} \\ 0 \\ 0 \end{bmatrix}$$

$$f_1\left(k\right) = x_v\left(k\right) + T(\frac{\left(V_d\left(k\right) + V_e\left(k\right)\right)}{2}\cos\left(\phi_v\left(k\right)\right)$$

$$f_2\left(k\right) = y_v\left(k\right) + T(\frac{\left(V_d\left(k\right) + V_e\left(k\right)\right)}{2}\sin\left(\phi_v\left(k\right)\right)$$

$$f_3\left(k\right) = \phi_v\left(k\right) + T(\frac{\left(V_d\left(k\right) - V_e\left(k\right)\right)}{B})$$

$$f_4\left(k\right) = x_i(k) + 0$$

$$f_5\left(k\right) = y_i(k) + 0$$

$$\mathbf{z}_v\left(k\right) = \mathbf{h}\left[\mathbf{x}_v\left(k\right), \mathbf{p}_i\left(k\right)\right]$$

$$\mathbf{p}_i\left(k\right) = \begin{bmatrix} x_i\left(k\right) \\ y_i\left(k\right) \end{bmatrix}$$

$$\mathbf{h}\left[\mathbf{x}_v\left(k\right), \mathbf{p}_i\left(k\right)\right] = \begin{bmatrix} \sqrt{\left(x_i\left(k\right) - x_v\left(k\right)\right)^2 + \left(y_i\left(k\right) - y_v\left(k\right)\right)^2} \\ \tan^{-1}\left(\frac{y_i(k) - y_v(k)}{x_i(k) - x_v(k)}\right) - \phi_v \end{bmatrix}$$

Dead-Reckoned

$$xo(k) = xo(k-1) \oplus u(k)$$
$$u(k) = \ominus xo(k-1) \oplus xo(k)$$

The input u(k) now comes from the background composition between old and new values from as an example the odometry sensor, instead from the kinematics or dynamic model of the robot. The observation model remains the same.

4 Tests and results

Two main tests were conducted with proposed architecture. In the first, two basic control behaviors were implemented in the USARSim environment with a P2DX robot. One was a well-known behavior-based switching algorithm and the second was a neuro-fuzzy based control algorithm. In this test, three simple indoor Virtual Robots environment rooms were chosen, and the robot path was recorded for booth controllers. Figure 6 show robot path in these tests.

The second test was related with the SLAM algorithm. A P2AT robot was released in an environment and its true and estimated pose in the environment were recorded for some steps. Implemented algorithm in this test was the EKF SLAM with dead reckoning, with optimized covariance parameters. Figure 7 present x, y and θ errors.

5 Conclusion and ongoing work

As shown in figures 6 and 7, the proposed controller has shown to be robust and reliable. In general lines the controlled robots have showed to work properly when dealing with the environment complexity. Indeed, the developed environment has demonstrated itself good enough in order to allow deeper research with mobile robots, stimulating deeper studies in a large number of areas by this work group.

Some of the ongoing works are:

- The improvement of the navigation algorithm
- The improvement of the SLAM algorithms
- The improvement of the visual detection

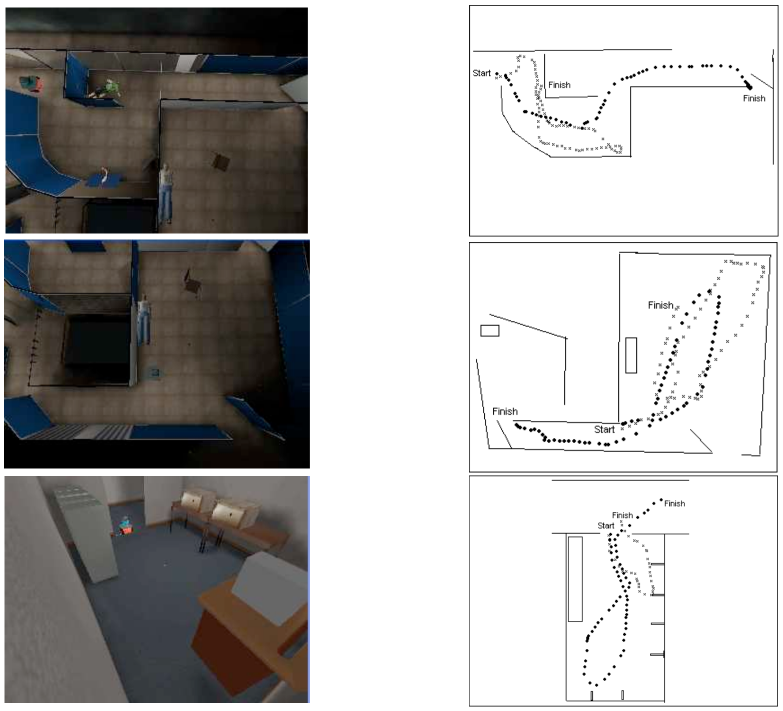


Fig. 6. Results. **Left:** Indoor scenes with real-world-like obstacles. P2DX shown pose is the initial robot pose in each test. **Right:** evolution of robot path. Filled points represent robot path with the neurofuzzy controller, while empty points show robot path with pure behavior-based controller.

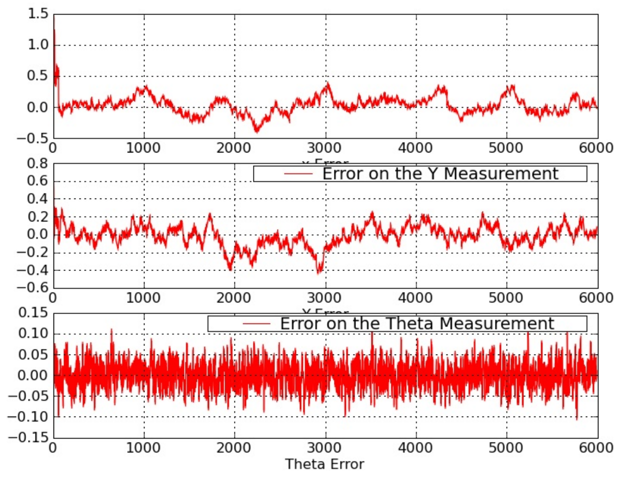


Fig. 7. Error of SLAM algorithm with steps evolution: a) error in X axis (in meters); b) error in Y axis (in meters); c) error in angle (in radians).

References

1. Borenstein, J., Everett, H.R., Feng, L.: Navigating Mobile Robots: Systems and Techniques. Addison-Wesley (1996)
2. Wang, J., Lewis, M., Hughes, S.: Validating USARsim for use in HRI research. Proceedings of the human factors and ergonomics society **49** (2005) 457–461
3. Colombini, E.L., Matsuura, J.P., Simões, A.S., Prado, G., Franchin, M., Lima, A.M., Neto, A., Souza, M., Oliveira, T.F., Guerra, W., Rosenberg, M., Ribeiro, M.: Brazil-vr team description paper. http://www.itandroids.com/Brazil-VR$_T DP.pdf$(2008)
4. Robotics, A.: Activmedia robotics website (2006) Available at: http://www.activrobots.com/. Accessed in: april-2006.
5. Simmons, R.: The curvaturevelocity method for local obstacle avoidance. (1996) 3375–3382
6. Rumelhart, D.E., Hinton, G.E., Williams, R.J.: Learning internal representations by error propagation. Parallel Distributed Processing. MIT Press (1986) 318–362
7. M., B.J., R, B.J.: Fuzzy logic guidance and obstacle avoidance algorithms for autonomous vehicle control. In: Procs of the Int Workshop on Int Autonomous Vehicles, Southampton, UK (1993) 41–52
8. Tunstel, E.: Mobile robot autonomy via hierarchical fuzzy behavior control. In: Procs of the 1st World Automation Congress, Montpellier, FR (1996) 837–842
9. Simões, A.S., Reali, A.H.C.: Classificação de laranjas baseada em padrões visuais. In: Simpósio Brasileiro de Automação Inteligente, Bauru (2003)
10. Simões, A.S.: Aprendizado não-supervisionado em redes neurais pulsadas de base radial. PhD thesis, São Paulo University, São Paulo (2006)
11. Newman, P.: Ekf based navigation and slam. SLAM Summer School 2006, Oxford (2006)

12. Akersson, B.M., Jorgensen, J.B., Poulsen, N.K., Jorgensen, S.B.: A tool for kalman filter tuning. In: Proceedings of the 17th European Symposium on Computer Aided Process Engineering, Elsevier (2007)
13. Kim, C., Sakthivel, R., Chung, W.K.: Unscentered fastslam: a robust and efficient solution to the slam problem. IEEE Transactions on robotics (2008)
14. Nuchter, C.A.: Slam tutorial. 6D SLAM. ECMR (2007)
15. Arleo, A., del R. Millán, J., Floreano, D.: Efficient learning of variable-resolution cognitive maps for autonomous indoor navigation. IEEE Transactions on Robotics and Automation **15** (1999) 991–1000
16. Bosse, M., Newman, P., Leonard, J., Teller, S.: Simultaneous localization and map building in large-scale cyclic environments using the atlas framework. The International Journal of Robotics Research (2004) 1113–1139