MULTI-BODY SIMULATOR USER'S GUIDE

Paul R. Schmitt
John E. Hogan
Xiao Wu
Wayne J. Book

Georgia Institute of Technology
The George W. Woodruff School of Mechanical Engineering
Atlanta, Georgia

Sponsored by
The Westinghouse Savannah River Plant

Feb. 5th, 1995

Table of Contents

Introduction– What is a Multi–Body Simulator?

A Multi–Body Simulator (MBSim) is an object–oriented system that models, simulates, and animates the kinematics and dynamics of robotic arms and vehicles. This system creates a three dimensional graphical environment which can be used as a powerful tool in robotic design and control. In addition, MBSim incorporates rangefinder as well as ultrasonic sensor model to yield environmental feedback.

In fact, a motivation for developing MBSim stemmed from the realization that most commercially available packages are not as flexible as is often needed. For example, while it is rare for a typical system to handle industrial robots AND vehicles, it is quite uncommon for them to be able to handle sensors as well as a mechanism's kinematics and dynamics.

To give you an idea of where MBSim is heading, our ultimate goal for MBSim is studying safe mechanism motion through sensor–based motion control. We will be testing new motion planning and control algorithms on numerous mechanisms in their typical environments. This is especially important for vehicles with car–like steering in highly constrained environments (e.g., parallel parking a Cadillac).

This document gives an overview of MBSim's capabilities and describes in detail how one utilizes these capabilities. First an explanation of our joint paradigm, or mechanism construction, is given. From there we give an overview of sensors, sensor data storage array (i.e., map), and dynamics in MBSim. Following this, there is a detailed description of what script files are and how a user formulates the script files for an application. Then an explanation of how the user can determine the kinds of forces and torques needed for a desired motion (displaying force–torque plots) is given. An example program is given and explained followed by a description of additional developments currently being worked upon.

Main MBSim Features Overview

This Multi−Body Simulator seeks to model mechanisms with sensors traveling in an often unknown environment. Each mechanism is constructed by a series of links. Each link can get information based on sensor reading. Sensor data is stored in maps which keep track of environment information and gives good approximation of obstacle locations. With Newton−Euler dynamic formulation, MBSim is able to do inverse dynamic calculation as well as force and torque control.

JOINT PARADIGM

In MBSim, each mechanism is constructed as a series of links. Each link incorporates the joint kinematics that connect it to the preceding link. In other words, each link "knows" how to move itself with respect to the previous link. This general joint paradigm allows holonomic and non−holonomic constraints to be treated similarly. This paradigm also facilitates modeling and simulation of the kinematics and dynamics of multibody mechanisms with a wide variety of link types. The links contain graphic objects (See GRAPHICS DESCRIPTION below) which describe the physical shape of the link and sensors which provide feedback of the environment.

SENSORS

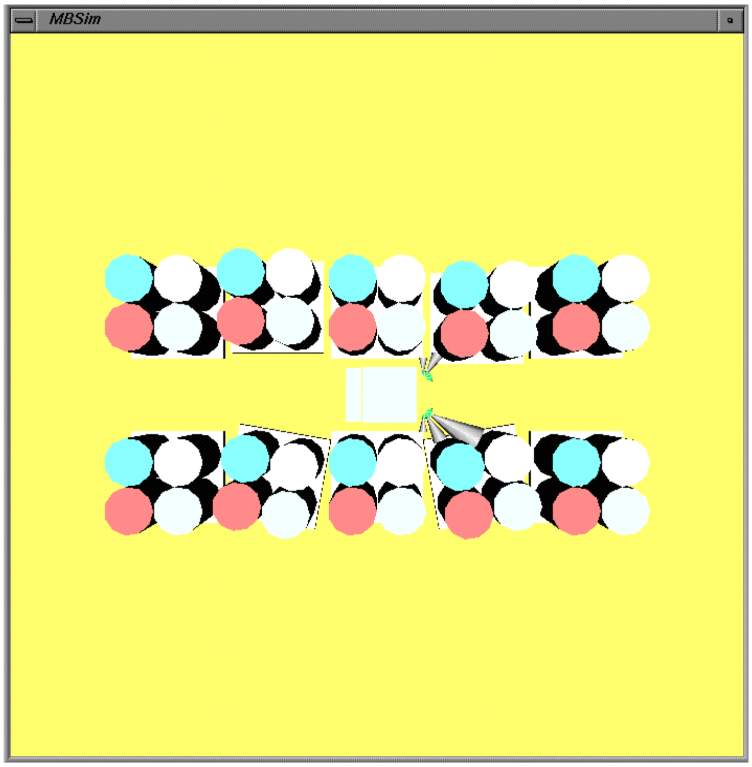
MBSim incorporates an idealized rangefinder and ultrasonic model. When activated, the Rangefinder sensor determines whether the objects' enclosing spheres lie on the sensor's line of sight. In addition to checking for the intersection of the sound cone with an object, Ultrasonic checks the angle between the sound wave and the object's surface. A reflection will return to the transmitting sensor only if the incident angle is nearly normal, and the object is not shielded by another object. Ultrasonic works for cylinder with verticle axes and rectangular parallelcpipids with verticle edges corresponding to drums on pallets. The sensor axis is assumed to be in a horizontal plane. See "Ultrasonic Sensor Model and Configuration Simulation for Mobile Robot Navigation in Narrow Aisles" by Hogan, Schmitt and Book (Contract Report) dated Dec 9, 1994 for more information.

Through a sensor description file, the user places any number of sensors in any position or orientation on any link. See SENSOR DESCRIPTION for more details.

MAP

Map is an intelligent range data storage array [1]. It is designed to "move" along with the mechanism, automatically storing sensor data (Rangefinder and Ultrasonic) from pinged sensors. Range data is added to previous data. In this way collective range data is held to yield a good estimation of obstacle locations in the environment. The user tailors the size of a map and the number of sensors utilized as needed through a map description file. See MAP DESCRIPTION for more details.

The data in a map can also be used to construct and display a Histogram. Histogram accumulates data in a display that does not move with the vehicle. The color of a cell in the histogram represents the cumulative sensor readings of a point fixed in space. Figure 1B shows a histogram display of the robot moving in environment of Figure 1A..



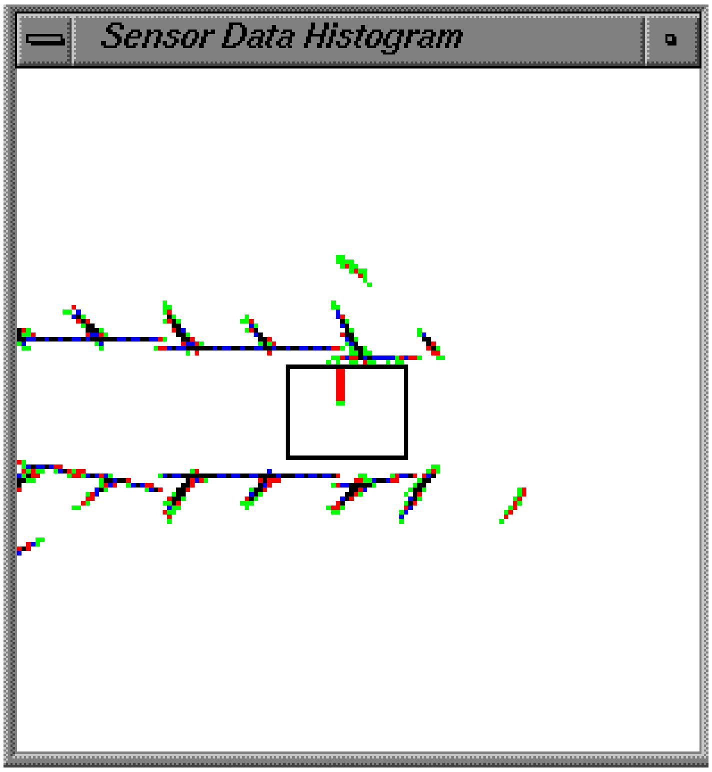


Figure 1A – Sample SWAMI with senors

Figure 1B – Sample Sensor Data Histogram

Under development is a Path Planner that generates a feasible path avoiding obstacles based upon information located in Map.

DYNAMICS

We are currently controlling velocity driven mechanisms. The inverse dynamics of these mechanisms yield the forces and torques that each link's actuator would need to generate motion based on each link's positions, velocities, and accelerations. This is useful to determine whether the desired velocity commands are feasible for a particular actuator. The inverse dynamics of the system is calculated using an iterative Newton–Euler dynamic formulation.

For a mechanism driven by force/torque commands or a low level controller, the forward dynamics of the system is required. Although this is currently not implemented on MBSim, it is readily implemented via the iterative Newton–Euler dynamic formulation.

Each link has a consistent velocity and acceration interface which enables only one function at the mechanism level to calculate the dynamics for each link in any mechanism in any configuration. By embedding the specific characteristics of a link inside the link class and using consistent interfaces to each link, general functions can be used to yield desired quantities regardless of the mechanism being studied.