

# 3D Mapping: testing algorithms and discovering new ideas with USARSim

Paloma de la Puente, Alberto Valero, Diego Rodriguez-Losada

**Abstract**—It is usually not very convenient to use real robots for software development, testing and evaluation. Having access to a good simulator allows researchers to recreate special experimental conditions and gain increased flexibility and reliability, saving a lot of time. In the 3D mapping context specific issues arise. Here we describe and compare our experiences in this area with a real robot and with the Unified System for Automation and Robot Simulation (USARSim).

## I. INTRODUCTION

Most of the problems that direct experiments with real robots pose are well known and recognized. Set up of equipment and hardware configuration changes are time consuming and not always possible. Environmental conditions may not be adequate at every moment, neither is it easy to arrange real scenarios to be like we would want them to. In addition, running out of battery in the middle of an experiment may require to come back to a starting position perhaps far away located. These are just a few examples, not to talk about the risk it implies testing new software on a mobile, probably expensive, platform. Hence the important role simulators play in the development and validation of algorithms and techniques in the research field [1].

In contrast with some existing robotic simulators which only offer 2D virtual worlds, like MobileSim [2], by ActiveMedia Robotics [3], or Stage, belonging to the Player/Stage/Gazebo project [4], USARSim [5] provides a wide variety of 3D highly realistic models of different environments, more than 23 robotic commercial platforms and more than 15 sensors [6]. The available world models can be edited to generate new desired ones. USARSim also approximates kinematics and dynamics with a good precision. Furthermore, the user can easily see that perceptual fidelity for a better human-robot interaction (HRI) is one of the most outstanding features that USARSim presents.

Recently, we have been working with USARSim with two main purposes. At first, we wanted to collect data to obtain 3D point clouds associated to odometry data as the robot operated in a stop-scan-go manner. Even though we do have a 3D mapper real robot working, for the afore-mentioned reasons this allowed us to speed up our 3D data gathering processes and increment the number and variety of data sets we could use to test and tune our data processing and mapping algorithms [7]. Afterwards, we got involved in the



Fig. 1. Left:3D data acquisition system, mounted on our robot Nemo.

RoboCup’09 Search and Rescue Virtual Robots Competition [8], [9]. For the integration of the 3D mapping components [10] and aiming to get a suitable functional solution according to the competition’s goals and conditions, different modifications and adaptations were incorporated and a new robot configuration was designed.

The paper is organized as follows. Section 2 is a description of our P3AT robot, Nemo, and its 3D laser sensor, along with a brief report on how we operate with it. Section 3 is about our simulated Nemo P2AT, emphasizing its similarities and differences with the real robot. Section 4 presents ATRVJr3D, the final robot configuration we chose for the competition. In Section 5 we outline some of the techniques we developed for the competition. Section 6 presents some experiments and analyzes the performance of our models. The paper finishes with our conclusions.

## II. THE REAL ROBOT, NEMO

Our real robot is a Pioneer 3AT by MobileRobots.Inc [11], we call it Nemo (Fig. 1). At its front we have mounted a SICK LMS200 laser sensor on top of a servo pan-tilt unit. This device’s model is PowerCube Wrist 070 (PW70), by Amtec Robotics [12]. The wrist requires 24V cc, which are obtained by means of a dc/dc converter from 12 to 24V. Both the laser and the wrist are connected to a mini laptop onboard the robot with USB to RS-232/422/485 adapters. A data server processes the data and sends synchronized updated information about odometry, PW70 and laser measurements at clients’ cyclical requests. The port’s baud rate for the laser scanner is set at 500 kb so as to gain velocity and allow for a good precision in the synchronization with the PW70. This is of utmost importance to avoid distortions when applying the relative transformations to calculate each point’s cartesian 3D coordinates.

This work was partially funded by the Spanish Ministry of Science and Innovation, DPI2007-66846- c02-01

P. de la Puente, A. Valero and D. Rodriguez-Losada are with the Intelligent Control Group, Universidad Politecnica de Madrid, 28006 Madrid, Spain

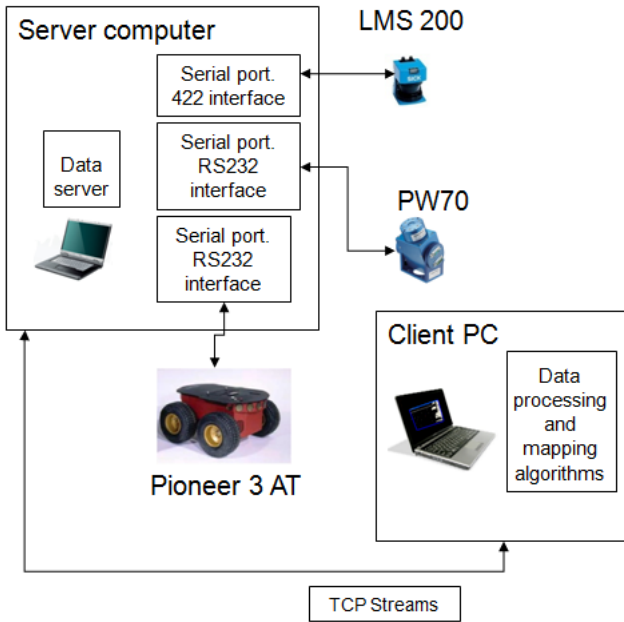


Fig. 2. Distribution schema for our work with the real robot

Fig. 2 depicts the general configuration schema that we use.

The pan/tilt unit presents a robust and compact design, yet being light enough to permit a good turning speed (a maximum of  $248^\circ/\text{sec}$  for the tilt angle). The movement is smooth and quiet. The tilt angle is in the  $\pm 120^\circ$  range, with a repeatability of  $\pm 0.02^\circ$ . The PowerCube device can receive position, speed and torque commands for each joint, and can provide feedback about the pan and tilt angles at any given moment.

The data collection procedure is implemented as a finite state machine. As the scan order is given, a command to move the wrist up to its final superior position is sent. When the error between that value and the actual tilt value given by the reading from the wrist sensor is below a threshold, the 3D data acquisition process is ready to begin. The wrist is commanded to go down and the laser measurements associated to the tilt value at every cycle are stored in both memory and text files. When the difference between the tilt value and the due final tilt value downwards looking is below the threshold, a command to get the wrist up again is sent and the data are stored in reverse order, to keep the point cloud structured for further processing. As mentioned before, this process is controlled by a scan parameter that sets the frequency with which the point clouds are obtained.

At first, we teleoperated the robot and made it stop before manually demanding to take a 3D scan, collecting data only when the wrist was going down. Owing to the fact that for the competition we required continuous updating to achieve real 3D obstacle avoidance capabilities, we decided to make it capture data when coming up as well.

### III. THE SIMULATED ROBOT NEMO P2AT

Building a robot able to take 3D scans in USARSim is not a complicated task. There are 2 straightforward possibilities:

- The first one is to use the 3D range scanner sensor. USARSim includes with its stable release a 3D range scanner that returns 3D point clouds within a range that can be configured by the sensor parameters.
- The second one is to use the already available robot Kurt3D. Kurt3D is a robot which has a SICK range scanner mounted on a device that can be tilted and it is included in the latest versions of USARSim. Its tilting device accepts position commands, so that you can bring the laser to a desired angle. The device does not accept speed, nor torque, commands, and it does not provide the tilt angle either. Kurt3D's scanner does not have pan motion.

However, none of these two solutions could reproduce properly our real robot's behavior.

-The 3D range scanner is significantly different from our real sensor. Using it, one cannot control the scan's limits. We could want, for example, to initiate our scan 45 degrees under the zenith and end it at a tilt value of 10 degrees above it, which is not possible with this kind of sensor. Furthermore, it does not let us control the tilt angle interval between two consecutive 2D scans.

-Kurt3D was a close solution albeit it also presents some drawbacks. The most relevant one is the fact that it cannot provide feedback about the value of the tilt angle. This can be solved in two ways. The first option is to give a small angle increment and wait long enough in order to make sure that the commanded angle has been reached. An alternative may be to mount an additional sensor, like an INS (to get the angle directly), or an encoder. Both solutions are relatively easy to implement in USARSim. Another problem this configuration presents is the fact that you cannot control the tilt device using speed or torque commands.

In our first experiments to gather 3D data from the simulator we used Kurt3D's tilt unit and implemented a step by step motor system as described in [7]. Our simulated Nemo is shown in Fig. 3. Its basis is a robotic P2AT standard platform. We added Kurt3D's *scannersides* sensor to the USARBot.NemoP2AT model we had created and set it as the parent for the SICK LMS laser. The main modifications introduced to the USARBot.ini file are the following.

```
JointParts=(PartName="ScannerSides",
PartClass=class'USARModels.Kurt3DScannerSides',
DrawScale3D=(X=1.0,Y=0.4,Z=1.0),
bSteeringLocked=true,bSuspensionLocked=true,
BrakeTorque=100.0,Parent="",
JointClass=class'KCarWheelJoint',
ParentPos=(X=0.16,Y=0.0095,Z=-0.16),
ParentAxis=(Z=1.0), ParentAxis2=(Y=1.0),
SelfPos=(Z=0.0), SelfAxis=(Z=1.0),
SelfAxis2=(Y=1.0))
Sensors=(ItemClass=class'USARModels.SICKLMS',
ItemName="Scanner1", Parent="ScannerSides",
Position=(X=0,Y=-0.0095,Z=-0.11),
Direction=(X=0,Y=0,Z=0))
```



Fig. 3. Our simulated Nemo P2AT

The corresponding NemoP2AT.uc file to create the class was based upon the existing one for the standard P2AT.

After adjusting the 3D laser height parameters, this simulated robot performed well and let us obtain 3D point clouds from several different environments. In Section 6 more details about parameters' values and results will be given.

#### IV. THE SIMULATED ROBOT ATRVJR3D

The configuration described in the previous section was fine to acquire 3D data in a stop-scan-go manner, but that system was too slow to work in more realistic conditions. We added an encoder to avoid needing to wait every time we sent a command and be able to move the *scannersides* all at once to its final position up or downwards looking. However, as the device's speed could not be regulated, the readings did not arrive on time and the error was much larger; it was difficult to configure the settings to obtain consistent data.

For these reasons, we decided to build a pan/tilt device emulating the real PowerCube070 wrist. In USARSim this is quite easy to do thanks to the mission packages. We built a laserpan tilt mission package which consisted of two rotational joints. These joints can be controlled independently with angle, speed and torque, and they provide feedback about their position. A similar example of a mission package is the camerapan tilt package, distributed with USARSim. Once the SICK range scanner was mounted on top of such a device we were able to acquire 3D data in pretty much the same way as with our real robot.

For the RoboCup 2009 competition, we mounted this 3D device on an ATRVJr platform; we called our robot ATRVJr3D. The reason why we did so is simple, a P3AT cannot afford to carry two SICK range scanners without compromising the dynamics of the platform. We wanted to have both a 2D and a 3D range scanner on the same robot, the 2D one for 2D SLAM and the 3D one to obtain the point clouds that would be processed afterwards. We opted to use a SICK laser for the 3D data acquisition rather than an Hokuyo laser device as proposed in our Team Description Paper [10] mainly to cover a wider distance range and to have less noise. We also took into consideration the fact that



Fig. 4. Our simulated ATRVJr3D

Hokuyo lasers scan a wider field of view and therefore make ray tracing in USARSim get notably heavier.

With the P3AT, equipped with just one SICK as described in Section 2, we acquire the 3D scans with the robot still, for the robot's position cannot be accurately corrected in the meantime, while the laser has not yet finished the acquisition of a complete 3d point cloud. With the simulated ATRVJr3D, as the robot can be localized using the fixed SICK while the other laser may be still collecting data, we can continuously acquire 3D data, stopping the robot only when requiring enhanced precision. The simulated ATRVJr3D is shown in Fig. 4.

For this robot we created the USARBot.ATRJVr3D class and added the following lines to its definition in the USARBot.ini file so that the new sensor be properly positioned, on top of the robot without colliding or seeing the camera and the other laser scanner.

```
MisPkgs=(PkgName="LaserPanTilt",
Location=(X=0.16,Y=0.0095,Z=-0.25),
PkgClass=Class'USARMisPkg.LaserPanTilt')
Sensors=(ItemClass=class'USARModels.SICKLMSr',
ItemName="Scanner3D", Parent="LaserPanTilt.Link2",
Position=(X=0,Y=-0.0095,Z=-0.06),
Direction=(X=0,Y=0,Z=0 ))
```

The SICKLMSr is a normal SICK LMS laser except for the fact that it is drawn downwards looking in the simulator (Fig. 4 must not lead the reader to confusion, it was taken before the last changes only for esthetic purposes). We used this class during the competition because we could not add yet another laser model and we wanted to change the maximum range in order to reduce the simulator's burden.

#### V. NEW TECHNIQUES INTRODUCED FOR THE COMPETITION

The major novelty we introduced for the competition was the creation of a ground map to make sure that the robot would not fall into holes nor collide with low obstacles when in autonomous operation. To do so, we checked the 3D points obtained from the data provided by the tilting laser. The points whose  $z$  coordinate's absolute value was below a threshold were converted to 2D points in the map image's reference frame and considered as belonging to an area the robot could safely traverse. We established several criteria to overwrite the colors in the 2D grid map generated with the

data coming from the horizontal laser. Our list of colors is the following:

- Black: obstacles in the map generated with the 2D data
- White: free areas in the map generated with the 2D data
- Blue: unexplored areas in the map generated with the 2D data
- Gray: obstacles detected by the 3D laser
- Green: solid ground free of holes and 3D obstacles (traversable areas)

The three first colors were used as defined in the rules for the competition. We added the other two colors to provide more information for the operator and to exclusively use the green areas for safe navigation.

At first, the ground map was not updated until a whole 3D point cloud was acquired, and green was never allowed to overwrite gray. This method overloaded the system a bit, and whenever a 3D scan was completed an important delay prevented the robot's pose from being properly updated. So, we decided to update the ground map periodically, with a relatively small period, only using the data that had been obtained within the last cycle.

Not letting green substitute gray was important to avoid removing obstacles situated at intermediate heights when the wrist was driving the laser down. However, we did want to remove 3D obstacles which were no longer there: dynamic objects (other robots), nonexisting small obstacles generated by noisy measurements . . . . We also wanted to eliminate some ugly lines that cropped up from time to time as a result of suddenly stopping the robot, as severe decelerations made it incline.

Our choice was to allow green overwrite gray only when the 3D laser was capturing data towards its upper position. This way, real 3D obstacles detected below the robot's height were not permanently there but as soon as the 3D laser saw them again when coming up, they were restituted so that reactive control could avoid them.

When the robot was spawned to operate autonomously right from the beginning of a mission it would not move, since the wrist does not get the laser to capture data below it. The same thing could happen as well when the robot explored new narrow areas appearing from behind a corner, for example. To solve that problem a fixed size green square was created around the robot's pose on the assumption that there would not be a hole in the closest surroundings, so that the movement towards the chosen frontiers for the autonomous exploration could be initiated. Fig. 5 shows some examples of ground maps generated like this.

The areas explored by the 3D laser sensor were sometimes not continuous due to the discrete distribution of the laser's beams, whose effect is most noticeable when the, normally slightly tilted, laser scans further obstacles. To reduce this undesirable result which did not always let the robot advance we adopted two measures.

The first one was rather simple and came to not only coloring green the pixels corresponding to a single 3D point considered an obstacle but also those in a window around that pixel in the map image. This worked quite well, as we

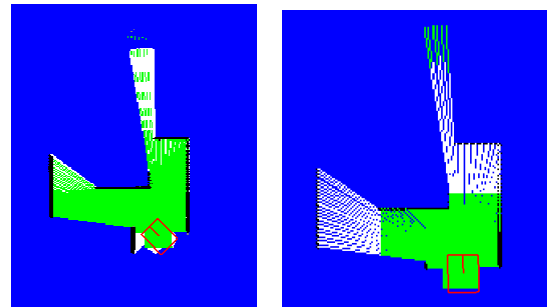


Fig. 5. Creation of a green square around the robot's pose so that it can start autonomous exploration. Left: small square. Right: larger square

kept on making sure that these pixels could not erase gray or black obstacles. Still, there remained some small white holes inside clearly green safe areas and this made the robot's motion slower.

To get green areas even, we decided to apply a *closing* (dilate + erode) morphological operation to the map image. What we did in the end was to iteratively apply a *floodfilling* algorithm to every white pixel in the map image, filling the area they belonged to with a different color. Afterwards, the new colored areas' size was checked to avoid removing real holes from the ground map; small holes were filled with green while bigger holes were turned white again. Fig. 6 is one such map image before classifying and recoloring the detected holes.

For the implementation of the algorithm we used the OpenCV libraries [13]. All our software is composed of a collection of self-developed and open-source libraries and modules, within the OpenRDK framework [14].

When there were a lot of white points inside green areas (due to excessive noise, in certain world models...) the execution of the *closing* module took too long and the whole system was affected. Therefore, the application of the *floodfilling* algorithm was restricted to a smaller region of the image containing the robot's pose and this technique was performed only when significant changes in the robot's pose were appreciated. Moreover, we introduced a limit in the number of holes so that when there were too many small white points close to the robot some of them were left white for subsequent cycles. All this yielded an overall good performance of the system.

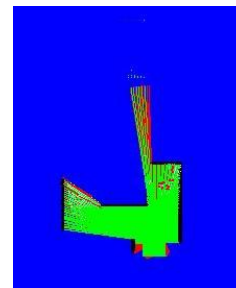


Fig. 6. Map image after applying a floodfilling algorithm with white pixels as seeds

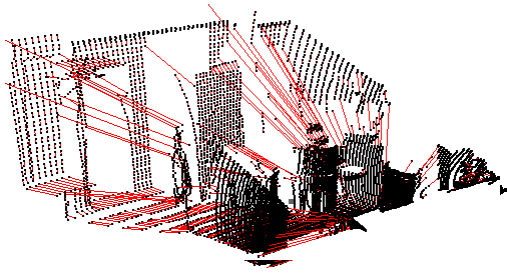


Fig. 7. 3D point cloud obtained with our real scanner



Fig. 8. Segmentation image obtained from the 3D scan in Fig. 7

## VI. EXPERIMENTS AND RESULTS

USARSim has served us as a great tool for development and experimentation. As commented in the introduction, at first we used it as a means to obtain new data to validate our already developed 3D mapping algorithms [7]. Our main concerns in the initial stage dealt with the resolution of the 3D point cloud. To capture the 3D data with the real robot, the laser is tilted from  $-25$  degrees (upwards looking) to  $30$  degrees (downwards looking) at a constant speed of  $0.05$  rad/s, which typically results in a  $70 \times 181$  matrix of measurements. Fig. 7 and Fig. 8 show a point cloud gathered with our real system and the segmentation image obtained from it by applying the algorithms described in [15], [7], of which a brief account follows.

The residuals from fitting each points neighborhood to a plane are used as measurements to generate a range image from the structured 3D data acquired by the laser scanner. A second order *closing* (dilate + erode) morphological operation is applied to the image so as to help edges be better defined, without breaking improperly. To eliminate false borders caused by the presence of noise, the image is binarized so that only important edges are left. Once this has been accomplished, a floodfilling algorithm is used to assign the same color to the pixels inside each large enough region enclosed by the remaining borders. This way, taking advantage of the ordered nature of the data, real-time capabilities are achieved. A final check comparing local normal vectors is performed to make sure different surfaces are correctly separated even if some edges in the image are not satisfactorily closed. This allows for further robustness without being significantly more time consuming.

When working with the simulated Nemo P2AT, at first we set the tilt limits at  $23$  degrees and  $-30$  degrees (notice that the tilt's sign is inverted in relation with the real PowerCube wrist) and got a laser 2D scan at every  $0.25$  degrees in order to obtain continuous data from the floor and the ceiling.



Fig. 9. Segmentation image with too many rows and not covering lower parts

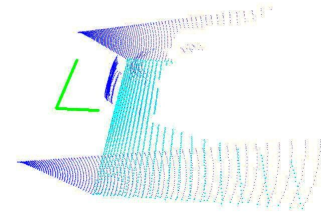


Fig. 10. A point cloud captured with the simulated Nemo P2AT in the DM.Mapping 2006 USARSim world. View from above, the floor is depicted in cyan. Only the rear part of the car can be seen from the robot's current pose, marked in green, with the robot facing the shortest axis

This, however, led to having too many rows in the 3D data structure and the images created did not allow for an easy recognition of the objects in the scene. Furthermore, the tilt variation proved not be enough. An example of this is shown in Fig. 9, obtained from a 3D scan got from the *PlayerStart* pose in DM.Mapping.

With the final configuration, using the mission package and commanding the wrist with a final angle, the results improved significantly. Fig. 10 shows a 3D point cloud obtained from the same pose and Fig. 11 is the segmentation image corresponding to that point cloud. The threshold referred to in Section 2 is slightly smaller than the one we used when operating with the step by step motor system, for a more precise final position of the laser is desired. Apart from that, the other parameters are kept the same. A useful comparison to perform may be to acquire some data sets with both the real robot and its simulated model and analyze the discrepancies in the number of 2D scans (rows in the corresponding image) integrating the 3D scans of both categories.

The uncountable series of experiments we conducted for



Fig. 11. Segmentation image obtained with the final configuration obtained from the same pose as the one shown in Fig. 9. The car can be better identified now

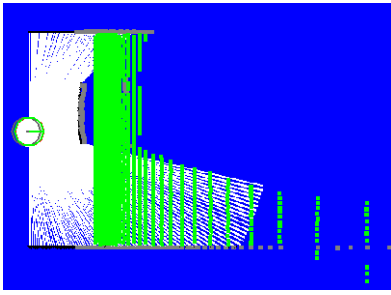


Fig. 12. Ground map corresponding to the point cloud in Fig. 10

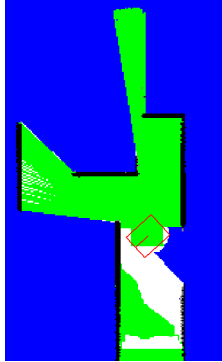


Fig. 13. Ground map corresponding to a 3D scan acquired at *robot1* pose in DM.Mapping

the competition were not reproduced with the real robot, for two main reasons: lack of time and the fact that our robot is not equipped with two laser range finders.

The segmentation of the simulated data worked quite well with no changes in the parameter's values with respect to those used with the real data. However, if the threshold used for the image binarization is smaller, somewhat better results are obtained if the laser's noise presents its standard value for the SICK LMS200.

Tests with USARSim let us decide on the values of the parameters for the construction of the ground map so that in the end it was quite effective. Fig. 12 and Fig. 13 show some images of such maps.

## VII. CONCLUSIONS

In this paper we have presented the robot models we have used to acquire 3D data from USARSim environments along with some comparisons regarding our work with the real robot and some of the innovations we have introduced for the RoboCup'09 Search and Rescue Virtual Robots Competition. USARSim has undoubtedly been a very valuable tool for acquiring a wide variety of data to validate our algorithms and it has saved us a lot of time, making things convenient and realistic.

## VIII. ACKNOWLEDGMENTS

The authors gratefully acknowledge the Spanish Ministry of Science and Innovation (DPI2007-66846- c02-01), the Comunidad de Madrid (Consejeria de Educacion) and the European Social Fund (ESF) for supporting this work.

## REFERENCES

- [1] (2008) Iros'08: Workshop on robot simulators: available software, scientific applications and future trends. [Online]. Available: <http://www.robot.uji.es/research/events/iros08>
- [2] Mobilesim. [Online]. Available: <http://robots.mobilerobots.com/wiki/MobileSim>
- [3] Activmedia robotics. [Online]. Available: <http://www.activrobots.com/>
- [4] Player/stage/gazebo project. [Online]. Available: <http://playerstage.sourceforge.net/index.php?src=index>
- [5] (2009) The USARSim website. [Online]. Available: <http://usarsim.sourceforge.net/>
- [6] B. Balaguer, S. Balakirsky, S. Carpin, M. Lewis, and C. Scrapper, "USARSim: a validated simulator for research in robotics and automation," in *IROS'08. Workshop on robot simulators: available software, scientific applications and future trends*, Nice, France, 2008.
- [7] P. de la Puente, D. Rodríguez-Losada, A. Valero, and F. Matía, "3D Feature Based Mapping Towards Mobile Robots Enhanced Performance in Rescue Missions," in *Proc. IEEE International Conference on Intelligent Robots and Systems (IROS'09)*, 2009.
- [8] (2009) The RoboCup rescue website. [Online]. Available: <http://www.robocuprescue.org/>
- [9] (2009) The RoboCup rescue website. virtual robots competition. [Online]. Available: <http://www.robocuprescue.org/wiki/index.php?title=VRCompetitions>
- [10] A. Valero, G. Randelli, P. de la Puente, D. Calisi, D. Rodríguez-Losada, F. Matia, and D. Nardi, "UPM-SPQR rescue virtual robots team description paper," Team Description Paper for RoboCup 2009.
- [11] MobileRobots Inc. [Online]. Available: <http://www.mobilerobots.com>
- [12] Amtecrobotics. [Online]. Available: <http://www.amtec-robotics.com/index.html>
- [13] Opencv libraries. [Online]. Available: <http://sourceforge.net/projects/opencvlibrary/>
- [14] (2009) The OpenRDK website. [Online]. Available: <http://openrdk.sourceforge.net/>
- [15] P. de la Puente, D. Rodríguez-Losada, R. López, and F. Matía, "Extraction of geometrical features in 3D environments for service robotic applications," in *Proc. 3rd. International Workshop on Hybrid Artificial Intelligent Systems (HAIS'08)*, ser. LNCS, A. A. P. W. Corchado, E., Ed., vol. 5271. Springer-Verlag, 2008, pp. 441–450.