Implementation of Semi-Virtual Multiple-Master/Multiple-Slave System

Boris Gromov[1] and Jee-Hwan Ryu[2]
BioRobotics Laboratory
Korea University of Technology and Education (Korea Tech)
Cheonan-City, 330-708, Republic of Korea
[1]boris.gromov@kut.ac.kr, [2]jhryu@kut.ac.kr

***Abstract*— Building an experimental robotic setup can be a very tedious, prone to hardware faults and expensive process. A common way to circumvent some of this problems is to model a part or entire system in software. Moreover, virtual environments can be the only option to model hazardous or inaccessible sites. However, implementation of teleoperated robotic systems with force feedback exhibits additional problems. The human-robot interfaces should exist in hardware and this in turn requires simulated system to work in a proximity with a real-time.**

In this paper we describe a successful implementation of such a system in Gazebo simulator and Robot Operating System. We built an experimental Multiple-Master/Multiple-Slave setup in virtual environment for peg-in-hole task that consists of two 7-DOF Schunk LWA-3 robots and a common object to manipulate. To display forces to operators two SensAble PHANToM devices were utilized.

***Keywords*— cooperative teleoperation, multiple-master multiple-slave, semi-virtual teleoperation, gazebo, robot operating system.**

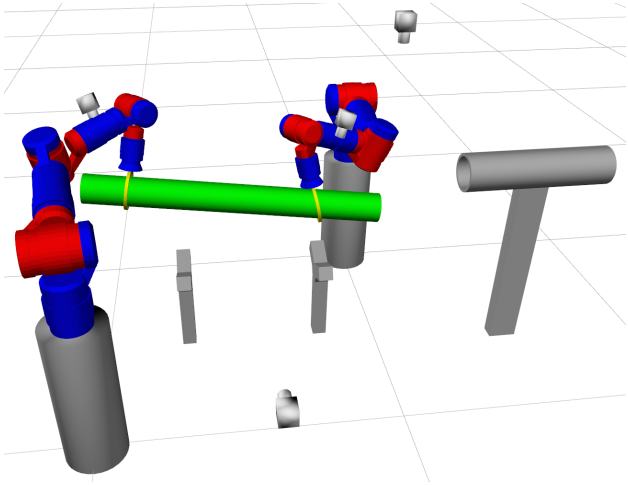

Fig. 1. Overview of experimental setup for MMMS system in Gazebo.

1. Introduction

Modeling a teleoperated robotic system can be a daunting task, which may be impossible to perform by a single researcher or even by a small research group. Owing to a wide variety of modern robot simulation and development tools [1] as well as to inexpensive computing power, this task is not longer a problem.

The goal of this work to provide a general experimental setup for the Multiple-Master/Multiple-Slave (MMMS) systems, that would serve as a platform for experimental validation of diverse control algorithms in cooperative teleoperation domain.

As a case study we implemented some of our ideas from previous works. Particularly the concept of Virtual Master/Slave [2] with the Field of View Deficiency-based (FoVD) control decomposition [3].

2. Simulation Environment

Keeping in mind real-world applications and the fact that it is necessary to unify program interfaces with a real hardware we chose the Robot Operating System (ROS) as underlying platform for our system.

ROS is not an operating system in its pure sense, but a software framework that provides abstraction of various hardware components, such as sensors and actuators, by means of structured communication layer above the hosting operating systems [4].

In the following list we will highlight some of the concepts used by ROS. Please refer to ROS Wiki[1] to see a more detailed explanation.

- *Nodes* are the run-time processes. Node can be a pure computation process, e.g. calculating inverse kinematics of a robot, or a driver for a sensor or actuator. Usual robot setups consists of many nodes.
- *Messages* are the data structures comprising typed fields. Nodes communicate with each other by means of messages. A message can consists both of basic data types (integer, floating point, boolean etc.), arrays and data structures themselves.
- *Topics* are named and strictly typed message buses. Any node can publish/subscribe to a topic and send/receive messages as long as they are of a right data type.
- *Packages* are the main unit for organizing software in ROS on the level of file system. A package may contain nodes, plugins (shared libraries), datasets, configuration files, or anything else that is usefully organized together.
- *Stacks* are collections of packages that provide an aggregated functionality, for instance, navigation or image processing algorithms and routines.

[1]http://www.ros.org/wiki/ROS/Concepts

This paper was supported by the project (Development of a Tele-Service Engine and Tele-Robot Systems with Multi-Lateral Feedback) funded by the Ministry of Trade, Industry & Energy of Korea.

As of the year 2010, the number of packages in ROS officially exceeded one thousand. At the same time many of the released packages are published under permissive open-source BSD-license, which allows to use the code in commercial projects without exposing your own proprietary solutions.

Due to its very abstract interfaces, ROS, is very successful in integrating third-party libraries, toolkits and frameworks. And one of the great toolkits integrated is the Gazebo – an open-source multi-robot simulator for outdoor environments [5, 6].

There is a variety of plugins to interface ROS and Gazebo. These plugins run in Gazebo memory space and expose ROS-compatible interfaces via topics and services, and thus, appear to ROS as the regular nodes. For example, Gazebo camera plugin has the same interfaces as any ROS camera driver, i.e. `sensor_msgs/Image`, `sensor_msgs/CameraInfo` etc. That allows for simulated robots and sensors appear to ROS as if it would be a real hardware.

Amongst the available sensors there are mono and stereo cameras, laser scanners, sonars, force-torque sensors etc. Additionally to existing modules users can integrate their own plugins that would simulate custom sensors.

All of the aspects of simulation in Gazebo are described by XML-files according to Simulation Description Format (SDF). Two types of the files are mainly in use:

- *World* files – describe properties of simulated world, i.e. general settings of the physics engine such as gravity vector, time step, simulation frequency etc.; visualization settings, e.g. color of ambient light, shadows; objects – models of the robots, static parts of environment, light sources etc. And finally a description of plugins to be attached to a world.
- *Model* files – describe simulated models (robots), i.e. their kinematic, dynamic and visual properties as well as collision geometry and sensors.

Basically the whole world with all its objects can be defined inside a world file. However in order to reuse the models developed by others it is a good idea to put them in the separate model files. Then the latter have to be included in a world file by means of `<include>`-tag.

3. System Architecture

The basic simulated MMMS system consists of two master devices and two virtual slave robots. Master devices are represented by SensAble PHANToM Omni and PHANToM Premium 3.0 force displays, while the slaves are by the virtual models of Schunk LWA-3 robots.

As a result of a high popularity of ROS in the robotics research community most of the components that are necessary to implement in our system are already there. For instance, one of the key components in our simulation – a model of Schunk LWA-3 manipulator is available from Care-O-Bot stack and can be easily adopted with minor modifications.

However, Gazebo model itself is just a set of rigid links connected by joints. To turn them to actively controlled actuators one should design an appropriate controller.

Here, it is very easy to see the demarcating line between 'bare hardware' (Gazebo) and controlling software (ROS). The model of manipulator presents a passive articulated object with a set of dynamic properties such as inertia, mass etc., which is unable to move by itself. In general a real manipulator would have the same properties and therefore control algorithms developed on the base of simulation can be transferred to it without a big change.

To simplify development of the robot controllers ROS provides a special interface by means of PR2 Controller Manager (CM). Despite its name CM can be used with any robot that is possible to define with Unified Robot Description Format (URDF) and not only with Willow Garage PR2.

PR2 Controller Manager is a vital component of the system. It builds abstraction layer over hardware actuators whether it is a real actuator or a simulated one. While the `pr2_controller_manager` package provides the infrastructure to run controllers in a hard realtime loop on a real robots[2] the `GazeboRosControllerManager` provides the same interface for controllers in simulation[3].

Our implementation of the system in Gazebo and ROS consists of the following components:

- *Sensable Phantom node*: provides interface to haptic force display via SensAble OpenHaptics HD API.
- *Teleoperation node*: provides the mapping of the local coordinate frame of a haptic device to a global frame of the virtual environment. This node is in charge of the motion scaling and indexing as well as switching the modes of teleoperation, e.g. 'solo-control' and 'cooperative control' of the virtual slave.
- *Virtual Slave node*: presents the multiple slaves as a single slave to the master site. Provides control decomposition routines.
- *GazeboRosControllerManager plugin*: provides CM interface for JT Cartesian Controller in Gazebo.
- *JT Cartesian Controller*: calculates inverse dynamics of the robot with a null-space optimization.
- *FT Sensor plugin*: provides ROS interface for forces and torques calculated by physics engine in simulation.
- *Camera plugin*: simulates the cameras located in a virtual environment and provides ROS interface for image visualization tools.

To interface various PHANToM devices with ROS we generalized and refactored the `phantom_omni` package that was originally developed by Healthcare Robotics Lab at Georgia Tech[4].

Intercommunication of numerous components of the system is outlined in Figure 2. However, for the clarity of explanation the scheme shows only one single-master/single-

[2] `http://ros.org/wiki/pr2_controller_manager`

[3] `http://ros.org/wiki/pr2_gazebo_plugins`

[4] Modified package is available at: `https://github.com/brl-koreatech/sensable_phantom.git`

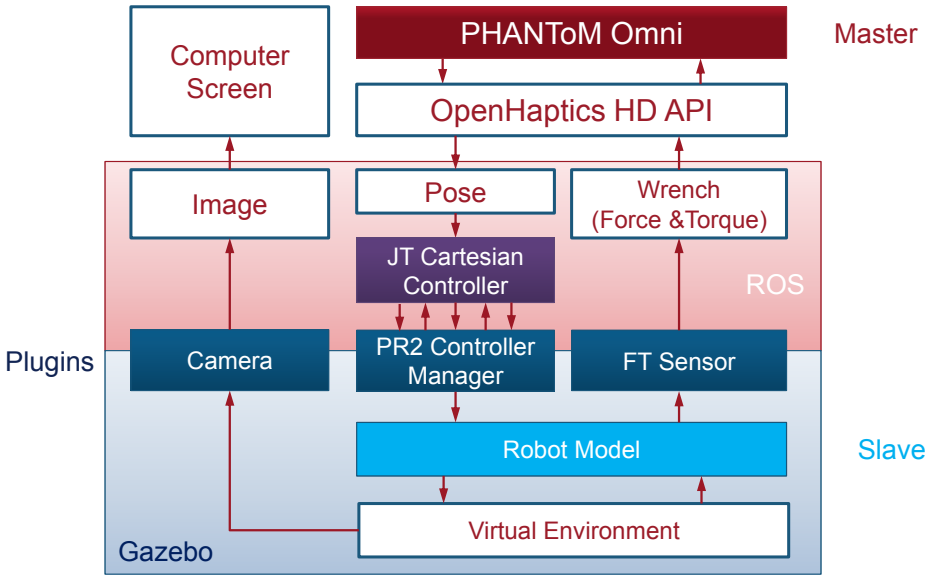

Fig. 2. Intercommunication of Master and Slave in terms of Gazebo and ROS.

slave block, the actual implementation consists of two such blocks.

4. Experimental Setup

In order to test developed system we created a Gazebo world consisting of two robots with simple ring 'grippers', a movable pipe (green) on a stand and another static pipe (grey) of bigger diameter. For operators to feel the interaction forces the force-torque sensors were placed between the flanges and the grippers of the robots (Fig. 1).

Depending on a particular task operators are allowed to either use a 3D view of Gazebo GUI or the cameras. There are two cameras rigidly attached to the last (7^{th}) link of each manipulator. Note, however, that in actual simulation the brackets are not visualized in order to keep the 3D view less polluted. Another two cameras – showing an overview of the workspace from the side and from the top – are statically positioned in the environment.

A common for teleoperation systems peg-in-hole task is set as the goal. Particularly, two operators are required to lift the movable pipe together and then feed it to the static pipe. The task is designed in such a way that single-operator/single-robot would not be able to perform it on his/her own. Namely the load capacity of the single robot is 10 kg and the weight of the movable pipe is 15 kg. Additionally the ring gripper has a diameter that exceeds the one of the pipe and thus guarantees a loose-fit.

Therefore, to successfully perform such a task operators should coordinate their motions and keep the pipe in parallel to the ground while lifting it and coaxial with target static pipe while inserting it.

The main application of described experimental setup is evaluation of different algorithms developed in our lab that are aimed to increase the performance of Multiple-Master/Multiple-Slave systems.

To test the experimental setup we considered two control decomposition algorithms. First, the Virtual Slave concept [2] – an approach when control inputs of several masters and force feedbacks of several slaves are modified by intermediate block, such that multiple slaves appear to operators as a single slave, and thus, conventional control decomposition approaches [7, 8, 9] maybe utilized. In the frame of the first concept, the second approach we considered is the Field of View Deficiency-based control decomposition [3] – this concept suggests to decompose the control of several operators based on their corresponding inability to perceive a remote environment through a 2D view, i.e. image from a monocular camera.

5. Results and Conclusion

In general experimental setup based on Gazebo and ROS has shown good results. However one of the major problems we faced was the significant drop of performance (in terms of coherence with wall-clock time) during dynamic interaction of several objects. Thus, a reported by Gazebo *real-time factor* of initial environment, i.e. initial poses of the robots do not produce any dynamic interactions, was ≈ 0.96. During the pipe lifting stage by both robots real-time factor dropped to ≈ 0.6. And the worst performance was shown during the pipe inserting stage (≈ 0.45).

Although the divergence of wall-clock and simulated-clock time may be treated as a time-delay (a property of any real teleoperation system) it poses additional difficulties which we originally were trying to avoid by using simulation.

To summarize, following are the pros and cons of the developed system.

A. Pros

Free of charge software. Most packages in ROS are available under permissive BSD-licence, while Gazebo is distributed under Apache Licence v2.0. Thus the system can be used in commercial applications.

Rapid prototyping. The large number of very generic ROS packages allows to rapidly build a prototype of a system. Particular subsystems later can be replaced with more sophisticated versions without changing the interface.

Compatibility with real-world robots. Since the system utilizes conventional ROS interfaces the simulated environment can be replaced with the real teleoperation system with very little effort.

B. Cons

Not real-time. Although certain ROS packages, e.g. `pr2_controller_manager`, are able to work in the hard real-time loops, the simulation subsystem, i.e. Gazebo, is not designed for it. That introduces a variable time-delay, which is also depends on the complexity of simulated environment.

Difficult to tune physics engine parameters. The version of Gazebo that we used in our tests still utilize legacy Open Dynamic Engine (ODE). The main problem of ODE is that the parameters are not very intuitive to tune and the engine itself is not actively developed. The newer versions of Gazebo are promised to support Bullet physics engine, which should help to solve some of the issues and also bring new features such as soft body contacts and GPU-accelerated calculations.

Nevertheless, developed system possesses the great potential that mitigates most of the problems we mentioned above.

Open-source packages we used in this work (Gazebo and Robot Operating System) are very rapidly developing software with a great community support. For instance, the version of Gazebo we started to use at the beginning of the year 2013 was v1.5 and right now (September 2013) Gazebo v1.9 is available for download.

So hopefully most of the performance issues will be rectified in upcoming versions.

References

[1] J. Kramer and M. Scheutz, "Development environments for autonomous mobile robots: A survey," *Autonomous Robots*, vol. 22, no. 2, pp. 101–132, 2007.

[2] B. Gromov and J.-H. Ryu, "Supervisory model-mediated teleoperation for multiple-master/multiple-slave system," in *Ubiquitous Robots and Ambient Intelligence (URAI), 2012 9th International Conference on*, pp. 108–110, 2012.

[3] B. Gromov, G. Ivanova, and J.-H. Ryu, "Field of view deficiency-based dominance distribution for collaborative teleoperation," in *Control, Automation and Systems (ICCAS), 2012 12th International Conference on*, pp. 1990–1993, 2012.

[4] M. Quigley, B. Gerkey, K. Conley, J. Faust, T. Foote, J. Leibs, E. Berger, R. Wheeler, and A. Ng, "ROS: an open-source Robot Operating System," in *ICRA'09 Workshop on Open Source Software*, 2009.

[5] N. Koenig and A. Howard, "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 vol.3, 2004.

[6] N. Koenig and J. Hsu, "The many faces of simulation: Use cases for a general purpose simulator," in *ICRA'13 Workshop on Developments of Simulation Tools for Robotics & Biomechanics*, 2013.

[7] S. Katsura, T. Suzuyama, and K. Ohishi, "A realization of multilateral force feedback control for cooperative motion," *Industrial Electronics, IEEE Transactions on*, vol. 54, pp. 3298 – 3306, dec. 2007.

[8] L. Liu, G. Liu, Y. Zhang, W. Guo, K. Lu, and M. Zhou, "Separate DOF control and mutual guidance in networked haptic collaboration maze game: Design and evaluation," in *Robotics and Automation (ICRA), 2011 IEEE International Conference on*, pp. 913 – 918, may 2011.

[9] P. Malysz and S. Sirouspour, "A kinematic control framework for single-slave asymmetric teleoperation systems," *Robotics, IEEE Transactions on*, vol. 27, pp. 901 – 917, oct. 2011.