Development of an Autonomous Rescue Robot Within the USARSim 3D Virtual Environment

Giuliano Polverari, Daniele Calisi, Alessando Farinelli, and Daniele Nardi

Dipartimento di Informatica e Sistemistica
Università di Roma "La Sapienza"
Via Salaria 113, 00198 Roma, Italy
<lastname>@dis.uniroma1.it

Abstract. The increasing interest towards rescue robotics and the complexity of typical rescue environments make it necessary to use high fidelity 3D simulators during the application development phase. USARSim is an open source high fidelity simulator for rescue environments, based on a commercial game engine. In this paper, we describe the development of an autonomous rescue robot within the USARSim simulation environment. We describe our rescue robotic system and present the extensions we made to USARSim in order to have a satisfying simulation of our robot. Moreover, as a case study, we present an algorithm to avoid obstacles invisible to our laser scanner based mapping process.

1 Introduction

Robotic systems have been proposed in recent years in a variety of settings and frameworks, pursuing different research goals, and successfully applied in many application domains. Technological improvements both in the hardware and in the associated software of robotic platform push their application towards more and more complex scenarios.

Search and Rescue robotics is one of the most challenging and interesting application environments for AI and robotics. Such an application requires the robots to be equipped with several complex sensors and to be able to perform complex manoeuvres in cluttered and unstructured spaces.

When working with an expensive and complex hardware, the presence of a simulator is of significant importance. On the one hand, it enables the evaluation of different alternatives during the robot system design phase leading to better decisions and cost savings. On the other hand, it supports the process of software development by providing a replacement when robots are not available (e.g. broken or used by another person) or unable to endure long running experiments. Furthermore, the simulation offers the possibility to perform an easier and faster debugging phase.

Several robotic simulators for 3D environments have recently been developed, providing a valid alternative to the canonical 2D-oriented ones. A high fidelity 3D environment adds to the simulation the possibility to test extremely realistic interactions, with a superior graphic rendering, extending the range of sensors to be tested.

G. Lakemeyer et al. (Eds.): RoboCup 2006, LNAI 4434, pp. 491–498, 2007.

USARSim is an open source 3D simulator for the urban search and rescue (USAR) environment based on a commercial game engine, currently supported by an international community.

This paper aims to describe the realization of an autonomous robotic system for search and rescue missions using USARSim. The robotic system is based on a Pioneer P3AT[1] commercial platform equipped with a sonar ring. We customized the platform adding a SICK Laser Range Finder, a Stereo Color Camera mounted on a pan-tilt unit, an Infra Red Sensor and a wireless acces point to communicate with a ground station. The purpose of the robotic system is the autonomous exploration of a rescue scenario searching for victims and building the map of the explored area. The autonomous navigation system, which is based on a two level path-planner, is able to guarantee safe navigation in highly cluttered space [8]. The mapping system is based on Laser Range Finder readings and uses a scan matcher based approach so to localize the robot and build the map. Finally, Stereo Vision is used to detect victims.

The first task was to build an interface between USARSim and our robotic development platform to simulate our real robot and its equipment. In particular we both modeled our system with the available built-in features (e.g. Pioneer robotic platform and SICK Laser Range Finder) and extended the simulator, so to correctly represent all our equipment (e.g. the Stereo Color Camera). Moreover, we improved the existing simulation environment, synchronizing sensor readings and correcting the simulation of transparent objects. Interfacing our development platform with USARSim we are able to test the same code on both the real robot and the simulator: as a consequence, we are now able to use USARSim as a powerful debugging environment in the development phase of our robotic applications.

Furthermore, we present a case study concerning path-planning in unknown and cluttered environments. We modeled in USARSim several test scenarios and developed a speed tracking based stall recovery subsystem to deal with invisible obstacles. We tested the algorithm in USARSim, saving time and preserving the robot from dangerous impacts.

The paper is organized as follows: in the next Section we describe the USARSim simulator. Section 3 shows our work with the simulator, the interface we built and the customization we made. In Section 4 we discuss the case study. Section 5 discusses related works and Section 6 concludes the paper.

2 USARSim

USARSim (presented in [1]) is a 3D high fidelity simulator of USAR robots and environments. USARSim can be a valid tool for the study of basic robotic capabilities in 3D environment. USARSim provides a high quality rendering interface and it is able to accurately represent the robotic system behavior.

USARSim development started in the University of Pittsburgh and is currently supported by an international community. It is released as open source

[1] ActiveMedia: Pioneer. http://www.activrobots.com

software[2] and has been adopted as the standard simulation tool for the RoboCup[3] Virtual Robots Competition in the upcoming 2006 edition.

The current version of USARSim consists of: i) standardized environmental sample models; ii) robot models of several commercial and experimental robots; iii) sensor models, like Laser Scanners, Sonars and Cameras; iv) drivers to interface with external control frameworks, like MOAST, Pyro and Player.

USARSim uses Epic Games Unreal Engine 2[4] to provide a high fidelity simulation at low cost. Unreal is one of the leading engines in the first-person shooter genre and is widely used in both the game industry and in the academic community. The use of the Unreal Engine provides several interesting features to USARSim: i) a high-quality, fast 3D scene rendering, supporting mesh, surface (texture) and lighting simulation; ii) a high fidelity rigid body physical simulator, Karma, supporting collision detection, joint, force and torque modeling; iii) a design tool, UnrealEd, that enables developers to build their own 3D robot models and environments; iv) an object-oriented scripting language, UnrealScript, which supports state machine, time based execution, and networking; v) an efficient client-server architecture to support multiple players.

3 Modeling an Autonomous Rescue Robotic System in USARSim

To fully integrate our robotic rescue system within the USARSim virtual environment we performed the following steps: i) we modeled our robotic platform in the USARSim framework and developed a low level interface to the simulator environment; ii) we modified the simulator to improve sensors' realism; iii) we introduced in USARSim a Stereo Vision Camera sensor and a 3D Camera.

In the following, we discuss each phase of the development. Moreover, we show some validation results concerning autonomous exploration in a USARSim simulated environment.

3.1 Modeling Our Robot in USARSim and Building the Interface

The robot we currently use is a Pioneer P3AT. We equipped the virtual chassis (already modeled in USARSim) with a full Sonar ring made of 16 sensors, a SICK Laser Range Finder and Camera mounted on a Pan-Tilt unit. Figure 1 shows a comparison between our real robot and its model in USARSim.

Our development framework, is based on a set of independent modules that interact and communicate among each other using a centralized blackboard-type repository [4]. To interact with the USARSim environment we built specific modules that directly communicate with the USARSim server. Since these modules use the standardized framework interface, they can be directly replaced with those that communicate with real hardware or different simulator environments.

[2] USARSim project page: http://sourceforge.net/projects/usarsim
[3] RoboCup 2006: http://www.robocup2006.org
[4] Epic Games: Unreal Engine. www.epicgames.com

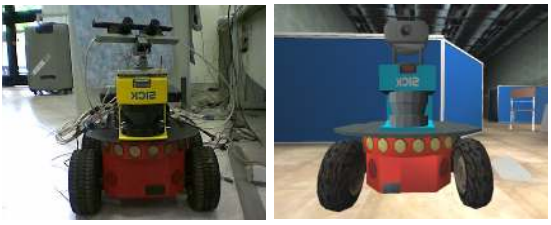

Fig. 1. Our robot and its model in USARSim

This way, we can use all the other modules (e.g. navigation module, mapping module, etc.) without the need of any modification.

In particular, we developed four basic modules: i) the **robot** module, which manages the communication socket, receives and stores odometry and current speed data and sends motion commands to the server; ii) the **laser** module, which stores data gained from the simulated Laser Scanner Sensor, and views/changes its configuration; iii) the **sonar** module, which manages a set of simulated Sonar Sensors; iv) the **camera** module. Camera Sensor simulation is obtained in USARSim using the video feedback of Unreal Client, the Unreal Engine application for 3D scene rendering; in particular, an ImageServer is provided to capture the Unreal Client data and to serve it through TCP/IP. Our camera module holds a dedicated socket to connect the ImageServer and get the virtual Camera data; moreover, the camera module is used to view the Camera configuration and to move the simulated Pan-Tilt unit.

3.2 Improving Sensors' Simulation

USARSim does not provide timestamp information for sensor readings. However, when processing data coming from different sensors, synchronization can be a critical issue. For example, several of our platform subsystems (e.g. the SLAM subsystem) need timestamps for Odometry, Laser and Sonar readings, in order to calculate data confidence and perform coherent state estimation. We added a timestamp information to the Sonar, Laser and Odometry data.

We experienced that the simulated Laser Scanner sensor detected transparent objects as if they were opaque. Every object in USARSim holds a "material" property: we modified the Laser Scanner erroneous behaviour, spreading the laser beam over the transparent objects until it hits another material or it reaches the sensor max range. Thanks to such a modification we have been able to test in USARSim our scanmatcher-based SLAM (simultaneous localization and mapping) and the glass detection subsystem for the identification of transparent materials (which are undetectable by the Laser Scanner) based on the Sonar data.

3.3 Stereo Vision in USARSim

Naturally enough, within a Rescue environment the victim recognition subsystem carries a major weight. Our current approach uses a human detection

algorithm driven by a Stereo Vision unit, which is composed of a couple of synchronized cameras with the same orientation.

As seen before, camera sensor simulation is obtained in USARSim through the capture of the video feedback of Unreal Client. Currently, only one running copy of Unreal Client at a time is allowed for each operating system. This limit comes from the single-user nature of the simulation: consequently Unreal-based Stereo Vision seems to be impossible until future versions of the Unreal Engine are released.

Since it is impossible to have multiple camera simulation on the same screen, we extended the robot definition code. Each virtual robot is described in the simulation by an Unreal Script definition code, storing information about its model and instructions to handle input data, to make movements and to draw the camera data.

We modified the function usually used to draw double-exposure images on the screen. Every time a frame is being drawn on the screen, we split vertically the output window overriding the first half with the left camera data and the second with the right camera data, maintaining data synchronization. With this new self-developed Stereo Vision sensor we are now able to have a complete high fidelity simulation of our rescue robot.

3.4 3D Camera Sensor

The Swiss Ranger Camera[5] is a sensor able to add a distance information to every pixel of the image data captured by its internal camera. Such sensor can be extremely useful in the USAR environment, both for navigation and for victim detection.

We added a Swiss Ranger Camera simulation in USARSim introducing a new IRC (Infra-Red Range Camera) sensor providing, for each pixel, the distance from the objects in the scene. By using the IRC sensor together with an ordinary Camera with the same position, orientation and resolution, we add the distance information to every pixel of the camera image, obtaining a simulation of the Swiss Range Camera.

Figure 2 shows, side by side, the Camera feedback (on the left) and the IRC sensor output (on the right, the brightness is proportional to the distance).

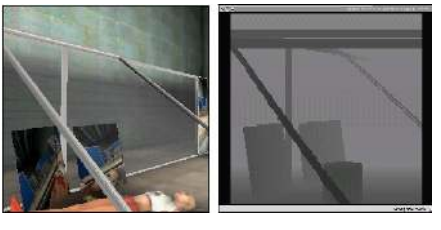

Fig. 2. A Camera image and the corresponding IRC sensor feedback

[5] CSEM: Swiss Ranger Camera. http://www.swissranger.ch/products.php

3.5 Validation Results

We performed several tests to validate the whole system configuration. We placed our robot into different USARSim virtual environments, to perform an autonomous exploration. The system behaviours consistently matched the real robotic system. In particular, we verified that the data gained from the sensors and the motion commands execution were as expected.

Figure 3 shows our rescue robot while autonomously exploring an unknown virtual 3D environment generated by USARSim. In the map on the right the unknown parts are drawn in blue (grey), while walls and obstacles are drawn in black and free space in white. The small table in front of the robot is not drawn by the SLAM module (i.e. in black), because it is invisible to the Laser Range Finder. However, our stall recovery subsystem, described in the following paragraph, identifies the impact surface and draws it on the map.

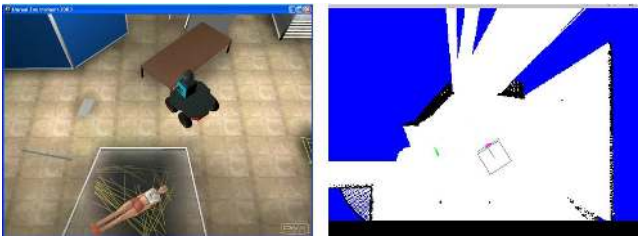

Fig. 3. Our rescue robot exploring an unknown virtual environment

4 Case Study: Exploration with Invisible Obstacles

During autonomous exploration missions in rescue environments, stall problems ofter arise. Our frontier based autonomous exploration subsystem, presented in [8], uses a two-level approach for navigation. It is based on a global topological path-planner and on a local motion planner, which is an extension of the well-known Randomized Kinodynamic Tree [9]. This kind of algorithms works by building a tree of safe, randomly-generated robot configurations. This local motion planner may be stuck by obstacles that are undetectable by the Laser Scanner, because they do not lie on its scanning plane.

We built a **stall recovery subsystem**, whose development was highly simplified because of the use of USARSim: the simulated environment helped us to save testing time and to preserve the real robot from dangerous impacts with unknown obstacles. We modeled small obstacles such as a tube, a ramp and a small table and observed the reactions of the virtual robot towards these objects.

The main cycle of the subsystem is composed of the following steps:

1. The subsystem first calculates the actual value of linear and angular speed, given the actual and previous robot poses (from the SLAM subsystem).
2. The differences between desired and actual speeds are monitored for several positions around the robot surface, using different stall conditions.

3. To avoid false positives we integrate over time the stall conditions.
4. If a stall condition is verified for several cycles, an obstacle is drawn on the map and an alarm is sent to the navigation subsystem to allow for a fast re-planning.

We tested the stall recovery subsystem in USARSim obtaining valuable results. Figure 3 shows on the left, the robot hitting a small table invisible to the laser; on the right, the obstacle representation in the robot map. Subsequent tests were performed on the real robot, using different obstacles such as chairs and bricks: the subsystem correctly identified stall situations tracking all the objects and allowing complete explorations of the environment.

5 Related Works

Moast[6], is a development framework providing a multi-agent simulation environment, a baseline control system, and a mechanism to migrate algorithms from the virtual world to the real implementation. Moast is intended to provide USARSim users with a customizable control system allowing for a high level interaction with the simulator. Compared to Moast, our system does not need to migrate the developed algorithms to the real implementation; in fact, our system runs indifferently on the real robot and on virtual environments.

Several works related to USARSim focus on validation of sensors such as Laser Scanner [2] or robot mobility [3]. In comparison to these works, we focused more on improving sensor data coherence (e.g. synchronizing sensor readings and testing sensorial fusion tasks) than on validating single sensor simulation.

As for obstacles which are not detectable by Laser Scanner sensors or cameras, different solutions are proposed in literature. Several approaches are based on touch sensors: for example in [5] the authors describe a cylindrical robot with a total coverage bumper, while in [6] an actuated whisker is used to identify objects. Such approaches however require additional sensors. Another way to address the problem is proposed in [7]. In this work the authors describe a mobile robot used as a tour guide, which is able to deal with invisible objects given the known map of the environment, lowering the speed when the localization error is higher. Unfortunately such, a technique is useless in a USAR environment, where the environment map is not known in advance.

To the best of our knowledge, the rescue system presented in this paper is one of the first complete autonomous rescue systems both working on real robots and integrated in the USARSim simulator.

6 Conclusions

Our experience in simulation before USARSim was limited to two dimensions. Several features of our robotic system such as the glass detection subsystem or the victim recognition subsystem were impossible to test during a simulated mission. Using USARSim, we had the widely acknowledged advantages of a high

[6] MOAST Project page. http://moast.sourceforge.net

fidelity 3D simulation, such as an accurate model of robot mechanics, different materials available on 3D surfaces etc.

In this paper we presented the development of an autonomous working system within USARSim. We modeled our robotic system within USARSim, significantly extending the simulation environment. In particular, we added the possibility to use Stereo Vision for our victim recognition subsystem, and synchronized all sensor readings in order to have a coherent map building process. Moreover, we addressed the problem of safe navigation in presence of obstacles which are invisible to the 2D Laser based mapping process. We proposed a solution to this problem and tested our system in the USARSim virtual environment.

The performed tests within the USARSim virtual environment of our robotic system confirm that such a framework is suitable for preliminary validation during the robotic application development phase. In fact, using our virtual robotic system we have been able to conduct experiments involving invisible obstacles preserving the real robot's integrity. Moreover, we can now perform a high fidelity experimental analysis of different rescue system configurations without the need to modify the actual robotic platform.

As a future work we plan to deeply investigate the interactions between the invisible obstacle detection process and the navigation and mapping process. In particular, it would be interesting to represent invisible obstacles as dangerous or forbidden configurations inside the navigation world model, and to study how this different obstacle representation would impact on the system performance.

References

1. Wang, J., Lewis, M., Gennari, J.: USAR: A Game-Based Simulation for Teleoperation. In: Proc. 47th Ann. Meeting Human Factors and Ergonomics Soc. (2003)
2. Carpin, S., Birk, A., Lewis, M., Jacoff, A.: High fidelity tools for rescue robotics: results and perspectives. In: RoboCup International Symposium 2005 (2005)
3. Wang, J., Lewis, M., Koes, M., Carpin, S.: Validating USARsim for use in HRI Research. In: Proc. of the Human Factors And Ergonomics Society 49th Annual Meeting, pp. 457–461 (2005)
4. Farinelli, A., Grisetti, G., Iocchi, L.: SPQR-RDK: a modular framework for programming mobile robots. In: Proc. of Int. RoboCup Symposium (2004) pp. 653–660 (2004)
5. Jones, J.L., Flynn, A.M.: Mobile Robots - Inspiration to Implementation, A K Peters Ltd. Wellesley, Massachusetts (1993)
6. Scholz, G.R., Rahn, C.D.: Profile Sensing with an Actuated Whisker. IEEE Transactions on Robotics and Automation 20(1), 124–127 (2004)
7. Fox, D., Burgard, W., Thrun, S., Cremers, A.: A hybrid collision avoidance method for mobile robots. In: Proc. IEEE Int'l Conf. on Robotics and Automation (1998)
8. Calisi, D., Farinelli, A., Iocchi, L., Nardi, D.: Autonomous Navigation and Exploration in a Rescue Environment. In: RoboCup International Symposium 2004 (2004)
9. LaValle, S.M., Kuffner, J.J.: Randomized Kinodynamic Planning. In: Proc. of IEEE Int'l Conf. on Robotics and Automation (1999)