Analyzing Bagged Data in Octave/Matlab

Wyatt Newman
October, 2014

This document describes how to log data in ROS and interpret the data in Matlab or Octave.

Bagging data:
We have seen that ROS provides a convenient logging tool, "rosbag". For Atlas (physical or simulated), it is often convenient to log sensor data for post processing. These files can quickly get very large for the sensor head, so it is prudent to specify which topics are to get logged.

Start with a robot running—either physical Atlas or drcsim. Start up whatever nodes are desired and begin logging. You can start/stop the logger at any time without disrupting the robot control.

When performing logging, first navigate to a directory where you desire the logs to reside. From a terminal at this directory, enter:

```
rosbag record atlas/atlas_state
```

This will start recording messages transmitted to the topic atlas/atlas_state. The output file will, by default, be named with the current date and time. Optionally, you can specify an output file name, e.g.

```
rosbag record -O my_log  atlas/atlas_state
```

in which case the bag file will be contained in a new file called "my_log.bag".

You can also list more topics to be bagged, if desired.

When you have enough data of interest, stop the logging with a control-C in the terminal from which you started rosbag.

The bag file may be examined with:

```
rosbag info my_log.bag
```

Converting to Matab format:
For post-processing logged data, it is convenient to convert the bag file into a format compatible with Matlab/Octave. There is an HKU tool for this (written by Devin Schwab and Neal Aungst of CWRU). It is located in …/catkin/src/hku_tools/rosbag_to_matlab. The README file contained in this package includes details of operation.

To use this converter, enter:

```
rosrun rosbag_to_matlab bag_to_mat.py <path to bag file> <output directory>
```

`<path to bag file>` is the path to the bag file you wish to convert

`<output directory>` is the path to the directory where the .mat files will be written.

For example, from the same terminal used for bagging (and still in the same directory), enter:

```
rosrun rosbag_to_matlab bag_to_mat.py my_log.bag .
```

This will convert the file "my_log.bag", located in the current directory, and will create an output

file called “my_log_atlas_atlas_state.mat” in the same directory (since “.” is Unix shorthand for “current directory”). The file name prepends the original file name and post-pends the name of the topic logged (atlas_atlas_state). The output file will be in a format compatible with Matlab or Octave.

Interpreting Data in Matlab/Octave:
From a Matlab or Octave command window, navigate to the directory where your data resides. For the current example, the converted file is called “my_log_atlas_atlas_state.mat”. In the command line, enter:

load my_log_atlas_atlas_state.mat

Then type “who”, and you will see that two new variables exist: “data” and “header”
Enter: header
and you will see a long list of elements stored from atlas/atlas_state. A snippet of this follows:

```
header =


header.seq
header.stamp.secs
header.stamp.nsecs
header.frame_id
position.0
position.1
position.2
position.3
position.4
position.5
```

Entering: size(header) reveals that the header describes 291 variables stored in the bag file.

An inconvenience is that one must assign the correct index number to each of the data items logged, in order to plot or analyze specific sensor values.

An example Octave file is located in
…/catkin/src/examples/example_rosbag_analysis/example_octave.m

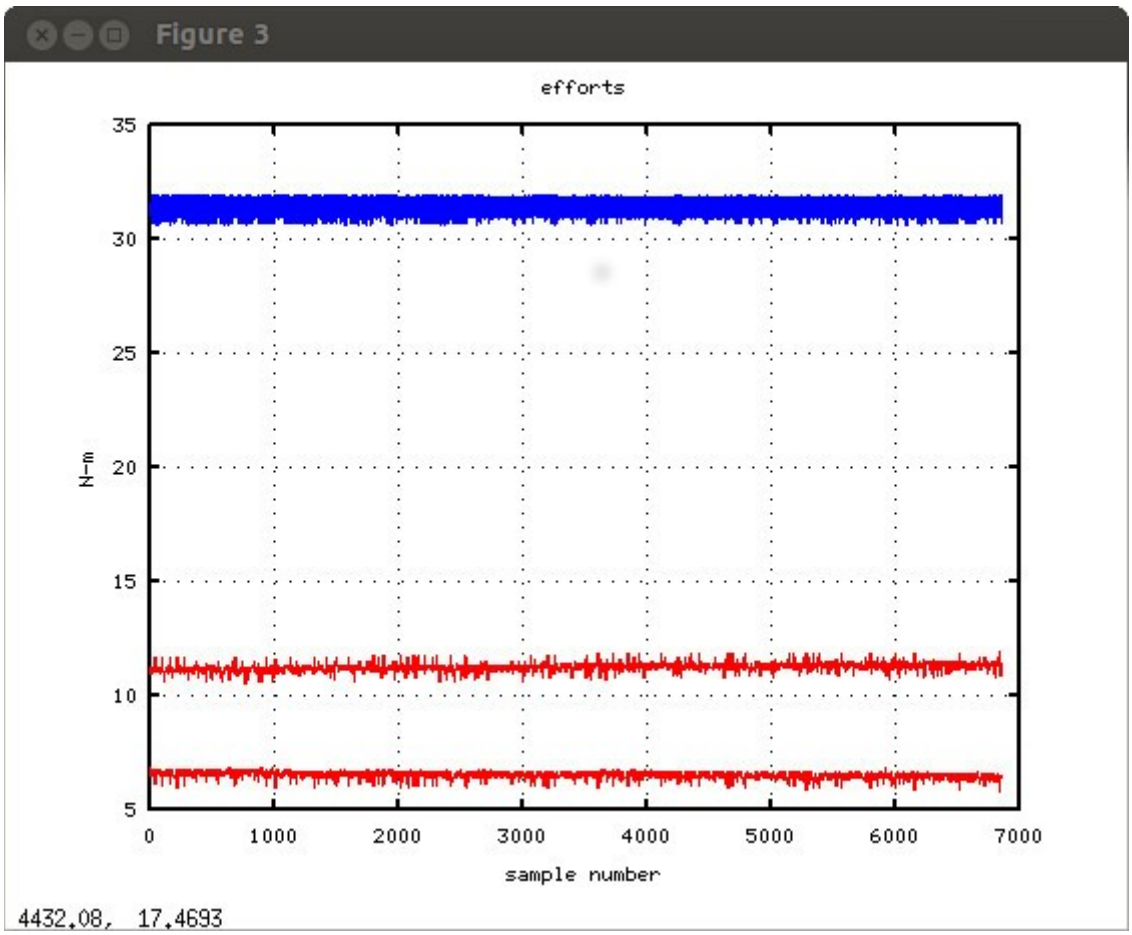
An excerpt from this program is:

```
for i=1:28 %extract just the joint angles from the logged data

positions(i,:)=state_data(:,4+i);
end
```

In the above, the offset of “4” is needed to point to the start of the joint-angle data in the log file. The result of this loop is that the “positions” matrix will have 28 rows of data, each corresponding to a joint number. NOTE, however, that Matlab starts counting arrays from index=1. Therefore, the angle of (ROS) joint “0” is at position(1,:). (And similarly for the velocities and efforts).

Running this Octave code on example drcsim data yields useful plots, an example of which is below:



This plot shows the two ankle efforts (per the joint-torque “effort” data), shown in red, and the back-bend (Atlas 2nd torso joint, i.e. joint-1), shown in blue. Positive torques on the ankles indicate that Atlas's weight is forward of his ankles (towards his toes). As computed by Octave, the mean of the sum of the two ankle torques is 17.7 N-m. Additionally, the mean of the z-forces from Atlas's foot force/torque sensors is 1475 N (which is consistent with Atlas' weight of 150 kg). Thus, Atlas's center of mass is 17.7/1475 = 0.012m = 1.2cm forward of his ankles (for the pose chosen, and for the drcsim version used).

Having converted bag data in to Matlab-compatible data, and unpacking the data into separate matrices, Octave/Matlab can be used to analyze many useful properties.