Proceedings of the ASME 2012 International Symposium On Flexible Automation
ASME/ISCIE 2012
June 18-20, 2012, St. Louis, Missouri, USA

DETC2010/MECH-12345

TITLE GOES HERE

Stephen Balakirsky[*]
Intelligent Systems Division
National Institute of Standards and Technology
Gaithersburg, Maryland 20899
Email: stephen.balakirsky@nist.gov

Zeid Kootbally
Department of Mechanical Engineering
University of Maryland
College Park, Maryland, 20742
and
Intelligent Systems Division
National Institute of Standards and Technology
Gaithersburg, Maryland 20899
Email: zeid.kootbally@nist.gov

ABSTRACT

Abstract goes here (less than 150 words).

INTRODUCTION

Robots have now become a part of many people's everyday lives. Whether as simple toys used by children, floor cleaning robots used in the home, or high precision industrial manipulators used in manufacturing, these systems are quickly changing the ways in which we play and work. One can no longer make the assumption that robots will exist in enclosed areas, or that the programmer or developer of the systems will be highly-skilled robotics experts. Indeed, new open source projects in robotic control systems such as the Robot Operating System (ROS)[1] [**?**] allow anyone with a Linux computer to download and run some of the most advanced robotic algorithms that exist. If the users desire a deeper knowledge of how these algorithms work, there is even a free robotics course from Stanford that may be taken online.

One thing that many of these individuals are missing is robotic hardware. Simulators exist to fill this void and allow both experts and novices to experiment with robotic algorithms in a safe, low-cost environment. However, to truly provide valid simulation, the simulator must provide noise models for sensors and must be validated. One modern robotic simmulator, know as the Unified System for Automation and Robot Simulation (USARSim) [**?**] provides such a simulation platform. This simmulator has been used by the expert robotics community for several years and has played an important role in developing robotics applications, both for rapid prototyping of applications, behaviors, scenarios, and for debugging purposes of many high-level tasks ranging from legged robots playing soccer [**?**] to urban search and rescue [**?**, **?**]. In fact, a search for the keyword "USARSim" on Google Scholar returns over x articles that have referenced the simulation platform.

One reason for the simulation environments popularity is that it enables researchers to focus on algorithm development without having to worry about the hardware aspects of the robots. Simulation can be an effective first step in the development and deployment of new algorithms and provides extensive testing opportunities without the risk of harming personnel or equipment. Major components of the robotic architecture (for example, advanced sensors) can be simulated and enable the developers to focus on the algorithms or components in which they are interested without the need to purchase expensive hardware. This can be useful when development teams are working in parallel or when experimenting with novel technological components that may not be fully implemented or available.

[*]Address all correspondence to this author.

[1]Certain commercial software and tools are identified in this paper in order to explain our research. Such identification does not imply recommendation or endorsement by the authors, nor does it imply that the software tools identified are necessarily the best available for the purpose.

Simulation can also be used to provide access to environments that would normally not be available to the development team. Particular test scenarios can be run repeatedly, with the assurance that conditions are identical for each run. The environmental conditions, such as time of day, lighting, or weather, as well as the position and behavior of other entities in the world can be fully controlled. In terms of performance evaluation, it can truly provide an “apples-to-apples” comparison of different software running on identical hardware platforms in identical environments. Another important feature of a robotic simulator is easy integration of different robotic platforms, different scenarios, different objects in the scene, as well as support for multi-robot applications.

This paper examines a new interface that allows the ROS control framework to communicate directly with USARSim thus opening up sophisticated robot control and development to an entirely new audience. Novice robot developers can now work with world class algoirthms from the safety of their computer without the expense of actual robotic hardware. The basic premise of this paper is to describe the interface connecting the Unified System for Automation and Robot Simulation (USARSim) framework with the Robot Operating System (ROS) framework for manufacturing applications. The following sections describe, analyze and illustrate the new interface for the navigation of mobile robot and robotic arm.

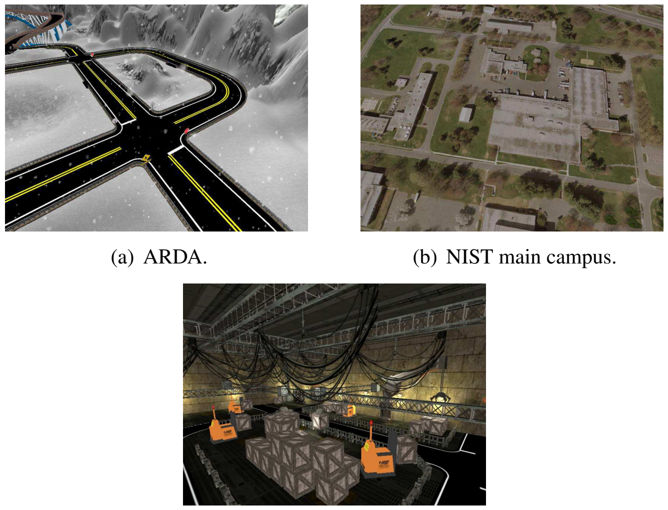
BACKGROUND

An effective interface to multiple robot programs is an important aspect to consider for a multi-robot applications simulation. The effort describes in this paper uses USARSim as a server allowing robot control programs (ROS) to act as clients.

The USARSim Framework

USARSim [?, ?] is a high-fidelity physics-based simulation system based on the industrial game engine Unreal Engine[2]. USARSim was developed under a National Science Foundation grant to study Robot, Agent, Person Teams in Urban Search and Rescue [**?**]. Since Unreal Engine has been deployed for the development of networked multi-player 3D games, it solves many of the issues related to modeling, animation and rendering of the virtual environment.

USARSim utilizes the Karma Physics engine [**?**] and high-quality 3D rendering facilities of the Unreal game engine to create a realistic simulation environment that provides the embodiment of a robotic system. The current release of USARSim consists of various environmental models, models of commercial and experimental robots, and sensor models. High fidelity at low cost is made possible by building the simulation on top of a game engine. By loading the most difficult aspects of simulation



(a) ARDA. (b) NIST main campus. (c) Factory.

FIGURE 1. Sample of 3D environments in USARSim.

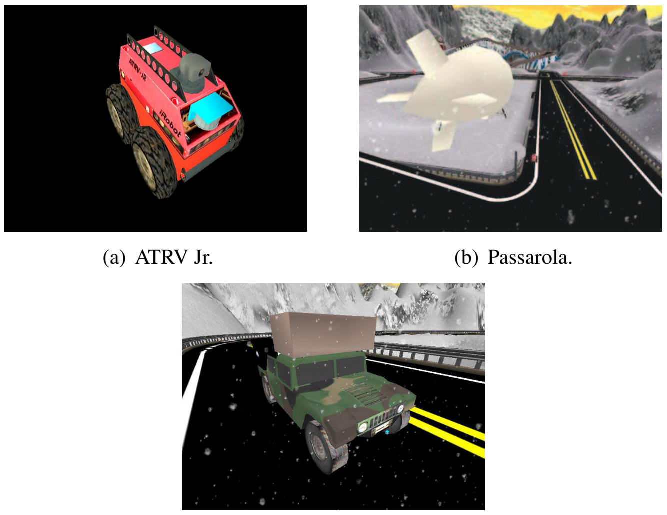
to a high volume commercial platform which provides superior visual rendering and physical modeling, full effort can be devoted to the robotics-specific tasks of modeling platforms, control systems, sensors, interface tools and environments. These tasks are in turn, accelerated by the advanced editing and development tools integrated with the game engine leading to a virtuous spiral in which a wide range of platforms can be modeled with greater fidelity in short time.

USARSim was originally based upon simulated environments in the USAR domain. Since then, USARSim has been used worldwide and more environments have been developed for different purposes. In addition to USAR, the simulator has been applied to the DARPA Urban Challenge (see Figure **??**). Other environments such as the NIST campus (see Figure **??**) and factories (see Figure **??**) have been used to test the performance of algorithms in different efforts [**?**, **?**, **?**].

USARSim was initially developed with a focus on wheeled robots, in particular differential drive systems. Interest and wide community support offer multiple robots, including underwater vehicles, legged platforms, and humanoids. In USARSim, robots are based on the real robots and are implemented by specific classes, thus making it easier to develop new platforms that model custom designs. Three base classes model different kinds of wheeled locomotion, namely differential drives (Figure **??**), omnidirectional vehicles (Figure **??**) and Ackerman steered vehicles (Figures **??**).

All robots in USARSim have a chassis, multiple wheels, sensors and effecters. The robots are configurable (specify types of sensors/effecters for example). The properties of the robots can also be configured, such as the battery life and the frequency of data transmission.

[2]http://www.unrealengine.com/



(a) ATRV Jr. (b) Passarola. (c) NIST HMMWV.

FIGURE 2. Sample of vehicles in USARSim.

The ROS Framework

ROS[3] is an open source framework designed to provide an abstraction layer to complex robotic hardware and software configurations. ROS delivers libraries and tools to help software developers create robot applications. ROS has been used in many robotic applications such as Willow Garage[4] Personal Robots Program [**?**] and the Stanford University[5] STAIR project [**?**].

ROS possesses a large range of tools and services that both users and developers alike can benefit from. The philosophical goals of ROS include an advanced set of criteria and can be summarized as: peer-to-peer, tools-based, multi-lingual, thin, and free and open-source [**?**]. Furthermore, debugging at all levels of the software is made possible with the full source code of ROS being publicly available. Thus, the main developers of a project could benefit from the community and vice-versa.

Nomenclature The fundamental concepts of the ROS implementation are nodes, messages, topics, and services. These terms will be used throughout the rest of the paper and are detailed below [?].

- Node: An executable unit which communicates with other nodes. ROS is designed to be modular at a fine-grained scale: a system is typically comprised of many nodes. In this context, the term "node" is interchangeable with "software module". Nodes communicate with each other by passing messages.
- Message: A strictly typed data structure. Standard primitive types (integer, floating point, boolean, . . .) are supported, as are arrays of primitive types and constants. A node sends a message by publishing it to a given topic.
- Topic: A communication channel between two or more nodes. A node that is interested in a certain kind of data will subscribe to the appropriate topic. There may be multiple concurrent publishers and subscribers for a single topic, and a single node may publish and/or subscribe to multiple topics.
- Service: A remote procedure call defined by a string name and a pair of strictly typed messages: one for the request and one for the response.
- Stack: Packages in ROS are organized into ROS stacks. Whereas the goal of packages is to create minimal collections of code for easy reuse, the goal of stacks is to simplify the process of code sharing. Stacks are the primary mechanism in ROS for distributing software. Each stack has an associated version and can declare dependencies on other stacks. These dependencies also declare a version number, which provides greater stability in development.

THE INTERFACE

Sensor Interface

Mobile Robot Interface

The control of mobile robots in USARSim is performed via the Navigation stack in ROS. The Navigation stack is a 2D navigation stack that takes in information from odometry, sensor streams, and a goal pose and outputs safe velocity commands that are sent to a mobile base.

Robotic Arm Interface

SETUP AND RUN THE INTERFACE

CONCLUSION AND FUTURE WORK

[3]http://www.ros.org/wiki/
[4]http://pr.willowgarage.com
[5]http://stair.stanford.edu