Towards Open Architectures for Mobile Robots: ZeeRO

Radu Bogdan Rusu[1,2], Radu Robotin[1], Gheorghe Lazea[1] and Cosmin Marcu[1]
[1]Robotics Research Group, Department of Automation, Technical University of Cluj-Napoca
26-28 Baritiu str., Cluj-Napoca, Romania
{radu.rusu; radu.robotin; gheorghe.lazea; cosmin.marcu}@aut.utcluj.ro
[2]Intelligent Autonomous Systems, Computer Science Department, Technische Universität München
Boltzmannstr. 3, Munich, Germany
rusu@cs.tum.edu

Abstract: **We report on ongoing research regarding the development of open mobile robot architectures. Preliminary results of our work show that a great degree of modularity and complexity in mobile robot design can be achieved, by simply using off-the-shelf available hardware components.**

A hardware robotic platform would not be nearly as efficient, if it wouldn't be sustained by an adequate software system. We are proposing the use and the development of the Player/Stage software project, of which we are active contributors, as the backbone of our software robotic platform.

This paper presents the experimental results regarding the development of the ZeeRO mobile robot, with an emphasis on its hardware and software architecture. We give insight on the structure of the main modules (navigation, sensing, processing), as well as the robot's interface with the Player server. Several tests with virtual environment navigation algorithms by means of simulation tools (Stage, Gazebo) have also been conducted.

A small section of the paper is dedicated to Javaclient, a Player/Stage client for the Java programming language, initially developed at the University of Southern California, and later moved to SourceForge.

Keywords: **mobile robot, sensor data fusion, modular robotic architecture, Player/Stage, Javaclient**

I. Introduction

Mobile robots represent a research subject in full ascension. Initially used in military projects or in the industry as Automated Guided Vehicles, they have now become one of the main attractions of the XXI century. The success of several projects developed by corporations such as Sony (Aibo, Qrio) or Honda (P3, Asimo) lead to a certain fact: some of the world's population is already ready to adopt cognitive mobile robotic systems in their homes. The prognosis of some field specialists is that by the end of this century, every house on the planet will have at least one cognitive robotic system with a humanoid appearance performing daily household chores.

It is obvious though, that in order to develop machine learning (artificial intelligence in general) algorithms, it is necessary for both the hardware and the software platforms to be able to ensure a high degree of efficiency, modularity and scalability. Some of these aspects have already been addressed, but there are still a lot of things that need to be done in this robotic standardization ongoing process.

This paper is presenting a novel way of designing and implementing a mobile robot architecture. The result of our research led to the creation of the ZeeRO mobile robot, built in the Robotics Research Lab, at the Faculty of Automation and Computer Science, Technical University of Cluj-Napoca, Romania. Some of the ideas used in the building process of this robot are quite new and innovative. The resulting architecture is making use of modern hardware technologies and state of the art software algorithms, thus enhancing the overall modularity and efficiency of the newly created robotic system.

The aim of the project was to design a low-cost, open, modular structure mobile robot, suitable for research, and easy to integrate with existing robots (Pioneer 2 and Pioneer 3) in cooperative applications.

ZeeRO uses the Player/Stage platform as the backbone of the software system, and integrates new high level functions and algorithms, making the mobile robot programmable in a variety of programming languages (Java, Lisp, C/C++, C#, Python, Ruby, etc).

The central processing unit uses a distributed architecture with a distinct separation between high level functions (master level) and low level operations (slave level).

The link with other mobile robots, but also the communication between ZeeRO and a PC is done using wireless Bluetooth, thus providing a higher degree of autonomy and flexibility to the system. The design of the sensor system allows further development of this project towards sensor data fusion and high level navigation algorithms based on intelligent control.

II. Related Work

There are several commercial companies that manufacture and sell mobile robot systems for research or domestic applications. Mobile robotic platforms such as Pioneer from ActivMedia or Kephera from CyberBotics, have already made it into several research labs and universities all around the world. Even newer systems, such as iRobot's domestic vacuuming robot, Roomba, have been adopted as a research platform by some. Given all this, one would think: why not use what's already out there and stop reinventing the wheel?

1-4244-0361-8/06/$20.00 ©2006 IEEE

There are certain things that must be taken into consideration though. First of all, the above mentioned products are:

- costly (in the rank of thousands of US $), especially for what they offer;
- closed, from an architectural point of view (ex. customizing a Pioneer robot isn't an easy task), thus hard to "hack" into;
- not modular enough for easy replacing or adding/removing sensors, changing sensor places, etc.

Although there have been people replacing the internal Pioneer boards for example, with better ones (Roboteq controllers for instance [2]), it doesn't really make sense to do that on a regular basis, from a financial point of view.

Our approach takes the above assumptions into consideration, and tries to solve them. That is why we propose that a mobile robot architecture should be:

- open and modular, in both senses: hardware and software, providing sufficient I/O ports and/or interfaces for upgrades or customizations, as well as clearly defined data structures for fast and easy software development;
- cheap, since funds for hardware purchases are never *enough* (very important especially if one considers research with several mobile platforms and not just one);
- layered, with distributed processing units, capable of handling concurrent operations, thus providing a high degree of robustness in case of failures;
- packed with as many different types of sensors as possible, good for researching sensor data fusion techniques (since no single type of sensor can give enough information about the environment).

Similar research has been conducted by our colleagues at the George Mason University, USA, on their FlockBots project, which started shortly after our project. While their approach looks similar to ours at first glance, there are several things that differentiate our projects. First of all, the FlockBots sensor system does not encapsulate ultrasonic sensors, which in our opinion are mandatory for a good navigation and localization algorithm. While there have been reports on using Infrared sensors alone for navigation, the information received from the Sharp IR GP2D02 sensors is very noisy and heavily inaccurate as a basis for a proper navigation system. Secondly, one of the biggest advantages of our software infrastructure is that, by making use of Player/Stage and its layered architecture, it's possible to write user-client programs in a variety of programming languages, on a variety of operating systems. The FlockBots lack this modular software system, so they interfere low-level programming routines with high-level behavior algorithms, which complicates the software development process. This issue is becoming quite obvious if one is working on developing intelligent navigation control routines.

While other groups are doing research in similar areas of mobile robot design (ex. the Marvin mobile robot from the Institute of Robotics and Intelligent Systems at ETH Zurich), their approach is totally different than ours (they use expensive microcontrollers and sensor boards), so a comparison with them will not be covered in this paper.

Considering the above statements, we believe that our project has an important role in the open architecture mobile robot design research area.

III. General Architecture

Figure 1 shows the general architecture of the ZeeRO mobile robot, with its main modules: navigation, sensors, processing and communication.

Communication is handled via Bluetooth by an unit of the central processing system. The sensor system is composed of infrared sensors (Sharp GP2D02), ultrasonic sensors (Devantech SRF08 and SRF10), a CMUcam2 video camera and a pyroelectric sensor (Eltec 442-3). We skip any in-depth explanation of the sensorial system, since they were already covered by other sources ([3], [5]).

The ZeeRO mobile robot is a two wheels differential drive vehicle. The two servo-motors are directly controlled by a component of the central processing unit (the Brainstem GP 1.0). Two additional servo-motors compose a pan-tilt unit for the CMUcam2 video camera, controlled directly by the CMUcam2's SX52 microcontroller (up to 5 servomotors can be controlled).

The robot motion control parameters are x and y (the position of the center of the wheel axis), and θ (the orientation of the mobile robot with respect to the horizontal X axis - see figure 2). These parameters are determined as a function of time, using equation (1) and (2) respectively.

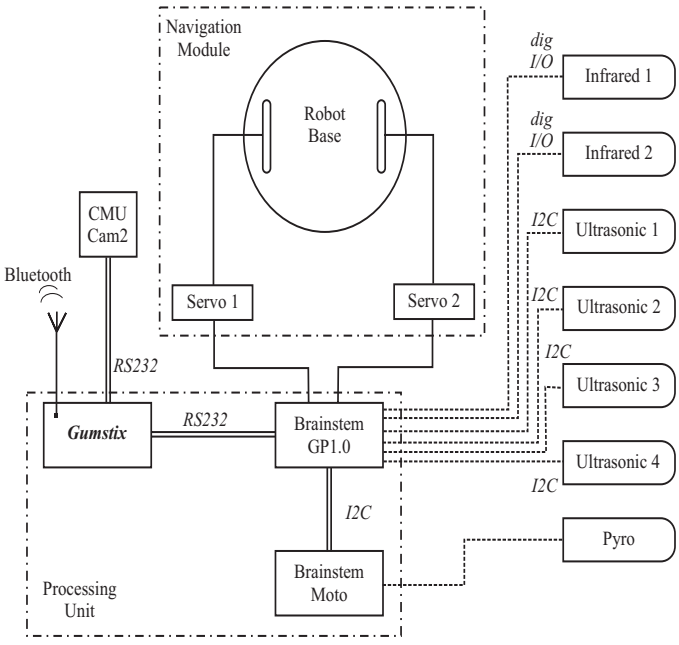

Figure 1. The ZeeRO architecture

The following equations are describing the mobile robot's position at time t. More in-depth information can be found in [3].

$$\begin{cases} x(t) = \frac{1}{2}\int_0^t \left[v_d(t) + v_s(t)\right] \cdot \cos\left(\theta(t)\right) \cdot dt \\ y(t) = \frac{1}{2}\int_0^t \left[v_d(t) + v_s(t)\right] \cdot \sin\left(\theta(t)\right) \cdot dt \\ \theta(t) = \frac{1}{l}\int_0^t \left[v_d(t) - v_s(t)\right] \cdot dt \end{cases} \quad (1)$$

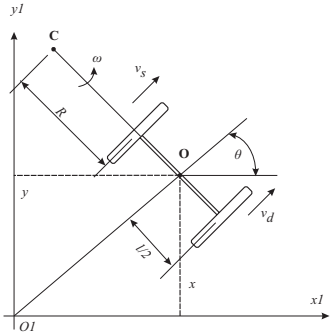

Figure 2. Motion control parameters

The point *C* in figure 2 represents the center of curvature, which, for a differential driven system is the point around which the robot rotates. v_d and v_s are the velocities of the right and left wheel, given by:

$$\begin{cases} v_d = \omega \cdot \left(R + \frac{l}{2}\right) \\ v_s = \omega \cdot \left(R - 0\frac{l}{2}\right) \end{cases} \quad (2)$$

where *R* is the curvature radius and *l* is the distance between wheels.

IV. Processing Unit

Figure 3 shows the connections between the main elements of the central processing unit. The Waysmall Gumstix is an ubiquitous device, powered by Intel's PXA255 XScale processor, running at 400Mhz. In our architecture, it provides the high level processing layer while Brainstem GP1.0 and Brainstem Moto1.0 represent the low level one.

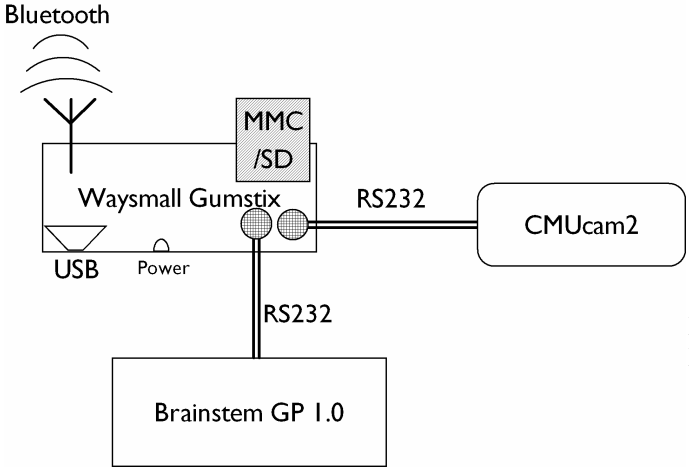

Figure 3. Central processing unit connections

A big advantage on using Gumstix is that it is powered by the Linux operating system, making it easy to deploy and develop free open source projects.

The Waysmall offers two RS232 (miniDIN-8) interfaces, as well as an USB client one. We used the serial interfaces to connect the Brainstem network (see figure 5) and the CMUcam2 video camera.

Besides the 4MB of flash memory on the Gumstix, the user has the possibility to add additional storage space through the Waysmall SD/MMC card reader (figure 4).

Gumstix is also equipped with wireless communication capabilities, kudos to the Infineon ROK104001 Bluetooth chipset.

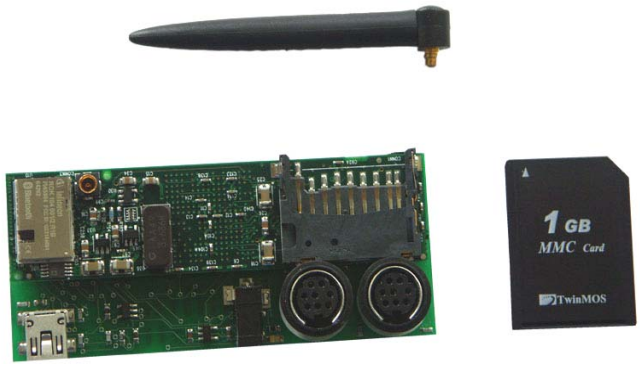

Figure 4. The Gumstix Waysmall

The Brainstem GP1.0 is built around a 40MHz RISC processor. It provides 5 10bit analog/digital input channels, 5 digital I/O lines, an I2C interface, 4 servomotor outputs, an IR port, a RS232 serial port, and a power supply connection.

The Brainstem Moto1.0 is also built around a 40MHz RISC processor. It provides 2 PWM controlled channels, 1 analog /digital (10bit) input channel, 1 digital I/O channel, an I2C interface, a RS232 serial port, and a H-bridge connector.

Figure 5. The Brainstem network

For the ZeeRO mobile robot, the Brainstem network comprises one GP module and one Moto module, this approach providing several advantages:

- support for communication between the modules;
- only one serial connector is used to interface the network with the higher level processing units;

- only one power supply is used by the network, the power for the Moto module being supplied via the GP module.

V. The Player Architecture

The Player server is a component of the Player/Stage project[5]. Developed initially at the University of Southern California, USA, in the Robotics Research Laboratory, the Player/Stage project is now sustained by an international community of researchers worldwide. Since 2001 it is hosted on SourceForge. Player/Stage has two main components:

- A server component (Player), which can be looked upon as a collection of drivers (for mobile robots, sensors and actuators) and interfaces (for accessing the information provided by the low-level drivers in a standardized manner);
- A simulator component (initially Stage, for 2D simulations, and later Gazebo for 3D).

The Player/Stage system is open source software, distributed under a GPL license. The project compiles and runs on pretty much any POSIX platform, including embedded systems, such as Gumstix.

Player uses a client-server model based on the TCP (or UDP) protocol, thus enabling the client applications to be implemented using almost any programming language. Furthermore, the client may run on any computer that is connected in a network containing the Player server connected to the mobile robot. Current client library implementations include languages such as: Ada, Java, Lisp, Octave, Python, Ruby, Scheme, and of course, C and C++.

The developers of the Player project [4] have made a clear separation between the "programming interface" and the control structure. The Player server is implemented in C++ using standard POSIX *pthread* functions to provide support for multithreading. Each client uses a TCP/UDP socket in order to connect to the Player server. If both the client and server are on the same computer, the connection will be a loop-back. On the other hand, Player connects to physical devices, most of the time using RS232 (see figure 7).

As figure 7 shows, Player receives information from the mobile robot (which can be a real mobile robot or a simulated one) usually through the serial port. This data is processed by the Player driver and then made available to the client. This approach is flexible, new devices may be added, either as a module in the Player server or as a dynamic linked library that Player will load before the client connects.

VI. Javaclient

Together with ZeeRO, we developed an advanced Java client library for Player/Stage, in an international cooperation project together with Maxim Batalin, PhD, from University of Southern California, USA. A clear result of this cooperation process lead to the usage of Javaclient by major research groups and AI/robotics institutions all around the world (the number is still increasing), thus enabling our users the possibility to use an advanced multi-threaded Java software system to develop applications for Player/Stage. Since 2005, the Javaclient project is hosted on SourceForge.

Javaclient implements all the interfaces described in the player manual, but has some additions of its own (virtual position and heading controllers were implemented using PID). Documentation, examples, the complete source code of the project as well as user mailing list are available on the Javaclient SourceForge web site[7].

The usage of Javaclient is straightforward: the user instantiates an object of type PlayerClient, then requests access to the desired interface(s), and then, using the Player messaging system, communicates with the underlying hardware drivers. The user doesn't need to know or care anything about the low-level routines for interfacing the hardware components. Three types of messages are currently supported: `data`, `command` and `request/reply`. Data messages are sent from Player to the client and contain useful sensor measurements or any other type of data that a physical or virtual device could provide. Commands are sent from the client to the Player server (ex. position commands or velocity commands for motors), and requests/replies are queries sent by the client to the server followed by answers with either an error status or the required data from the server to the client. A trivial pseudo-example on Javaclient usage follows:

```
import javaclient.*;
…
// Connect to the Player server and request access to the robot's
// ultrasonic sensors through the SonarInterface interface
PlayerClient robot = new PlayerClient ("localhost", 6665);
SonarInterface    si = robot.requestInterfaceSonar (0, 'a');
// Run Javaclient in multi-threaded mode
robot.runThreaded (-1, -1);
while (true) {
        // Get the sensor values and display them on screen
        int[] sonars = si.getRanges ();
        for (int i = 0; i < si.getSamplesCount (); i++)
                System.out.println (sonars[i]);
}
```

As can be seen from the above example, connecting to a Player server and requesting data from an ultrasonic array of sensors requires just a couple of lines of code.

The code repository contains three important packages: javaclient, javaclient.structures and javaclients.extra. More information about Javaclient can be found at [7].

VII. Experimental Results

The first prototype of the newly designed robotic architecture was finished in May 2005. As previously presented, the ZeeRO mobile robot has a variety of sensing capabilities, therefore allowing a wide range of navigation algorithms to be developed.

In order to be accessible from Player/Stage, a Player driver had to be developed, and an existing one modified. The resulted combination provides the data through the following standardized Player interfaces:

- `position2d` for the robot's servomotors (velocity/position commands and odometry data readings are supported);
- `sonar` for the robot's ultrasonic sensor array (sonar data values readings as well as geometry coordinates supported for 0..4);
- `ir` for the robot's infrared sensor array (infrared data values readings as well as geometry coordinates supported for 0..1);
- `aio` for the robot's pyroelectric sensor (data values readings supported);
- `camera`, `blobfinder` and `ptz` for the cmucam2 (the existing driver had to be modified to add support for getting video images through the camera interface, as well as to support changing the tracking color on the fly, while the user software is running).

An example of the resulted architecture for a user application using Javaclient is presented in figure 6.

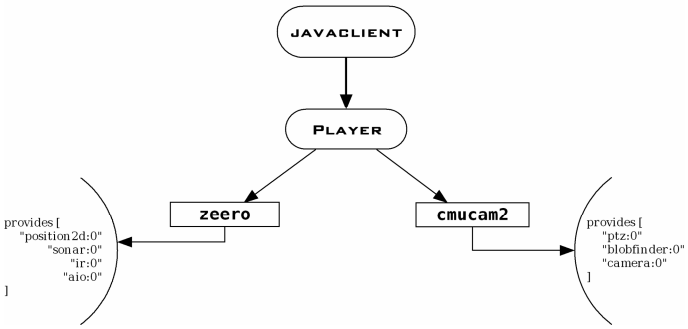

Figure 6. Client and Player driver diagram

The two drivers, `cmucam2` and `zeero` are pushing data through the above mentioned interfaces. A player configuration file for the ZeeRO mobile robot could look like the following:

```
driver (    name "zeero"
            plugin "libzeerodriver.so"
            provides ["position2d:0" "sonar:0" "ir:0" "aio:0"]
            port "ttyUSB0"
            speed 38400 )
driver (    name "cmucam2"
            devicepath "/dev/ttyS2"
            provides ["blobfinder:0" "ptz:0" "camera:0"]
            num_blobs 1
            color0 [105 155 0 41 0 41] )
```

Several algorithms and tests were conducted with the ZeeRO mobile robot in our laboratory (navigation with obstacle detection and collision avoidance, human detection via the pyroelectric sensor, multiple color blobs tracking, and others). While some preliminary results have already been presented in [8], research on navigation algorithms is still an ongoing process.

Since we only had one initial prototype developed, we felt the need to create a virtual model of the ZeeRO robot, and try our in-house built cooperation algorithms between several robots in the simulator. Some experiments were already presented in [1], by making use of a multi-agent multi-robot society concept and the Stage 2D simulator, but work on switching to the Gazebo 3D simulator is currently underway.

VIII. Conclusion and Future Work

The contribution of our work is the development of an open-architecture mobile robot, as well as the contribution of several modules and patches, together with an advanced Java software client to the Player/Stage international robotic community.

Inspired by the Open Source community, we released all the pictures, drawings/schematics, and software source codes of our mobile robot on the Internet[10]. By providing this service to the international robotic community, we hope to attract other research groups in doing the same and helping enlarge the existing knowledge base in this area of research.

Further work on this project will be focused on the extension of the sensorial system. The latest advancements in sensor miniaturization, have given birth to ultra-tiny and affordable laser sensors such as the Hokuyo URG units. Mounting a laser unit on ZeeRO could expand the current research opportunities to other areas of mobile robotics such as map building or SLAM (Simultaneous Localization and Mapping). Overall, a much more precise collision avoidance module can be built and used, instead of the current one who just relies on information from the ultrasonic and infrared sensors. We are definitely planning to use these type of sensors in the next iteration of our ZeeRO mobile robot.

Several other things need to be tweaked, especially in the locomotion system (navigation module). We learnt the hard way, that cheap is not always good in terms of the servomotors' performances. Surely, the next iteration will have better motors.

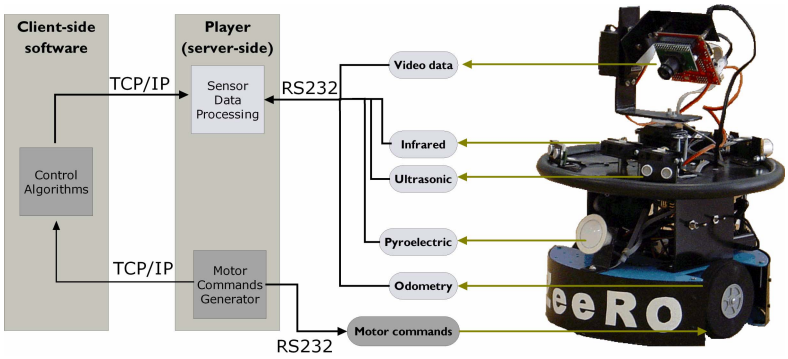

Figure 7. The ZeeRO mobile robot and its connection to the Player/Stage platform

Figure 8. An overview of the development process of ZeeRO. First to final version from left to right.

Our previous results on navigation in static and dynamic environments (using custom D*-like previously developed algorithms) [11] lead us to believe that we can apply the same techniques for swarms of several ZeeRO-like mobile robots, once we build them. Until then, we will continue our multi-robot research in the Player/Stage family of simulators.

ACKNOWLEDGEMENTS

We would like to thank our students Sime Romulus, Sime Remus and Mihaela Chira for their help with this project. Their diploma theses are also a good source of information about the results of the project. ZeeRO has its own web site, available at *http://www.robotux.info/zeero*.

REFERENCES

[1] Radu Bogdan Rusu, Liviu Miclea, and Szilard Enyedi. Robotux – a multi-agent robot based security system. In *Proceedings of IEEE-TTTC Automation, Quality & Testing, Robotics International Conference 2004, Cluj-Napoca, Romania, May 13-15, 2004, AQTRJ.*

[2] Freek Stulp, Michael Isik, and Michael Beetz. Implicit Coordination in Robotic Teams using Learned Prediction Models. In *Accepted for the IEEE International Conference on Robotics and Automation (ICRA)*, 2006.

[3] Radu Bogdan Rusu. Modern architectures for mobile robots: Javaclient and ZeeRO. *Dissertation thesis for Advanced Postgraduate Studies*, June 2005.

[4] B. P. Gerkey, R. T. Vaughan, and A. Howard. The Player/Stage project: Tools for Multi-robot and Distributed Sensor Systems. In *Proceedings of the International Conference on Advanced Robotics (ICAR)*, Coimbra, pp 317 – 323, 2003.

[5] Sime Remus. A sensorial system for the ZeeRO mobile robot. *Diploma thesis*, June 2005.

[6] Radu Bogdan Rusu. On Data Fusion Methods using Neural Networks, from a Practical Implementation POV*. *Unpublished paper series, Robotux.info web site*, May 2005.

[7] Javaclient for Player/Stage. *http://java-player.sf.net*

[8] Mihaela Chira. Behavior-based algorithms for the ZeeRO mobile robot. *Diploma thesis*, June 2005.

[9] J. Jones. *Robot Programming: a Practical Guide to Behavior Based Robotics*, Mc Graw-Hill, 2000

[10] The ZeeRO mobile robot. *http://www.robotux.info/zeero*

[11] A. Stentz. The Focussed D* Algorithm for Real-Time Replanning, In *Proceedings of the International Joint Conference on Artificial Intelligence (IJCAI)*, 1995.