Setting Up the ROS and V-rep Environments

Last edited on 4-10-15

The following documentation will highlight setup and focus on some of the solutions to problems encountered, many of which were simple but ended up taking many hours to solve. This file was created from a series of log files written before code was generated for the robot or simulation. It is highly recommended to closely follow ROS and V-rep documentation and to perform tutorials for both platforms to become adequately accustomed to their environments.

General Setup

Setup ROS on the primary computer: http://wiki.ros.org/indigo/Installation/Ubuntu
Setup ROS on the Raspberry Pi:
http://wiki.ros.org/ROSberryPi/Installing%20ROS%20Indigo%20on%20Raspberry%20Pi
When done, complete the tutorials focusing on creating the workspace, ROS command tools, writing a publisher/subscriber and writing launch files. Next download V-rep. To execute the program, go into the folder and enter *./vrep.sh*.

XBOX ROS Communication

The following section deals with the procedure used to have the XBOX controller working with turtlesim (as provided by ROS):

- Generally followed the tutorial: http://andrewdai.co/xbox-controller-ros.html
- Received explicit details for working within Python from: http://wiki.ros.org/joy/Tutorials/WritingTeleopNode
- Installed xboxdrv: driver for Xbox controller
- Installed joystick tools in Ubuntu using command *sudo apt-get install joystick*
    - enter *cat /proc/bus/input/devices* to see jsX ID where X will be specified next to "Handlers=".
- Installed and used ROS joy for turtlesim and entered the following commands:
    - *sudo apt-get install ros-indigo-joystick-drivers*
    - *rosdep install joy*
    - *rosmake joy*
- A scratch package called "learning_joy" was created using catkin tools, then the script was copy-pasted, then the cmakelists file was edited and a launch file created. Also used chmod to make the python script executable (see link 2).
    - launch file problem: in order to run the launch file, catkin_make must be called at the root of the workspace. If this produces an error saying that the Python script cannot be found, redo the type= area to: type="SCRIPT_NAME.py". The tutorial says not to do this, but it works.

- The simulation with controller can be run using "roslaunch PACKAGE_NAME.launch" after running roscore. Joy, the Python script, and turtlesim are conveniently called via launcher.

Integrating V-rep With ROS

- Downloaded version 3.2.0 into home/USERNAME directory
- Followed the tutorial: https://github.com/lagadic/vrep_ros_bridge
    - Used their "optional" option for Indigo. There was a problem initially running ROS services (an error regarding not locating library reference or somesuch), but this fixed by sourcing the environment. The following should be called in each new terminal window or the command can simply be added with an *--extend* appended to the end of the string when copying over to the .bashrc file (which can be accessed via home directory). The extend command allows for chain sourcing and should be used after a previous source. Do not use for separate workspaces which interact.
    - Entered: *source ~/catkin_ws/devel/setup.bash*
    - Came up against a cmake error when trying to build the plugin.
        - Despite creating environment variable VREP_ROOT_DIR set to new location, the CMakeLists.txt file in the plugin folder would repeatedly attempt to access old directory.
        - Solution: the problem had to do with the catkin workspace's build file CMakeCache.txt. In it, VREP_ROOT_DIR was set to the old directory. After writing in the new directory, catkin_make was able to successfully build the packages. Alternatively, deleting the build file and running catkin_make should resolve the issue.

Exporting URDF from Solidworks to V-rep

The following tutorial was used to export a .urdf file from Solidworks. The joint connections and textures were not maintained, but the mesh structures were. Rotary joints were then added to provide movement for the model within V-rep: http://blogs.solidworks.com/teacher/wp-content/uploads/sites/3/WPI-Robotics-SolidWorks-to-Gazebo.pdf.

An alternative option also attempted involved using was a .obj exporter module for Solidworks. The files were created successfully, but there was a problem carrying over joint information. Right now, it looks like .urdf is the more promising format.

Analyzing Solidworks to Migrate Relevant Data to V-rep

In order to actuate joints within the main assembly file, select the relevant part, open it then articulate the joint (which worked with the mouse). After closing the window, the main assembly should apply the changes. For measurement information, select the part and see the bottom of the tools menu for the desired option.

Running ROS Topics and Subscribers From a Raspberry Pi

- Refer to ROS documentation for turtlesim tutorial.
- If using Linux, it is possible to ssh into the Pi and then return any graphical interface back to your laptop using the -X command; ie: ssh -X pi@<IPaddressOfPi>. No extra software is required.
- Turtlesim runs VERY slowly when sshing into the pi and running it with the graphical interface being sent back to a laptop(it is equally slow with a wireless or wired connection, which is interesting). It runs slowly enough that it is impractical to do so.
- When running turtlesim on the Pi by hooking up the hdmi to a monitor, and the usbs to a mouse and keyboard, turtlesim runs fairly quickly (not as fast as my computer, but very reasonable speed)

Becoming Accustomed With V-rep

- Refer to video tutorials at: https://www.youtube.com/watch?v=w68jmN1IBpo&list=PL38P7Q24q4XA7c0uNj0kO4or-bKhFYdIg
    - refer to documentation at: http://www.v-rep.eu/helpFiles/index.html
- To get a handle on basic child script control of a motor open a new scene and from Models -> examples add a "simple Ackermann steering" model into the scene. When run, this should allow for basic motion control using arrow keys. Refer to non-threaded scripts in documentation to understand RoNot (or robot model) code.
- For our initial implementation, three entries were removed to get rid of the red lines used to visualize wheel orientation. The child script was then converted into a non-threaded script by replacing the function thread signifier with the appropriate if condition and the last command was removed during the "actuator" phase. The scene was then switched to a V-rep scene with imported .urdf model of the robot. Motor and steering joints were then created for the model and the Ackermann script was added to allow for control of four motors.
- There was an issue with erratic body movement, but this was fixed by creating a new invisible chassis using cuboid shapes with dynamic properties which covered the robot floor and bin sides.
- Covering wheels or contacting them was avoided because doing so produces erratic joint motor behavior due to awkward collision of collidable materials.
- Reversed right wheel orientations were fixed by flipping the joints 180 degrees around the relevant coordinate relative to its own frame. The imported mesh files did not produce

a balanced object, so a few values down the line were off and had to be manually adjusted.

A Few V-rep and ROS Tips

- See V-rep's tutorial > Writing Code in and Around V-rep > V-rep API Framework > ROS Interface, for information about publisher and subscriber functions. The former sends packets which can be identified by entering *rostopic list* then *rostopic echo TOPIC* to see message information. In Python, attributes will parallel headers echoed from the aforementioned command. There is usually some attribute required to be accessed before accessing primitive data. For example, when subscribing to a proximity sensor publisher from V-rep, a Point32 object was passed. This object itself contained a Float64 object which could be accessed by calling the data header as an attribute.
- The rospy module has its own time package to handle, for example, sleep rates.
- It is good practice to use C++ for lengthy, resource intensive code and Python for short scripts (unless, perhaps, during testing).
- When moving your catkin workspace, make sure to dereference all relevant source information before migrating and rebuilding.
- If joints in V-rep buckle during testing, this is likely due to either an incorrect hierarchy or joint property option, in which case refer to V-rep's designing dynamic simulations tutorial, or the movement of the joint is in some way constrained; materials aren't bendable, so parts will usually reorient themselves awkwardly in an attempt to get around this problem.
- Be sure to make use of launch files to execute nodes in order to decrease time to execution.
- The following links to the list of API functions useful when constructing child scripts in V-rep: http://www.coppeliarobotics.com/helpFiles/en/apiFunctions.htm#simGetSystemTime
- ROS features tab completion! Packages within the catkin workspace should autocomplete if built correctly.
- The Github account being used currently eats .xml files which are needed to properly build the packages in a catkin workspace. Refer to ROS documentation and inclusion headers in code to recreate these if necessary.

Current Node Structure

- The joy package's joy_node publishes to lunabot's control.py, as does V-rep's sensor publishers (refer to scene's child script).
- The control node then publishes four wheel pwm values (from -255 to 255) to converter nodes to V-rep and the Pi, which themselves respectfully publish to V-rep and send the values via sbus to their corresponding Arduino unit.

- The following commands will then be necessary to control the simulator (if not using a launch file):
    - *roscore*
    - *./vrep.sh* (when in installation folder)
    - *rosrun joy joy_node*
    - *rosrun lunabot control.py*
    - *rosrun control_to_vrep.py*
    - Additionally, the IR handler can be used to get proximity data by running IR_handler.py from the lunabot package.
    - To move the robot, simply replace the control_to_vrep.py script call with control_to_robot.py.