High fidelity tools for rescue robotics: results and perspectives

Stefano Carpin[1], Jijun Wang[2], Michael Lewis[2],
Andreas Birk[1], and Adam Jacoff[3]

[1] School of Engineering and Science
International University Bremen – Germany
[2] Department of Information Science and Telecommunications
University of Pittsburgh – USA
[3] Intelligent Systems Division
National Institute of Standards and Technology – USA

Abstract. USARSim is a high fidelity robot simulation tool based on a commercial game engine. We illustrate the overall structure of the simulator and we argue about its use as a bridging tool between the RoboCupRescue Real Robot League and the RoboCupRescue Simulation League. In particular we show some results concerning the validation of the system. Algorithms useful for the search and rescue task have been developed in the simulator and then executed on real robots providing encouraging results.

1 Introduction

Urban search and rescue (USAR) is a fast growing field that obtained great benefits from the RoboCup competition. Society needs robust, easy to deploy robotic systems for facing emergency situations. The range for potential applications is very wide. Fire fighters inspecting vehicles transporting hazardous materials involved in road accidents is one end, while locating people and coordinating big rescue teams after a major natural disaster like a earthquake or a tsunami is at the other side of the spectrum. The two leagues introduced in the Robocup competition somehow represent these two extremes. Currently, in the Real Robot League the issues being addressed concern mainly locomotion, sensing, mapping and localization. Up to now, very few attempts were made in the direction of autonomy and cooperation using teams of robots. The technical difficulties encountered while dealing with the formerly indicated aspects still dominate the scene. In the Simulation League, the problem is addressed from the other side. Large teams of heterogenous agents with high level capabilities have to be developed. Topics like coordination, distributed decision making, multi-objective optimization are some of the crisp matters being addressed. It is part of the overall vision that in the future the two scientific communities will move towards each other, and will eventually meet. It is nevertheless evident that this is not going to happen soon. Deploying a team of 20 autonomous robots performing a rescue task over an area of a few hundred square meters and for a time

horizon of hours is beyond the current capacity. In this context, we illustrate a simulation project called USARSim that has been developed at the University of Pittsburgh. We envision that USARSim is the tool needed to foster and accelerate cooperation between the formerly described communities. On the one hand, USARSim allows a high fidelity simulation of real robots, and offers great flexibility when it comes to model new environments or hardware devices. On the other hand, the software allows the simulation of reasonably sized teams of agents. Section 2 describes the USARSim simulation software. Next, in section 3 we illustrate our current experience in using the USARSim software to develop algorithms to be used to control real robots. Finally, conclusions are offered in section 5.

2 USARSim

USARSim is a high fidelity simulation of USAR robots and environments intended as a research tool for the study of HRI and multi-robot coordination. USARSim supports HRI by accurately rendering user interface elements (particularly camera video), accurately representing robot automation and behavior, and accurately representing the remote environment that links the operator's awareness with the robot's behaviors. The current version of USARSim consists of: environmental models (levels) of the National Institute of Standards and Technology (NIST) Yellow, Orange, and Red Arenas, and Nike site which serve as standardized disaster environments for mobile robot studies, robot models of commercial and experimental robots, and sensor models. USARSim also provides users with the capabilities to build their own environments and robots. Its socket-based control API allows users to test their own control algorithms and user interfaces without additional programming. USARSim uses Epic Games' Unreal Engine 2 [1] to provide a high fidelity simulator at low cost. Unreal is one of the leading engines in the "first-person shooter" genre and is widely used in the gaming industry. It is also gaining a strong following in the academic community as more researchers use it in their work. Recent academic projects have included creating VR displays [2], studying AI techniques [3], and creating synthetic characters [4]. In addition to the egocentric perspective, there are several other features of the Unreal Engine that make it particularly appealing for HRI research.

- **Graphics**: The Unreal Engine provides fast, high-quality 3D scene rendering. It supports mesh, surface (texture) and lighting simulation, and can import models from other popular modeling tools such as Maya [5] and 3D Studio Max [6]. Moreover, its dynamic scene graph technology enables simulation of mirrors, glass and other semi-reflective surfaces. The high fidelity of these graphics allows the Unreal engine to simulate realistic camera video, the most critical feature in current approaches to human control of mobile robots.
- **Physics engine**: The Unreal Engine integrates MathEngine's Karma Engine [7] to support high fidelity rigid body simulation. The details of physical

simulation, including collision detection, joint, force and torque modeling are encapsulated within the high level game programming language. This feature lets the simulation replicate both the physical structure of the robot and its interaction with the environment.

- **Authoring tool**: The Unreal Engine provides a real-time design tool for developers to build their own 3D models and environments. The editing tool, UnrealEd, is fully integrated into Unreal Engine to provide users a what-you-see-is-what-you-get style authoring tool. UnrealEd permits HRI researchers to accurately model both robots and their environments.
- **Game Programming**: The Unreal Engine provides an object-oriented scripting language, UnrealScript, which supports state machine, time based execution, and networking on a programming language level. With UnrealScript, the rules of the simulation can be manipulated. This affords the ability to customize the interaction with the simulation to match the specifics of desired robot behaviors.
- **Networking**: The Unreal Engine uses an efficient client-server architecture to support multiple players. This embedded networking capability allows USARSim to support control of multiple robots without modification.

Figure 1 shows Unreal Engine components and the expandable library of robot-themed models and environments and control interfaces to acquire sensor data and issue commands we have added to create the USARSim simulation.

2.1 Robot models

USARSim currently provides detailed models of six robots: the Pioneer P2AT and P2DX [8], iRobot ATRV-Jr, the Personal Exploration Rover (PER) [9], the Corky robot built for this project and a generic four-wheeled car. Figure 2 shows some of these simulated and real robots. These models were constructed by building the components of the robot and defining how these parts were connected using joints which serve as mechanical primitives for the Karma physics engine. Since the physics engine is mechanically accurate, the resulting movement of the aggregate robot is highly realistic. Karma uses a variety of computational strategies to simplify, speed up, and exclude non- interacting objects to achieve animation level speed without sacrificing physical fidelity. Because USARSim is intended for use by researchers from diverse backgrounds we have added a reconfigurable robot model to allow researchers to construct and customize their own robots without detailed mechanical modeling. Building a new robot model using this facility is as simple as 1) building geometric models for the robot, 2) configuring the robot model to specify the physical attributes of the robot and define how the chassis, parts and auxiliary items are connected to each other and 3) performing additional programming only if the robot needs features or behaviors not included in the robot model.

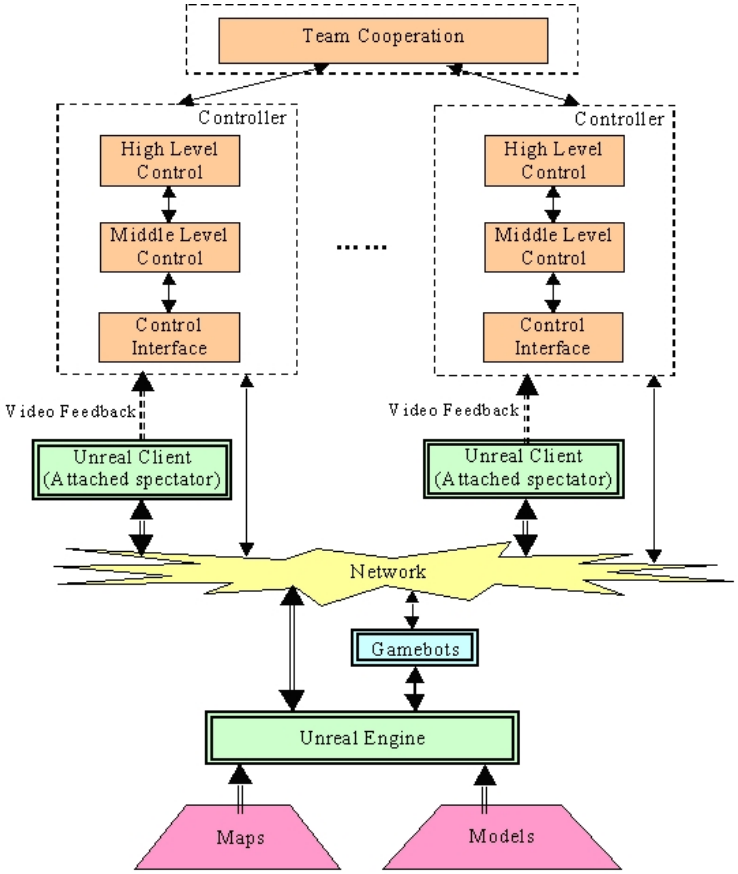

Fig. 1. System architecture

2.2 USAR environments

USARSim includes detailed models of the NIST reference arenas [10], [11] and will soon include a replica of the fixed Nike site reference environment. Significantly larger disaster environments are under development for the Virtual Robot USAR demonstration at RoboCup 2005. To achieve high fidelity simulation, 3D CAD models of the real arenas were imported into Unreal and decorated with texture maps generated from digital photos of the actual environments. This ensures geometric compatibility and correspondence between camera views from the simulation and actual arena. In addition to this basic structure, the simulated environments include the real and simulated USAR arenas (figure 3). A collection of virtual panels, frames, and other parts used to construct the portable arenas are included with USARSim. These efforts attempt to simulate specific, physical spaces. Using the UnrealEd tool, it is possible to rearrange these el-

Fig. 2. Some robots in USARSim

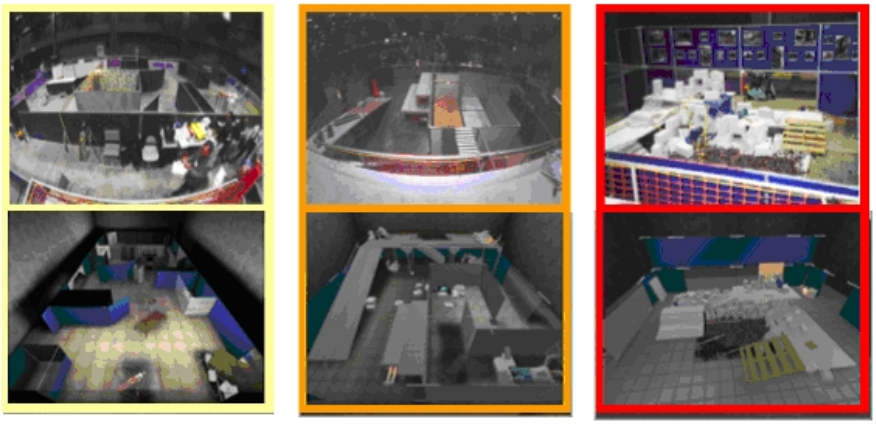

Fig. 3. The real and simulated USAR arenas

ements to quickly develop alternate USAR layouts in much the same way the arenas are reconfigured during USAR contests.

2.3 Sensor models

Sensors are a critical part of the simulation because they both provide the basis for simulating automation and link the operator to the remote environment. USARSim simulates sensors by programmatically manipulating objects in the Unreal Engine. For example, sonar and laser sensors can be modeled by querying the engine for the distance given the sensor's position and orientation to the first object encountered. To achieve high fidelity simulation, noise and data distortion are added to the sensor models by introducing random error and tailoring the data using a distortion curve. Three kinds of sensors are simulated in USARSim.

– Proprioceptive sensors: including battery state and headlight state.
– Position estimation sensors : including location, rotation and velocity sensors.
– Perception sensors: including sonar, laser, sound and pan-tilt-zoom cameras.

USARSim defines a hierarchical architecture (figure 4) to build sensor models. A

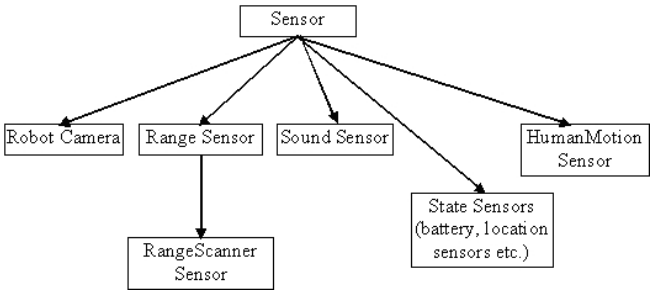

Fig. 4. Sensor Hierarchy Chart

sensor class defines a type of sensor. Every sensor is defined by a set of attributes stored in a configuration file. For example, perception sensors are commonly specified by range, resolution, and field-of-view. To get a sensor with specified capability, we can either directly configure a sensor class or derive a new sensor from an existing sensor class. Once the sensor is configured, it can be added to a robot model, by simply including a line in the robot's configuration file. A sensor is mounted on a robot specified by a name, position where it is mounted and the direction that it faces.

2.4 Simulating video

Cameras provide the most powerful perceptual link to the remote environment and merit a separate discussion. The scenes viewed from the simulated camera are acquired by attaching a spectator, a special kind of disembodied player, to the camera mount on the robot. USARSim provides two ways to simulate camera feedback. The most direct is to use the Unreal Client as video feedback, either as a separate sensor panel or embedded into the user interface. While this approach is the simplest, the Unreal Client provides a higher frame rate than is likely to be achieved in a real robotic system and is not accessible to the image processing routines often used in robotics. The second method involves intermittently capturing scenes from the Unreal Client and using these pictures as video feedback an approach that is very close to how a real camera works. USARSim includes a separate image server that runs alongside the Unreal Client. This server captures pictures in raw or jpeg format and sends them over the network to the user interface. Using this image server, researchers are able to

better tune the properties of the camera, specifying the desired frame rate, image format and communication properties to match the camera being simulated.

3 Validation: preliminary results

Mapping is one of the fundamental issues when rescue robots are used to assist humans operating in buildings. Maps help rescue operators while finding victims and avoiding dangerous areas. To this end, at the International University Bremen (IUB) we investigated different mapping strategies with real robots. In particular, we focused on grid based maps, and we also tackled the problem of multi-robot map merging [12]. Recently, however, we have decided to move towards other approaches, like SLAM [13], that require the identification of features. In this context, we started to develop algorithms to extract natural landmarks in unstructured environments. The problem of features extraction has been faced first in simulation and currently on the real robots. The IUB rescue robots are self developed systems (see figure 5). The platform has a six-wheels

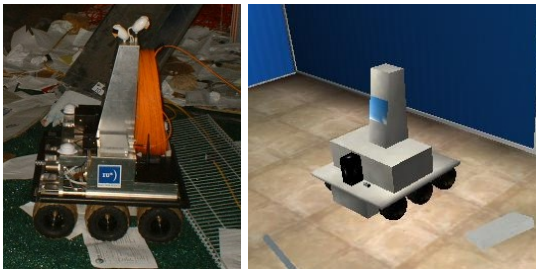

Fig. 5. On the left, the rescue platform developed at IUB. On the right, the model of the same robot while performing into the simulated yellow arena

differential drive, and is equipped with a number of different sensors, including a proximity range finder, odometry, an orientation sensor, and a set of different cameras [14]. Victim detection is human supervised, and is assisted by an infrared camera and a CO_2 sensor. Mapping is performed using the robot's pose (provided by odometry and orientation sensor) and the data coming from the range finder. Developing the model of the IUB robot for the USARSim software has been a straightforward process. USARSim is shipped with the models of several robots based on differential drive platforms. This allowed us to quickly develop the kinematic model of the robot. The proximity range sensor provided in the USARSim simulation environment can be configured in terms of the number of beams used to sample the sweeped area, the maximum reachable distance, and the noise. The real sensor we use (Hokuyo PB9-11) sweeps an area of 162 degrees with 91 beams. Its detection distance is 3 meters, and we experimentally

determined that under the conditions found in the IUB rescue arena the signal to noise ratio is about 30 dB. These properties can be easily transferred to the parameters controlling the simulated range finder. We first run the simulated robot into the model of the IUB rescue arena [15] and gathered the data produced by the simulated proximity range finder. Then, we run the real robot into the real arena and collected the same data. Features extraction at the moment is based on the Hough transform, a widely used tool coming from image processing and used also in robotics. For line detection, the parameterized form of the line is used, i.e. $\rho = xcos\theta + ysin\theta$ where ρ is the perpendicular distance of the line from the origin and θ is the angle that the normal makes with the x axis. The basic idea is that the Hough Transform maps points from the Cartesian space to the (ρ,θ) Hough space. Each point in the Cartesian space corresponds to a sinusoidal curve in the Hough Space. Once the hough transform has been performed on the image, a simple voting scheme can be set up in the hough space. In this way, for a given range of values for ρ and θ, each point in the Cartesian space is mapped to the hough space which accumulates the values in a two-dimensional histogram. Local maxima of this histogram correspond to lines detected in the image. In the case of an image, local maxima can easily be found by an appropriate hill climbing algorithm. However in the case of range finder data, we have only a few data points and a rather vast space for ρ and θ. This results in a very sparse accumulator for which hill climbing is not ideally suited. So in this case, it makes sense to find the global maximum, remove those scan points that contribute to this line, and repeat the procedure until the global maximum drops below a certain threshold of the initial maximum. Note that the discretization of the Hough space must be tuned according to the problem. If the discretization is too fine, we might find several local maxima too close to each other in the Hough space. However, the degree of discretization directly affects the precision of the detected lines, so it should not be set too low. The following figures show a comparison between the data collected with the simulator (figure 6) and with the real robot (figure 7), as well as the corresponding Hough transforms. The fine

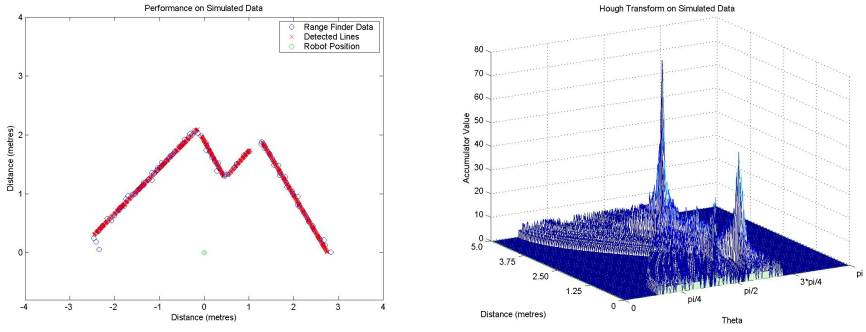

Fig. 6. On the left, the data collected with USARSim. On the right, the Hough transform calculated on the same data

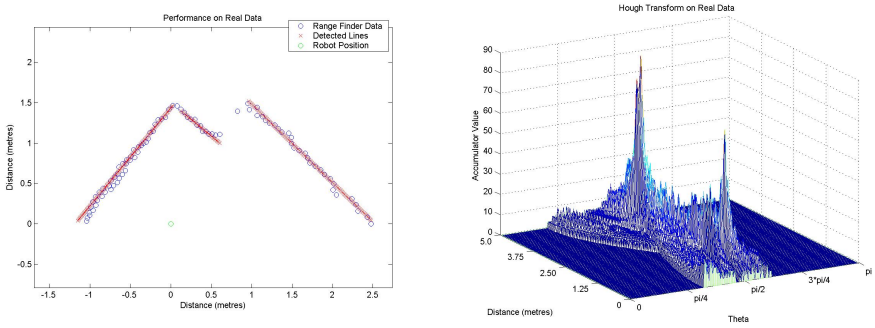

Fig. 7. On the left, the data collected with the real robot. On the right, the Hough transform calculated on the same data

tuning of parameters was completely performed within USARSim. No change was necessary when the code was later used to perform the same processing on real data.

4 Future work

Though the initial results with USARSim appear promising, further developments are needed in order to make it a really effective tool. Specifically, more models of robot sensors are under planning and will be developed. Given the importance that video input has in real robots, the improvement of its simulation is a must. For example stereo vision, which proved to be highly successful in the Real Robots League will be available inside USARSim as well. Along the same lines, the possibility to simulate infrared cameras has to be considered, because of their high potential for victim recognition. Also other sensors like CO_2 probes, thermometers, and similar will be included.
Along the same lines, more robot models will be developed. This is particularly important in the search and rescue domain, where custom platforms with high mobility are often developed to overcome obstacles. With this respect, one of the more urgent aspects to address is the simulation of tracked vehicles. Finally, in order to make the migration of code developed under USARSim towards real robots an easy task, common interfaces need to be developed. This is already partially achieved by the Player [16] interface currently available, though at the moment few experiments in this direction have been performed.

5 Conclusions

In this paper we introduced USARSim, a high fidelity simulation tool which supports development of advanced robotic capabilities in complex environments such as those found in the urban search and rescue domain. We showed how

USARSim's detailed models of the arenas used to host the RoboCupRescue Real Robot League competitions, along with kinematically correct robot models and simulated sensors, can provide a rich environment for development of robotic behaviors and innovative robot designs. We further showed initial experimental results captured at IUB which demonstrate that a high level of correlation can be obtained between the simulated environment and real arenas. This was shown through a feature extraction example using a simulated range sensor within the simulated environment and a real range sensor deployed on a robot within an actual arena. Such correlation reinforces expectations that algorithms developed, and shown to be effective, within the simulated environment can be transferred to real robots with reasonable expectations of effectiveness; thus USARSim can reduce the need for costly and problematic robot hardware to support iterative development and testing practices.

USARSim's usefulness in this regard will be on display to the community at this year's RoboCup 2005 in Osaka, Japan, where it will be demonstrated as the basis for a new league to compliment the existing RoboCupRescue leagues: the Real Robot League and the Simulation League. Specific rules for this proposed league, called the RoboCupRescue Virtual Robot, are under development. But the performance metric used in the Real Robot League has been adopted to maintain close ties between league goals and approaches. Existing models of the rescue arenas housed at NIST, IUB, and elsewhere are already available for dissemination, along with several robot models and sensors described previously in this paper. The plan is for each year's competition to feature an entire building, with partially and fully collapsed sections, which will be made available as practice environments after the competition. This proposed league, if adopted after the Osaka demonstration, would provide a logical link between the required perception and negotiation of physical environments within the Real Robot League arenas, and the citywide responder allocation tasks associated with the Simulation League. The goal is to ultimately combine efforts in all three levels of abstraction to help develop and demonstrate comprehensive emergency response capabilities across a city and within particular buildings.

Acknowledgments

The authors thank the numerous students who conducted the preliminary validation at the International University Bremen and at the University of Pittsburgh.

References

1. Epic games: Unreal engine. www.epicgames.com (2003)
2. Jacobson, J., Hwang, Z.: Unreal tournament for immersive interactive theater. Communications of the ACM **45** (2002)
3. Laird, J.: Research in human-level ai using computer games. Communications of the ACM **45** (2003) 32–35

4. Young, M., Riedl, M.: Towards an architecture for intelligent control of narrative in interactive virtual worlds. In: ACM Conference on Intelligent User Interfaces. (2003)
5. Aliaswavefront: Maya. http://www.alias.com/eng/products-services/maya/index.shtml (2004)
6. Discreet: 3d studio max. http://www4.discreet.com/3dsmax (2004)
7. Karma: Mathengine karma user guide. http://udn.epicgames.com/Two/KarmaReference/KarmaUserGuide.pdf (2003)
8. Activemedia: Pioneer. http://www.activrobots.com/ (2005)
9. Nourbakhsh, I., Hamner, E., Porter, E., Dunlavey, B., Ayoob, E., Hsiu, T., Lotter, M., Shelly, S.: The design of a highly reliable robot for unmediated museum interaction. In: Proceedings of the IEEE International Conference on Robotics and Automation. (2005)
10. Jacoff, A., Messina, E., Evans, J.: Performance evaluation of autonomous mobile robots. Industrial Robot: An International Journal **29** (2002)
11. Jacoff, A., Messina, E., Weiss, B., Tadokoro, S., Nakagawa, Y.: Test arenas and performance metrics for urban search and rescue robots. In: Proceedings of the Intelligent and Robotic Systems (IROS) Conference. (2003)
12. Carpin, S., Birk, A.: Stochastic map merging in rescue environments. In: Robocup 2004: Robot Soccer World Cup VIII. Springer (2005)
13. Dissanayake, G., Newman, P., Clark, S., Durrant-Whyte, H., , Csorba., M.: A solution to the simultaneous localisation and map building (slam) problem. IEEE Transactions of Robotics and Automation **17** (2001) 229–241
14. Birk, A.: The IUB 2004 rescue robot team. In: RoboCup 2004: Robot Soccer World Cup VIII. Springer (2005)
15. Birk, A.: The IUB rescue arena, a testbed for rescue robots research. In: IEEE International Workshop on Safety, Security, and Rescue Robotics, SSRR'04. (2004)
16. Vaughan, R., Gerkey, B., Howard, A.: On device abstractions for portable, reusable robot code. In: Proceedings of the IEEE/RSJ IROS. (2003) 2421–2427