Hinomiyagura 2015 TDP for RoboCup 2015 Rescue Infra Structure League : A realistic RoboCup Rescue Simulation based on Gazebo

Masaru Shimizu[1] , Tomoichi Takahashi[2] , Nate Koenig[3] and Arnoud Visser[4]

Chukyo University, Aichi, Japan[1], Meijo University, Aichi, Japan[2], Open Source Robotics Foundation, San Francisco, USA[3], Universiteit van Amsterdam, Amsterdam, The Netherlands[4]
shimizu@sist.chukyo-u.ac.jp, ttaka@ccmfs.meijo-u.ac.jp, natekoenig@gmail.com, A.Visser@uva.nl
https://github.com/m-shimizu/RoboCupRescuePackage

Abstract. More than a decade has passed since RoboCup Virtual Robot competition based on the USARSim interface started. During that time, the underlying simulation platform, Unreal Engine, has seen three major upgrades (UT2004, UT3 and UDK). The interface to USARSim has not been modified, so member of virtual robot league have concentrated on their control and perception algorithms. The Robot Operating System (ROS) has been steadily gaining popularity among robotics researchers as an open source framework for robot control. The community of USARSim is smaller than the community of ROS. Gazebo, the simulation environment native to ROS, is a much better choice for the future. This paper describes a new prototype of the USARSim interface; implemented as plugin to Gazebo, the simulation environment native of ROS. This plugin would attract new teams to the Virtual Robot competition.

1 Introduction

Inside the RoboCup Rescue Simulation League the challenge is to have a team of robots cooperate inside a devastated area. Yet, most research institutes have access to only a few robots with a limited sensor suite and do not have access to all the robotic hardware to built a complete rescue team. Simulators allow to experiment with algorithms for cooperation between robots in a safe, low-cost environment. Unified System for Automation and Robot Simulation (USARSim) environment has been used for many years by robotics researchers and developers as a validated framework for simulation[5]. The approach applied to USARSim is to perform the same experiment in simulation and with a real world system: the Pioneer robot [3], the Kurt3D robot [1], the Kenaf robot [10], the Nao robot [9], the camera sensor [4], the laser sensor [7] and the GPS sensor [2].

The Robot Operating System (ROS) has been steadily gaining popularity among robotics researchers as an open source framework for robot control [11].

was
Gazebo is the simulation environment embedded in ROS, although it was originally developed for the Player-Stage environment [8]. Gazebo is based on the Open Dynamics Engine (ODE) [6]. Table 1 shows functional comparison between USARSim and Gazebo to present the differenece between USARSim based on Unreal and USARSim based on Gazebo. USARSim has many benefits (for instance, the realism of the lighting from the Unreal Engine), but the community of USARSim is smaller than the community of ROS. Table 2 shows the merits of using Gazebo instead of USARSim and indicates that Gazebo is better than UDK for RoboCup Virtual Robot League. Gazebo is a much better choice for the future. This paper describes a prototype of simulation environment based on Gazebo to demonstrate the benefits of RoboCup Rescue community.

2 Design of interface between Gazebo and USARSim

The interface between Gazebo and USARSim is designed as a WorldPlugin; the preferred method to modify the simulation environment. The plugin starts a server-routine, which listens to port 3000. Multiple clients (currently limited to teams of 16 robot controllers, as in the original UT2004 version) can connect to this port and spawn a robot into the Gazebo world.

At the moment, the robot is spawned with a specific sensor-suite as configuration. In principal, an user can modify this configuration in the Gazebo GUI. The USARSim interface has commands to query the current configuration (`GETCONF` and `GETGEO` commands), but the format of the `STATUS` message should be update to notify the robot controller that the configuration has been changed (and should be queried again).

The location of the spawn-position is important, because with a team of robots you want each robot to have an unique start position. In an Unreal world start-positions are specified by inserting PlayerStart positions with their corresponding coordinate system to the world. This is a native feature of Unreal, because Unreal Tournament was a multi-player game where each player also needed an unique start position. In a Gazebo world in a comparible way start positions could be added, once such PlayerStart model is created.

Once a robot is spawned and configured, the regular sense-plan-act cycle starts. Sensor messages are received via `SEN` messages, the actuators are controlled by `DRIVE`, `SET` and `MISPKG` messages.

Each of the robot controllers of the robot team is handled by a separate thread, to prevent direct interaction between the robots. Yet, the threads share network and computational resources which means that they are never completely indepent of each other.

3 Proposal of our system and merits of RSL league

Our proposal is to use a platform which allows to control robots inside Gazebo via the USARSim interface. This would realize following points:

Table 1. Functional comparison between UDK and Gazebo

	USARSim with UDK	USARSim with Gazebo
Simulator	UDK	Gazebo
Physics Engine	Unreal Engine	ODE(Default), Bullet, DART, Simbody
3D Simulation	Possible	Possible
Performance for	Real Time	Accuracy
Kinds of robot included by simulator	AirRobot, ATRVJr, Cooper, ERS, HMMWV, Kenaf, Kurt, Lisa, P2AT, P2DX, Pssarola, QRIO, Rugbot, Sedan, SnowStorm, Soryu, Submarine, Talon, Tarantula, TeleMax, Zerg	Atlas, Kuka, Pioneer 2DX, Pioneer 3AT, PR2, Robo Naut, Quad Rotor, Kuka, youbot
Flying robots	Possible	Possible
Multi robots	Possible	Possible
Capability of adding objects and fields by users	Possible(Not so easy in scaling)	Possible
Capability of adding robot by users	Possible(Not so easy in scaling)	Possible
Realistic rendering	Possible	Possible
Simulation of water , mud , sand	Limited	Possible by change physics engine
Capability of connection with ROS	Possible	Possible but need to convert protocol
Arbitrary viewpoint	1	Any numbers of cameras which you want
Capability of connection with each camera video stream	Impossible	Possible
Getting Ground Truth data from map	Impossible	Possible
Active environment	Possible	Possible
Disaster environment	Possible	Possible
Movable obstacles	Possible	Possible
Lighting and shadowing	Possible	Possible
Fog effect	Possible	Possible

Table 2. Merits of using Gazebo instead of UDK

	USARSim with UDK	USARSim with Gazebo
Commands and sensor data transferring protocol	GameBot Protocol	Topic or Any protocols which you need
Transferring data protocol with ROS	GameBot Protocol(Need protocol convertor at ROS side)	Topic or Any protocols which you need
Changeability inside of simulator	Impossible(Not Open Source)	Possible(Open Source)
Programing Language	Unreal Script	C, C++, Python

- New robots could be dynamically spawned in the world by an `INIT` command.
- The robots could be configured with `SET` commands.
- The robot's sensor suite could be queried with `GETCONF` and `GETGEO` commands.
- The robots could be steered by sending `DRIVE` commands.
- Sensor updates would be send via `SEN` messages.
- Camera images would be published via a separate high-speed socket (typically port 5003)
- A private socket (typically port 50000) should be available for the Wireless Server Simulation, which needs Ground Truth information to calculate distances between robots and the number of walls in the line of sight between the robots.

The approach provides merits of RSL community. Firstly the progress made by the Open Source Robotics foundation in improving Gazebo would be directly available to the RoboCup Rescue Simulation League community. Secondly the Rescue Simulation community and the research institutes active in the DARPA Robotics Challenge would use the same simulation environment, allowing for cross development. Thirdly the maintenance of the simulation environment of the Virtual Robot competition would come in professional hands, now that USARSim is no longer active supported by NIST. Last, but not least, this would allow to attract new teams to the RoboCup Rescue Simulation League.

4 Implementation and simulation examples

This USARSim prototype is build by the following steps:

1. Creation of a design document with an indication where the USARSim interface should be incorporated in the Gazebo architecture(Figure 1 shows connections between an USARSim user client and Gazebo simulator with our prototype)

2. visit of the Open Source Robotics foundation (OSRF), where the design will be discussed and together with the OSRF developers an initial attempt will be made to create the USARSim sockets and send commands and responses over these sockets
3. completion of the full USARSim functionality over these sockets (at our universities)
4. final visit of the OSRF, to find solutions for open problems (probably the high-speed frames of the robot cameras), to test the implementation and to optimize the used solutions

An example of the level of detail needed to provide Ground Truth data for the Wireless Simulation Server over socket 5000 can be seen in Figure 2.

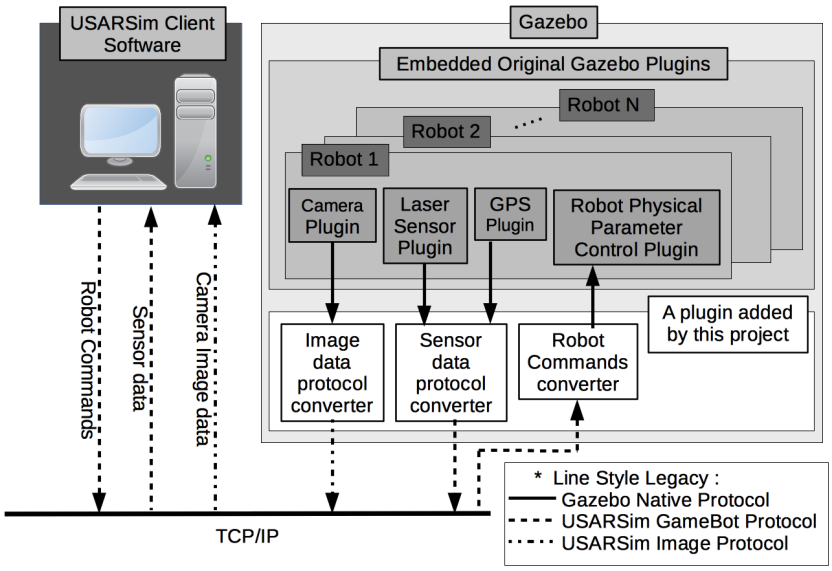

Fig. 1. Diagram of our prototype simulator with Gazebo. At right middle of the diagram, white blocks are our added plugin software in Gazebo. The plugin translate bidirectionally USARSim commands and Camera images and Sensor data between USARSim protocol(GameBot) and Gazebo Native protocol(Topic).

In the current implementation it is possible to query for start poses, to spawn a Pioneer 3AT robot and to publish images over a high speed binary channel. The later accomplishment was the most critical.

We were also able to import a world generated in Unreal Editor into Gazebo. This is quite attractive scenario, because the Editor of Unreal is really on of its biggest virtues. The latest version of Unreal, the Open Source version 4.7, is able to create very large maps, which is essential in rescue situations. In addition, a

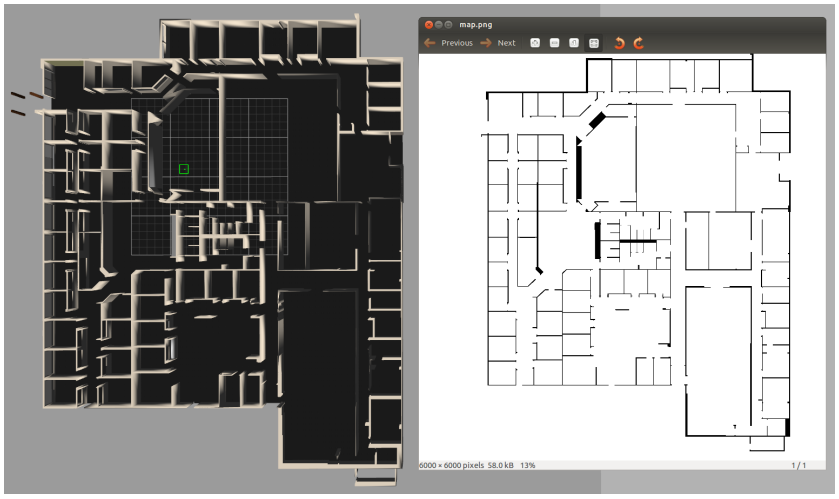

Fig. 2. Screenshot of 3D environment(left) and it's ground truth information(right). Ground truth information is used to calculate received Wi-Fi radio wave power strength between Wi-Fi base station and a robot or between multiple robots

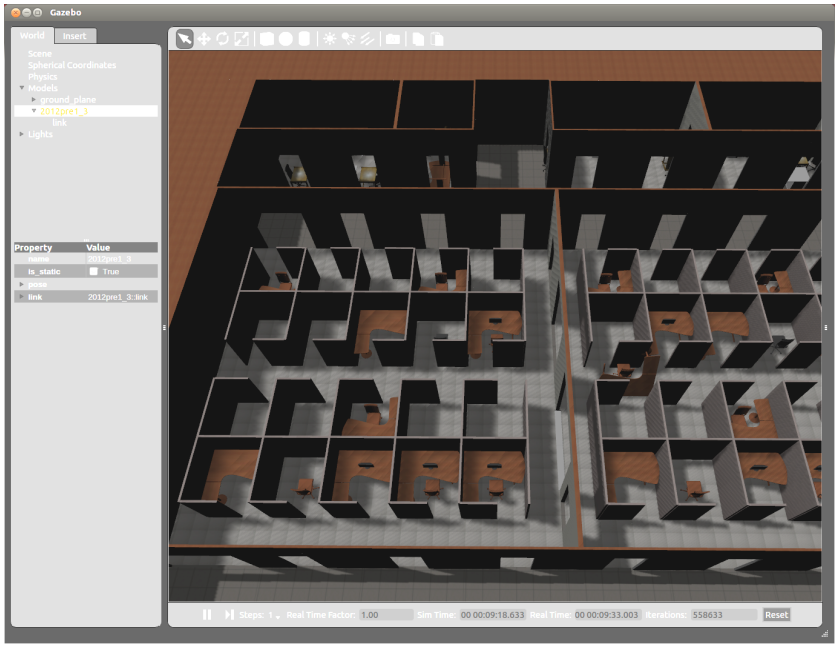

Fig. 3. A screenshot of a world in Gazebo, converted from a USARSim world created in Unreal Editor: RoboCup 2013 Virtual Robot competition Preliminary 1.

lot of effort is spent to create the realistic worlds for the RoboCup Virtual Robot competition. With this method the existing maps can be ported to Gazebo.

5 Future work and conclusion

The first performance is experimented with the Pioneer 3AT; accelerating along a straight trajectory and turning circles. Further exections will be continued with experiments on slopes and trajectories over obstacles based on test environments proposed by the NIST institute. Many RoboCup Rescue teams can also concentrate on tasks of mapping and object recognition and compare their experiment results of rescue tasks.

Our prototype will be presented at the RoboCup 2015 in China and will be open to teams. It will be used as showcase for both the RoboCup and the robotics rescue community as a whole.

References

1. Albrecht, S., Hertzberg, J., Lingemann, K., Nüchter, A., Sprickerhof, J., Stiene, S.: Device level simulation of kurt3d rescue robots. In: Proc. 3rd Int. Workshop on Synthetic Simulation & Robotics to Mitigate Earthquake Disasters (SRMED 2006). Citeseer (2006)
2. Balaguer, B., Balakirsky, S., Carpin, S., Lewis, M., Scrapper, C.: Usarsim: a validated simulator for research in robotics and automation. In: Workshop on "Robot Simulators: Available Software, Scientific Applications, and Future Trends" at IEEE/RSJ (2008)
3. Carpin, S., Lewis, M., Wang, J., Balakirsky, S., Scrapper, C.: Bridging the gap between simulation and reality in urban search and rescue. In: Robocup 2006: Robot Soccer World Cup X, pp. 1–12. Springer (2007)
4. Carpin, S., Stoyanov, T., Nevatia, Y., Lewis, M., Wang, J.: Quantitative assessments of usarsim accuracy. In: Proceedings of PerMIS. vol. 2006 (2006)
5. Carpin, S., Wang, J., Lewis, M., Birk, A., Jacoff, A.: High fidelity tools for rescue robotics: results and perspectives. In: RoboCup 2005: Robot Soccer World Cup IX, pp. 301–311. Springer (2006)
6. Drumwright, E., Hsu, J., Koenig, N., Shell, D.: Extending open dynamics engine for robotics simulation. In: Simulation, Modeling, and Programming for Autonomous Robots, pp. 38–50. Springer (2010)
7. Formsma, O., Dijkshoorn, N., van Noort, S., Visser, A.: Realistic simulation of laser range finder behavior in a smoky environment. In: RoboCup 2010: Robot Soccer World Cup XIV, pp. 336–349. Springer (2011)
8. Koenig, N., Howard, A.: Design and use paradigms for gazebo, an open-source multi-robot simulator. In: Intelligent Robots and Systems, 2004.(IROS 2004). Proceedings. 2004 IEEE/RSJ International Conference on. vol. 3, pp. 2149–2154. IEEE (2004)
9. van Noort, S., Visser, A.: Validation of the dynamics of an humanoid robot in usarsim. In: Proceedings of the Workshop on Performance Metrics for Intelligent Systems. pp. 190–197. ACM (2012)

10. Okamoto, S., Kurose, K., Saga, S., Ohno, K., Tadokoro, S.: Validation of simulated robots with realistically modeled dimensions and mass in usarsim. In: Safety, Security and Rescue Robotics, 2008. SSRR 2008. IEEE International Workshop on. pp. 77–82. IEEE (2008)
11. Quigley, M., Conley, K., Gerkey, B., Faust, J., Foote, T., Leibs, J., Wheeler, R., Ng, A.Y.: Ros: an open-source robot operating system. In: ICRA workshop on open source software. vol. 3, p. 5 (2009)