Softbody Simulation in Gazebo for Satellite Servicing

Hudson Thomas* and Simon Leonard

Lab for Computational Sensing and Medical Robotics,

Johns Hopkins University, Baltimore, MD

(Dated: August 18, 2016)

Abstract

The purpose of this project was to incorporate soft body simulation provided by the Bullet physics engine into the Gazebo simulation environment for a satellite servicing robot. The tasks of simulating and rendering the soft body were divided into two separate Gazebo plugins. These plugins, along with minor modifications to the gazebo source code, enabled us to simulate and render a realistic soft body that could interact with native Gazebo objects as well as with physical forces like gravity. This capability may also have applications in other fields like surgical robotics simulation.

I. INTRODUCTION

A. The Problem of Satellite Servicing

Satellites have revolutionized what human society can do as well as what it knows. In the coming age of cheap space flight, the technological and scientific benefits that satellites provide will only increase at a faster and faster pace.

However, the capabilities of satellites will always be limited by their cost. Millions if not billions of dollars are invested into each of these machines which can only operate until it collides with space junk or runs out of fuel. In the past, some high value satellites like the Hubble have been repaired via space-walks, but even these high-risk missions are no longer possible due to the termination of the space shuttle program.[1] The goal of NASA's RestoreL mission is to re-enable satellite servicing using robots.

One of the major technological challenges associated with robotic satellite servicing is the telemetry delay that occurs between the robot operator and the on-orbit servicing robot. This delay (1-7 seconds) greatly decreases the performance of the telerobotic task.

B. The Problem of the Time delay

The goal of the satellite servicing research at JHU is to "develop new methods for telerobotic on- orbit servicing of spacecraft under ground-based supervisory control of human operators to perform tasks in the presence of uncertainty and telemetry time delay of several seconds."[2]

The general setup of the lab's teleoperation solution to satellite servicing is shown in Fig. 1. The problems of uncertainty and telemetry time delay are addressed by each of the three components of the setup: the human operator uses a da Vinci surgical robot master controller for intuitive and precise robot control; the on-orbit servicing robot uses dynamic force sensing and image processing to check for errors in real time; and the simulation displays a live model of the satellite-robot system to eliminate the effects of the time delay from the operator's commands. As shown in the diagram, the pose information of the satellite-robot system in the simulation is sent to the robot which acts to match the real environment to the environment in the simulation.

The accuracy of the simulation is vital for minimizing unforeseen errors and increasing

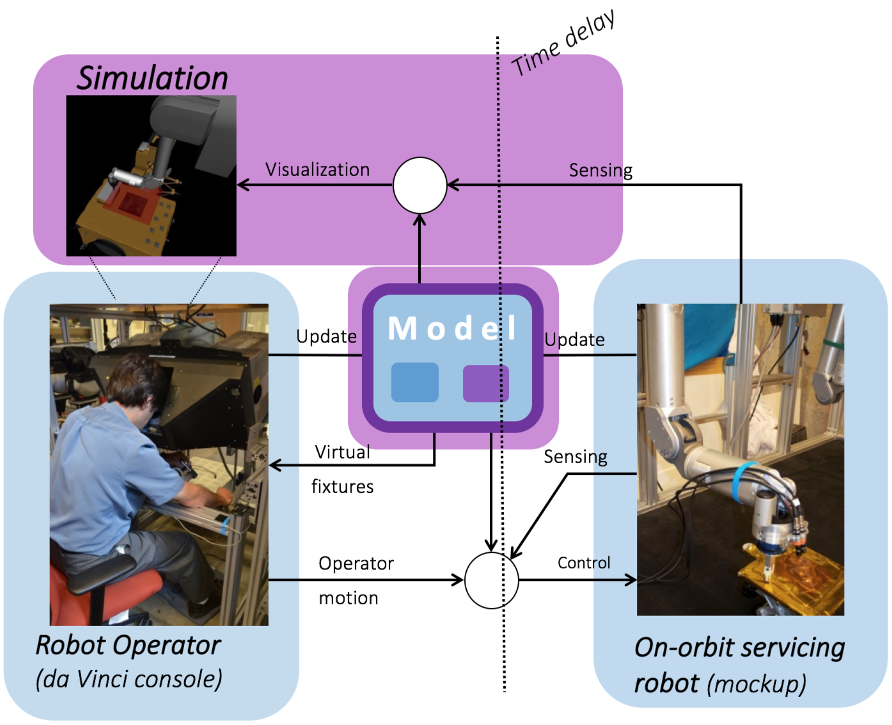

FIG. 1. The telerobotic satellite servicing project setup used by the Teleoperation Lab at JHU has three main parts, the robot operator on the ground, the on-orbit servicing robot, and the simulation with which the robot operator interacts.

precision. The goal of the project addressed in this paper was to further develop the realness of the simulation by adding the ability to model non-rigid objects (or soft bodies). Soft body simulation is important for modeling several parts satellite-robot system, including the soft mulit-layer insulation on the outside of the satellite (which needs to be cut for refueling and repair), as well as the refueling hose (which needs to not be cut accidentally during robot operation).

C. The problem with Gazebo

We used the Gazebo simulation software for our robot-satellite model, but although Gazebo is very useful for most robot simulation applications because of its support for ROS, it is limited by the fact that it cannot yet simulate deformable objects.

Gazebo has two main parts, the server and the client. The server processes the physics of the environment, which involves calculating the full state of each body involved in the simulation taking into account the forces and velocities on each object as well as any additional data inputted from external controllers. The client receives pose data in messages from the server and displays the environment in a neat graphical user interface that adds convenient functionality and visualization.

The server allows for the use of several different physics engines, each of which is optimized for different kinds of simulations.[3] We chose to use the Bullet physics engine because it can simulate deformable objects. However, while Gazebo supports Bullet , it does not yet support Bullet's soft body libraries specifically. Gazebo does, however, support third party plugins that allow for hard coded customization. We used these plugins to add Bullet soft body functionality to Gazebo.

D. Soft Bodies

Bullet simulates soft bodies as collections of nodes, links, and faces. Nodes contain the pose, mass, force, and velocity information of each vertex in the soft body. The links act as springs between the nodes and allows for customization of deformation constants like bending and stretching. The faces give the soft body surface area allowing it to interact with fluid forces like wind.

Collision shapes are commonly used in simulations to allow the user to simplify the simulated object in order to cut back on computational strain. Bullet processes rigid body collisions by determining the proximity of each body's collision shape. However, this shape is determined at compile-time, which is problematic for soft objects because their shape changes during the simulation.

Bullet handles collisions between a soft body and another body by iterating through the soft body's nodes any checking if any of them are in contact with the other body's collision object.

II. PROJECT DESIGN

We decided to follow Gazebo's two part model by adding two gazebo plugins to the simulation in order to handle the physics and collisions of the soft body as well as its visualization. The project setup is shown bellow in FIG. 2.

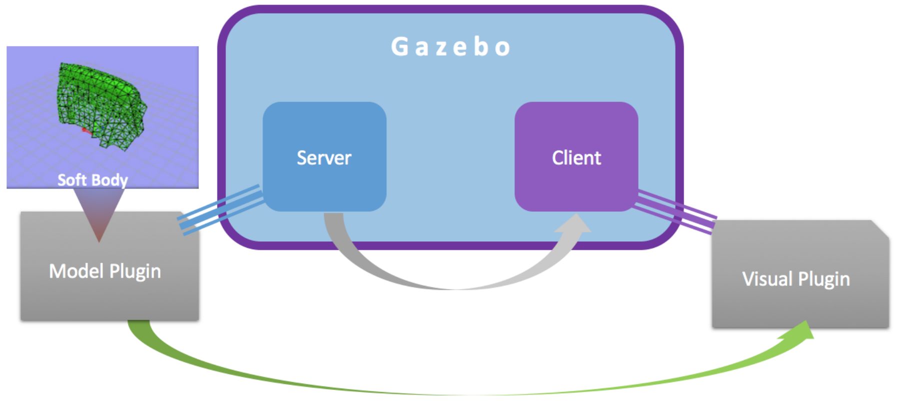

FIG. 2. The overall structure of the project: the model plugin handles the physics and collision of the soft body and the visual plugin handles the graphics

The model plugin is attached to an object already in the Gazebo world. This object is very useful because it connects the Gazebo world and the soft body that is hard coded into the plugin. (We acknowledge that having an extra object is not ideal because it adds computational strain on the simulation. In our project, we minimized this by making it very small (not graphically intensive) and by eliminating its collision element so it doesn't unintentionally collide with another object.) We were then able to create a Bullet soft body in the plugin and add it directly to the Gazebo world to allow for collision with other objects. The model plugin then copies the location of each of the nodes in the soft body into a Gazebo message and sent it to the visual plugin.

The visual plugin was attached to the visual component of the same object as the model plugin. In the same way that the model plugin needs this connection to get access to the physical simulation, the visual plugin needs this connection to access the pre-rendering event necessary for adding graphics. Because Gazebo does not have graphic support for deformable objects, we created a mesh in Ogre and added it to the 3D space Gazebo was using to display the other objects. The visual plugin then updated the position of each of the nodes in the mesh according to the node poses sent in the message from the model plugin. The message update callback function that was used to read the position of the soft body from the model plugin was not called contemporaneously with the pre-rendering event, so we used an ordering variable to ensure that the pre-render function was only called once the message had been completely read from the model plugin in order to avoid ”index out of bounds” errors.

In addition to these plugins, minor changes had to be made to the Gazebo source code to enable soft-rigid collision. The entire project as well as these changes to Gazebo will be made public in our Git repository.

With this setup we expected the soft body to be able to interact with other native rigid gazebo objects as well as other soft bodies, and to be affected by physical forces like gravity and electromagnetism. We predict that these characteristics of the soft body would make it sufficiently realistic for the satellite servicing simulation.

III. RESULTS

We were able to create a visually realistic simulation of a cloth that could interact with native gazebo objects as well as with gravity. Snapshots of the simulation are shown below in FIG. 3.

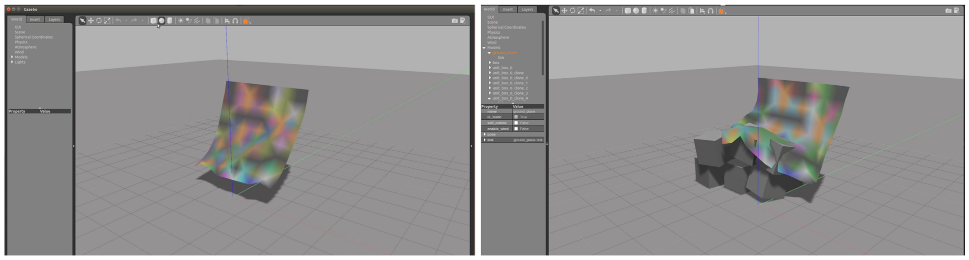

FIG. 3. The snapshot on the left shows the Bullet soft body that we created simply swinging due to the effects of gravity. The snapshot on the right shows the soft body colliding with rigid cubes native to gazebo.

IV. FUTURE WORK

One of the problems with our simulation (apparent in the rightmost image in FIG. 3) is that, although the major collision bodies are behaving physically, there are also places where some corners of the cubes penetrate the cloth. This is most likely due to the relatively coarse resolution of the nodes in the soft body and of the vertices in the cube, and we expect to be able to solve this problem with increased node/vertex count.

A second goal for future work is to be able to model other types of soft bodies beyond a simple cloth. We predict that the model plugin should be able to support the other types of soft bodies provided by Bullet like tetrahedral meshes and point clouds. However, because the Gazebo doesn't support visualization of any of these bodies, it will by necessary to create new visual plugins for each of these types of objects or a single more general plugin that can handle all types of Bullet soft bodies. Support for these other types of soft bodies may be useful for modeling volumetric soft bodies like the refueling hose in a satellite refueling mission and even soft tissue in a robotic surgery application.

In addition to creating visual support for volumetric soft bodies, we also plan to develop cutting methods for simulating cutting the soft multi-layer insulation surrounding the satellite. We have already made some progress in this direction, and are testing several different cutting methods to maximize realism and minimize computational effort.

V. CONCLUSION

We have demonstrated the ability to realistically model and visualize non-rigid bodies in Gazebo. This capability will increase the accuracy of the satellite-robot simulation, hopefully decreasing the probability for unforeseen errors during satellite servicing. More effective robotic satellite servicing will decrease the overall cost of maintaining a satellite and hopefully encourage future satellite innovation, helping us to continue to revolutionize our society and learn more about our world and the universe beyond.

ACKNOWLEDGMENTS

This work is supported by NSF grant #EEC-1460674. The authors gratefully acknowledges Jonathan Bohren and the rest of the Teleoperation Lab at Johns Hopkins University for their help and support during this project.

---

* hudson.thomas@my.wheaton.edu

[1] "On-orbit satellite servicing study project report. Technical report" NASA Goddard Space Flight Center, `https://ssco.gsfc.nasa.gov/images/NASA_Satellite\%20Servicing_Project_Report_0511.pdf`, 2010.

[2] Tian Xia, Simon Leonard, Isha Kandaswamy, Amy Blank, Louis L. Whitcomb and Peter Kazanzides, "Model-Based Telerobotic Control with Virtual Fixtures For Satellite Servicing Tasks" *2013 IEEE International Conference on Robotics and Automation (ICRA 2013)*, 1479 - 1484, May 2013.

[3] Steven Peters, John Hsu ,"Comparison of Rigid Body Dynamic Simulators for Robotic Simulation in Gazebo", `http://www.osrfoundation.org/wordpress2/wp-content/uploads/2015/04/roscon2014_scpeters.pdf`, 2014.