Gazebo MATLAB Bridge using Ros serialization
V1.0

Gowtham Garimella

November 15, 2015

1 Overview

This package provides a gazebo plugin and a mex interface for fast communication between MATLAB and Gazebo. The mex interface provides fast access to link and joint states; easy application of link and joint efforts; and setting of model and joint states.

1.1 Prerequisites:

- ros-hydro
  Install Ubuntu 12.04: sudo apt-get install ros-hydro-desktop
- Gazebo 1.9
  Install Ubuntu 12.04: sudo apt-get install ros-hydro-gazebo-ros
- ros-hydro-gazebo-packages
  Install Ubuntu 12.04: sudo apt-get install ros-hydro-gazebo-ros-pkgs

Although the code has not been tested on Ubuntu 14.04, it can be tested by installing ros-indigo-desktop, ros-indigo-gazebo-ros and ros-indigo-gazebo-ros-pkgs in a similar manner.

2 Package installation

Clone the package in catkin workspace and source the setup_script.bash with the arguments as MATLAB_ROOT and ROS_WORKSPACE.

```
Example Usage: source setup_script.bash /usr/local/MATLAB/R2014a ~/hydro_workspace
```

3 Usage

To use the bridge, first run the gazebo with one of the world files provided in "worlds" folder using these commands:

```
roslaunch gazebo_rosmatlab_bridge basic.launch scene:=scene1
///scene1 can be replaced by any of the scenes from world folder
                OR
///Goto the worlds folder and run:
gazebo scene1.world #Will load the world file directly
```

Then you can run one of the MATLAB scripts provided to control the models. For a simple example run **arm_closedloop.m** after loading **scene1.world** into Gazebo and see what happens. For other interesting examples look into **section 3.3**

NOTE: Run Gazebo before running MATLAB script always.

3.1 Gazebo Plugin Usage

The world plugin to connect MATLAB with Gazebo is specified as:

```
<plugin name="gazebo_rosmatlab_bridge" filename="libgazebo_rosmatlab_bridge.so">
   <joints>rear_left_wheel_joint;base_to_steeringblock1;rear_right_wheel_joint</joints>
   <links>carbody</links>
 </plugin>
```

The joints of interest for the control problem are specified in a semicolon separated string format. Similarly links are also provided in the same format. If there are multiple links/joints with same name, full scoped names such as (rccar::rear_left_wheel_joint) can be used to distinguish them. When MATLAB client is started, these links and joints show up as "Available_Names" in the GazeboMatlabSimulator class property.

A visual plugin is also provided which allows publishing of desired and current trajectories of different links from MATLAB to Gazebo. This plugin should be loaded in the visual region of a link description in a static model such as ground. See "models/ground_plane_with_visual_plugin/model.sdf" for an example on how to use the visual plugin.

3.2 MATLAB Documentation

MATLAB scripts to control gazebo models are provided in matlab_scripts folder. The documentation for the folder and helper classes can be found in "docs/matlab_documentation" folder. The description of the documentation files is as follows:

Contents_report.html	Description of the files in the matlab_scripts folder and what they do
GazebomatlabSimulator_doc.html	Documentation of the various functions and properties for the bridge between MATLAB and Gazebo
MatlabLinkInput_doc.html	Documentation of Link input class
MatlabRigidBodyState_doc.html	Documentation of the Rigid body state class

These files should be opened from inside MATLAB for the hyperlinks to work properly. For control examples look at the section of examples under "Closed Loop Controllers" section.

3.3 World Files description

3.3.1 Point stabilization or trajectory tracking

World File	Description	Matlab Script
scene1.world	Double Pendulum	arm_closedloop.m/arm_computedtorquelaw.m
scene2.world	Rcccar model	car_closedloop.m
scene3.world	Wam arm	wam_servo_control.m
scene4.world	Quadcopter	quad_bs.m
scene5.world	Youbot	
scene6.world	Quadcopter with 2DOF arm	
scene7.world	AUV vehicle	
scene8.world	Satellite	satellite_test.m
scene9.world	Cyton Arm	cyton_servo_control.m

3.3.2 Optimal control

World File	Description	Matlab Script
scene1.world	Double Pendulum	arm_shooting.m arm_sddp_analytic.m arm_sddp_gazeobo.m
scene2.world	RcCar Model	car_shooting.m
scene3.world	Wam arm	wam_shooting.m

3.4 Mex Documentaion (Created from the source file):

This section is only for advanced users and is not necessary for students interested in just using the interface.

This file provides a mex interface for communicating with Gazebo. This file should be used along with the world plugin gazebo_rosmatlab_bridge to complete the connection with gazebo. This interface works by calling string arguments for completing specific actions. The supported string arguments and example usages are provided below:

- NEW: This creates the bridge. You should always cleanup the class by calling DELETE after you are done with your simulation. It provides a pointer to be stored and passed on later.

```
MATLAB USAGE: A = mex_mmap('new');
```

- DELETE: This closes the bridge properly. It should be provided with the pointer created using NEW

```
MATLAB USAGE: mex_mmap('delete',A); %A is the pointer created in above step
```

- RESET: This resets the world to its initial state. Useful for resetting the simulation after every sample.

```
MATLAB USAGE: mex_mmap('reset',A); %A is the stored pointer
```

- AVAILABLENAMES: This provides the available link and joint names from the link. The indices of the links and joints are used below to configure the links and joints

```
MATLAB USAGE: [Link_Names, Joint_Names] = mex_mmap('availablenames',A);
```

- CONFIGUREPHYSICS: Configure the physics engine in Gazebo to run faster/slower according to your simulation requirements;

```
MATLAB USAGE: mex_mmap('configurephysics', A, 0.001,2000);
```

Explanation: This runs the physics engine twice the real time speed with a physics timestep of 1 millisecond. The arguments taken are the physics engine timestep Physics engine update frequency: The frequency at which the internal physics engine step is called

- SETMODELSTATE: This sets the complete state of model using Model Pose (x,y,z, qw,qx,qy,qz, body_vx,body_vy,body_vz,body_wx,body_wy,body_wz)[13x1] in world frame and using Joint states(Joint angles, Joint Velocities)[2xn matrix]. The Joint Velocities are not supported yet. Gazebo somehow does not set the joint velocities and should be replaced with link twists somehow

```
MATLAB USAGE I: mex_mmap('setmodelstate',h.Mex_data,'Airbotwith2dofarm'...
                ,[0 0 0 1 0 0 0 0 0 0 0 0 0],uint32(0:1),[0 pi;0 0]);
MATLAB USAGE II: mex_mmap('setmodelstate',h.Mex_data,'Airbotwith2dofarm'...
                ,[0 0 0 1 0 0 0 0 0 0 0 0 0]);
MATLAB USAGE III: mex_mmap('setmodelstate',h.Mex_data,'Airbotwith2dofarm'...
                ,[],uint32(0:1),[0 pi;0 0]);
```

Explanation: The 4th argument is the model pose in world frame. The fifth argument is joint indices (should be uint32) and starting index with 0. The final argument is the actual joint angles and joint velocities.

- SETJOINTSTATE: This sets a single joint angle and velocity(Velocities not working properly). NOTE: This should not be used for moving multiple joints and setmodelstate should be used instead.

```
MATLAB USAGE: mex_mmap('setjointstate', A, uint32(0) ,[1,0]);
```

Explanation: This takes in the zero based joint index as a single scalar which is uint32. The Joint State is just the angle and velocity.

- RUNSIMULATION: This function runs the gazebo physics engine for specified number of steps and provides back the Link and Joint States at the steps specified

```
MATLAB USAGE: [LinkData, JointData] =  mex_mmap('runsimulation', A,  ...
                JOINTIDS, JOINT_CONTROLS, LINKIDS, LINK_WRENCHES, STEPS);
```

A	Stored Pointer
STEPS	Vector[$N\times1$ or $1\times N$] providing number of steps for physics engine. It should start with zero and should be increasing and the last element provides the number of steps the physics engine should take
JOINTIDS	zero based joint index vector for which controls are provided [$n\times1$ or $1\times n$]. The indices are obtained from available names described above.
JOINT_CONTROLS	Matrix of $[2\times(n\times N)]$ with each $[2xn]$ matrix showing the Joint efforts at that step. The controls are constant in between two steps.
LINKIDS	zero based link index vector for which link wrenches are provided [nx1 or 1xn]. The indices are obtained from available names described above.
LINK_WRENCHES	The Link wrenches are provided as $[6\times(n\times N)]$. Each $[6\times n]$ matrix provides wrenches for the set of links at that step.

Table 1: Parameter explanation for runsimulation

If you do not want to apply jointids or joint controls you can replace them by empty matrix. This applies to link ids and wrenches too

Example System: Double Pendulum where you want to control the two joints indexed 0, 1. We want to run the simulation for 1 second (Assuming physics timestep is 0.001 this is 1000 steps) with state data every 0.5 seconds. Also we assume that the joint effort is 0.1 for both joints in first half of the trajectory and -0.1 in second half of the trajectory. This is called from MATLAB as follows.

```
[LinkData,JointData] = mex_mmap('runsimulation',A, uint32([0 1]), ...
                              [0.1 -0.1; 0.1 -0.1], [], [], [0 500 1000]);
```

NOTE: For more examples look at the examples from matlab_scripts folder of the package

3.5 KNOWN BUGS:

There are some known bugs with this interface that need to be fixed.

- You can run the MATLAB scripts only when Gazebo is running. If not it will get stuck in some infinite loops and MATLAB should be forced to closed down
- The plugin works for Gazebo 1.9.x. It is known to fail in Gazebo 4.x and has not been tested on Gazebo2.x etc