rqt_bag_diff: Tool for Visual Comparisons of Robot Sensor Data

Michael Jae-Yoon Chung

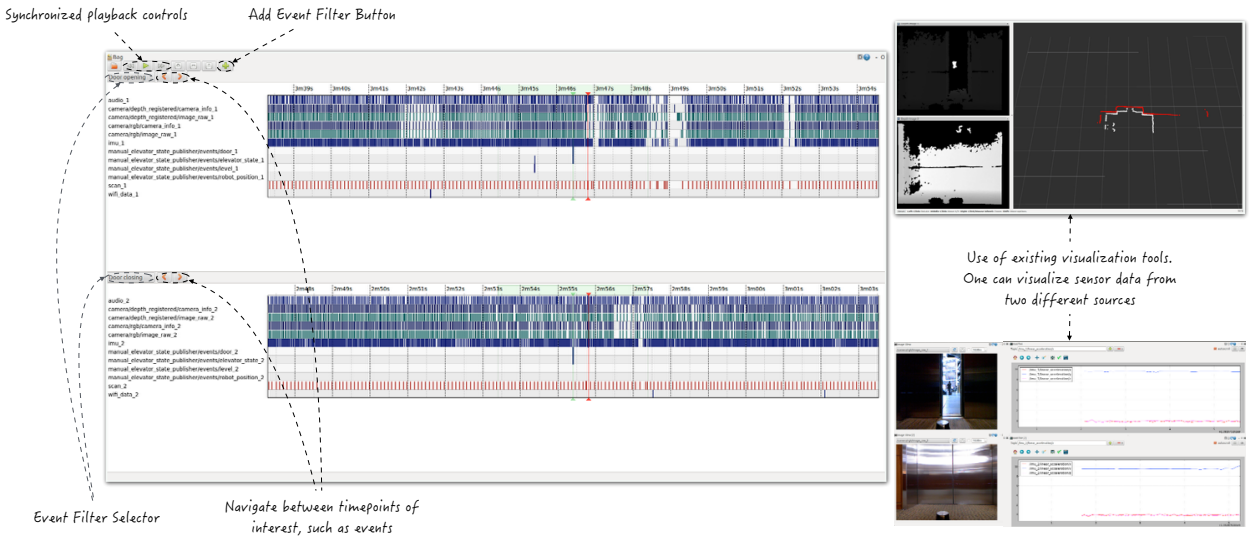


Fig. 1. Example use case of rqt_bag_diff. New control features that were not included in rqt_bag are annotated. Suggested configurations for rviz (upper right corner), rqt_plot and rqt_image (lower right corner) are used to visualize sensor data.

Abstract— rqt_bag_diff is a novel tool for visual comparisons of robotic sensor data. rqt_bag_diff provides effective data inspection and comparison functionalities by 1) building on top of an existing ROS-based visualization tool that is robust against size of the data, 2) providing new event-based navigation and filtering-based data management features and 3) providing templates for using existing ROS-based visualization tools leveraging small multiples and layering design principles.

1 Introduction

Building event detectors is a fundamental task for researchers working with robotic sensor data. By event, we mean any physical incident that we consider important to the robot, for example, door opening, robot bumping into a wall, etc. To analyze sensor data for building event detectors, researchers commonly 1) export sensor data into text files and read them from numeric computing and graphics software environments such as MATLAB [5], Python [3, 1] and R [6, 12], or 2) use ROS-based software, such as rviz [11], rqt_bag [9] and rqt_plot [10] to visualize and replay the data.

While both approaches have strengths, they also have significant problems. The first approach allows a user to create a custom visualization tool by providing them low-level graphics APIs. However, if sensor data is too large to fit in memory [1], then the user must write additional data management scripts. The user produced scripts are error prone and not generalizable to arbitrary sensor data.

• *Michael Jae-Yoon Chung is with University of Washington, Computer Science & Engineering. E-mail: mjyc@cs.washington.edu.*

For information on obtaining reprints of this article, please send e-mail to: tvcg@computer.org.

[1] a file containing 10 minutes of rgbd camera, imu, audio and laserscan sensor data collected with realistic data collection rate can have file size larger than 10gb.

The second approach provides visualization tools that can deal with large size sensor data. In addition, these tools are reusable since ROS has grown to become almost standard tools for dealing with almost any robotic sensor hardware. However, these tools are not designed for data comparison purposes–the user can visualize only one sensor source at a time, which makes it almost impossible to visually compare two sensor data from different sources. Therefore, performing data analysis tasks with the goal of building event detectors is extremely challenging.

Considering the pros and cons of previous approaches, we claim: robotic sensor data visualization tools for building event detectors must support effective *data inspection* and *comparison*. By effective data inspection, we mean the user must be able to inspect their sensor data regardless of its size, and be able to navigate to the timepoints of interest without complex scripting. By effective data comparison, we mean the user must be able to make visual comparisons of sensor data within existing visualization tools.

In this paper, we introduce rqt_bag_diff–a novel tool for visual comparisons of robotic sensor data. rqt_bag_diff provides effective data inspection and comparison functionalities by 1) building on top of an existing ROS-based visualization tool that is robust against size of the data, 2) providing new event-based navigation and filtering-based data management features and 3) providing templates for using existing ROS-based visualization tools leveraging small multiples and layering design principles. We then present a case study involving sensor data collected from a mobile robot.

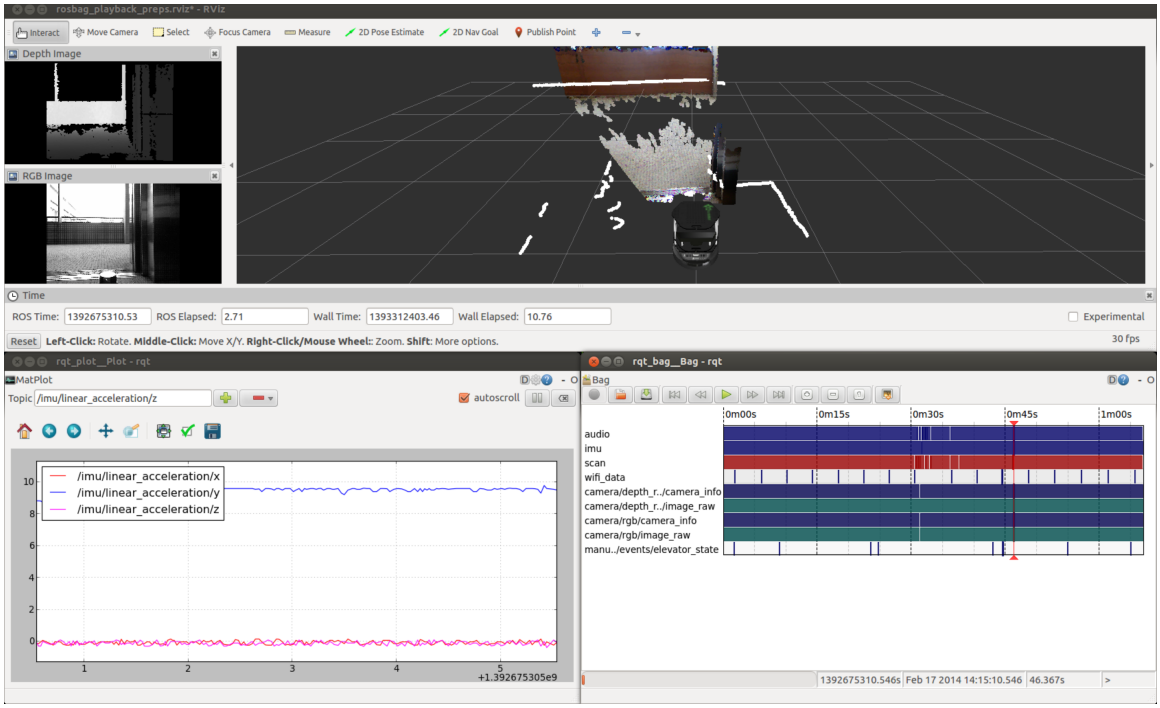


Fig. 2. Data analysis scene with rqt_bag. rviz (top) is displaying sensor data from rgbd camera and laserscan sensors. rqt_plot (lower left) is plotting three dimensional linear acceleration data (x,y,z) from imu sensor. Although not visible, recorded sound data is also being replayed.

2 Related Work

To the best of our knowledge, we have not yet seen an existing visualization tool that explicitly supports data analysis with the purpose of building event detectors. However, our work is closely related to visualization tools from multiple communities.

From the robotics community, rviz [11], rqt_bag [9] and rqt_plot [10] are the most related tools that can indirectly help perform data analysis with the intent of building event detectors. rqt_bag is a GUI for navigating and replaying .bag formatted sensor data (bag file[2])–a file format given to the data recorded using rosbag [8]. rviz allows a user to visualize sensor data in the simulated 3D environment. For example, a user can visualize a robot description model, laser scan data, and pointcloud data in spatially correct places. rqt_plot is a real-time line plotting tool for visualizing any numerical sensor data. For example, a user can display three dimensional accelerometer sensor data using rqt_plot with three overlaid lines with distinct colors.

Using these three tools together, a user can visualize and navigate sensor data with minimal effort. However, performing data analysis with the purpose of building event detectors with these tools is still a challenging task. First, the user must memorize summaries of previously visualized sensor data from the multiple time intervals that they is interested in. Then, they needs to compare those memorized summaries to come up with an idea for building event detectors. In practice, this is nearly impossible as the size of sensor data grows.

From the computer vision communities, we observed numerical computing and graphics software environments, such as MATLAB, R and Python are often used for data analysis. Their rich, low-level graphics APIs provides expressiveness to users. rqt_bag_diff attempts to adapt some of this expressiveness for data management purposes. However, rqt_bag_diff is fundamentally different since we do not require users to write any scripts for visualization purposes.

The Viz-A-Vis [7] project attempts to compactly visualize multiple computer vision features and their aggregations from large video data ($>$ 7500 hours) using 3D visualization techniques. In addition, it provides an interactive navigation functionality to help the user inspect the data effectively. While this tool provides a road map of computer vision features which can be useful for the understanding data patterns near the events, this tool does not support time interval comparison functionality which is critical for building event detectors.

Visualization tools from the video editing and annotation fields are also related. The ANVIL video annotation tool [4] allows the user to annotate video in a hierarchical manner. The user can annotate each time frame of the video data with user-defined "state", and those annotated states can later be organized into a hierarchical structure. The hierarchical annotation structure formed by the user gives a well-organized landscape of annotations, which in our case can be the landscape of the events.

Apples iMovie [2] also provides intuitions for organizing and navigating the time-series data. Their overall layout, video data drag and drop editing interface and instant playback while moving the mouse in the video clip area makes iMovie an intuitive inspecting and editing tool for multiple video data.

3 Methods

3.1 rqt_bag

Since we are using rqt_bag as a base framework for our work, we first describe rqt_bag. The lower right window in Fig. 2 shows the original rqt_bag interface. In the center, loaded bag files are displayed on a timeline. On the left side of the timeline, the names of recorded *topics*[3] are listed. On the timeline, the presence of sensor data from each topic is displayed as a vertical colored bar. For example, in Fig. 2, we can see that the data is sparse in the `wifi_data` and `manu../events/elevator_state` topics, whereas the other topics have denser data. The color of each bar represents the type of the sensor data; green represents camera-related data, red represents laserscan-related data and blue represents all other types of sensor data.

The control buttons are located on the top portion of the interface. The first three buttons on the left are for recording, loading and saving

[2]See `http://wiki.ros.org/Bags` for more details about ROS bags

[3]Topic is a term used in ROS to refer to a communication channel. For example, /audio refers to a topic where the audio data is transferred. Please see `http://wiki.ros.org/Topics` for details.

bag files. The following three buttons are for slowing down, starting/stopping and speeding up the playback speed. The next three buttons are for zooming in, zooming out and zooming to fit the timeline. The last button toggles datatype-specific extra visualizations. In addition to the buttons, the interface also provides basic keyboard inputs to move the timeline left or right and mouse inputs to seek to a desired position in the currently displayed timeline.

Fundamentally, rqt_bag provides a means to seek and navigate sensor data in real-time, which can be visualized by rviz and rqt_plot. When the data is being replayed, the location of data that is currently being replayed is indicated with the red playhead (in Fig. 2, it is located around 0m45s).

3.2 Event Filter

We introduce an *event filtering* method for on-the-fly data management and event-driven data inspection. Event filtering is a method for identifying timepoints that a user is interested in, e.g., all timepoints where door opening events occur. The extracted timepoints can later be used for timeline navigation. For example, a user might be interested in replaying data around the timepoints when the door-opening annotation data is set to true.

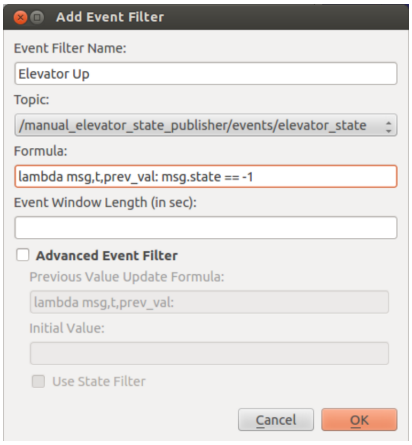


Fig. 3. Add Event Filter dialog box. Dialog box for creating an event filer.

We provide a programmatic approach to solve the event filtering problem. In particular, we provide an interface for a user to write filtering code and select an arbitrary topic as shown in Fig. 3. The user-provided line of code will be used to extract timepoints from the user selected topic. For example, if the user is interested in the timepoints where the door-opening annotation data is set to true, they can first select the topic which has door event annotation information. Then they can provide an anonymous python function that takes three variables `msg`, `t` and `prev_val` as inputs and returns a boolean value, which will be used for filtering the data. Here is an example.

```
lambda msg, t, prev_val: msg.state == DOOR_OPENING
```

`msg` is a data point in the selected topic, `t` is a corresponding timepoint and `prev_val` is an extra carried variable. `msg.state` is an integer field of a data point object and `DOOR_OPENING` is an integer value for representing door-opening event.

The extracted timepoint using an event filter is a single moment in time, not a time interval. Often, the user is interested in seeing the sensor data around the timepoint associating with an event. We provide an "Event Window Length (in sec)" section for the user to define a time interval around event timepoints that they like to replay using rqt_bag_diff.

In addition, the user can express more sophisticated filtering formula by using "Advanced Event Filter" option as shown in Fig. 3. Internally, our filtering implementation iterates the sensor data in increasing time order. "Previous Value Update" expects an anonymous python function that takes three variables `msg`, `t` and `prev_val` as inputs and returns any typed value. Once provided, "Previous Value Update Formula" sets `prev_val` variable independently from "Formula" in every iteration. Then the updated `prev_val` is available in both "Formula" and "Previous Value Update Formula". Optionally, the user can give initial value of `prev_val` using the "Initial Value" input line in Fig. 3.

Checking the "Use State Filter" makes the rqt_bag_diff focus on *states* rather than events. By states, we mean the time interval between two consecutive and distinctive events. Once the user checks "Use State Filter", the input for the "Event Window Length (in sec)" section will be ignored and the time interval between two consecutive and distinctive event will be used. For some analysis, replaying the data based on the states is preferred to the events.

3.3 rqt_bag_diff

rqt_bag_diff (shown in Fig. 1) offers a event filter based navigation. It allows a user to jump directly to one of the timepoints extracted by a selected event filter. On top of the two timelines, there are an event filter selector for selecting one of user created event filters, and two event filter based navigation buttons for jumping to a previous and next event timepoint.

Our tool also facilitates comparing two different timepoints. It is difficult to compare multiple events that occur in different timepoints using just a single timeline as in rqt_bag. Our tool provides stacked comparison of events as follows. First, it displays two copies of the timeline that are stacked together. On each timeline, the timepoint extracted by a selected event filter is marked with the green static playhead (WARNING: hard to see), and the user defined time interval is highlighted in green on top of the timeline. In fact, the green playhead is always located at the center of time interval highlighted with a green color. The green bars from the two timelines are vertically aligned whenever the user uses event-based navigation buttons. This behavior was implemented to encourage the user to focus on the event timepoints from two different timelines and compare them. However, two timelines can be moved and zoomed independently to provide more flexibility to the user.

When a user click the play button, the rqt_bag_diff replays sensor data from both timelines in a synchronized manner. This synchronized playback helps a user to visualize data from two timelines in rviz and rqt_plot simultaneously. A user can display these data in layered or side-by-side manner–by default, we provide layered configurations for sensor data displayed in the simulated 3D environment and side-by-side configurations for camera images and line-plots.

4 Case Study

We investigated the example usage of rqt_bag_diff in the context of building event detectors for multi-floor navigation. We were interested in building two event detectors; 1) door opening and door closing event detector and 2) elevator moving up and elevator moving down event detector. We used sensor data collected from on-board microphone, imu, laserscan, wifi and rgbd camera while the robot is traveling between multiple floors and riding elevators in the CSE building. The total length of the logged data was about 9 minutes[4].

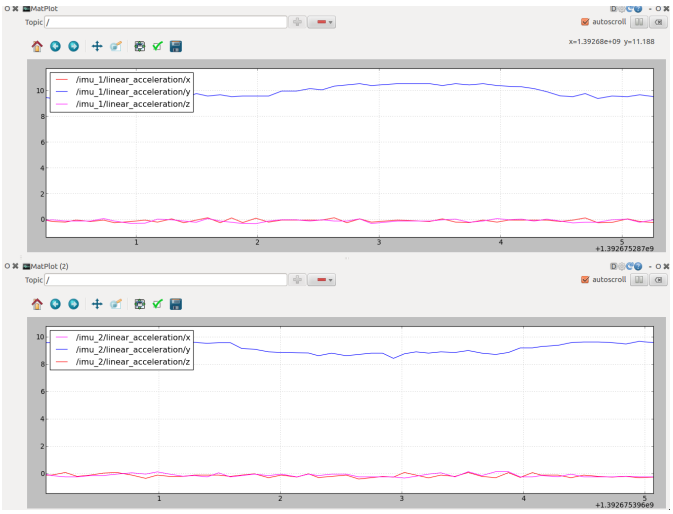
To investigate possibility of building event detectors, we needed to 1) estimate the sensor data distribution across the intra event class instances and estimate rough difference between inter event class instances and 2) identify which sensor is the most informative.

For building door opening and closing event detector, we first created two event filters using following formulas:

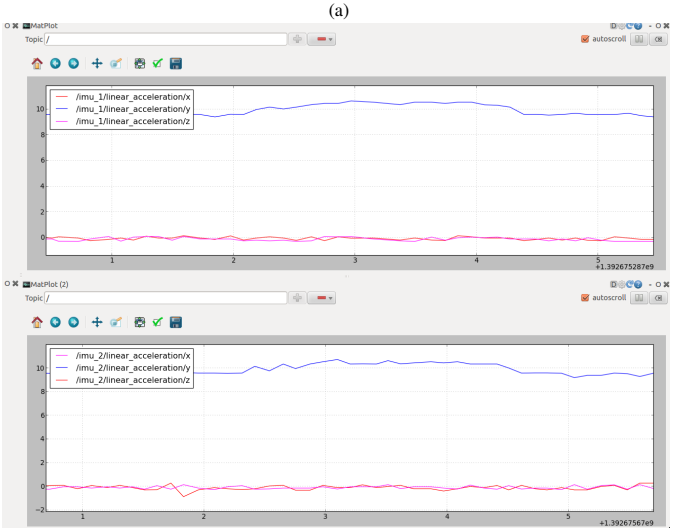
```
lambda msg, t, prev_val: msg.state == DOOR_OPENING
```

```
lambda msg, t, prev_val: msg.state == DOOR_CLOSING
```

[4]the data is available at `http://www.cs.washington.edu/homes/mjyc/shared/cse512-final/test_data2_rerecorded.bag`



(a)



(b)

Fig. 4. Intra and inter class comparisons for elevator moving up and down events. Three dimensional linear acceleration data (x,y,z) from imu sensor is plotted using rqt_plot. (a) Interclass comparisons: the top plots are showing data from one instance of elevator moving up events and the bottom plots are showing data from one instance of elevator moving down events. (b) The top plots are showing data from one instance of elevator moving up events and the bottom plots are showing data from another instance of elevator moving up events.

There were total 24 annotated door opening and door closing event (12 instances for each event class). To compare the inter class differences, we selected the door opening event filter from the top timeline and selected the door closing event filter from the bottom timeline. We then replayed the data and visually inspected rviz and rqt_plot (as shown in Fig. 1). We repeated same procedure for all door opening and closing event pairs.

From this procedure, we identified laserscan and rgbd camera sensor as the most informative sensors for detecting door opening and closing events. The average value of laserscan data was increasing near the door opening event and decreasing for the door closing event. The average of brightness values from rgb vamer or the average of depth values from depth camera had similar patterns for the door opening and closing events.

We repeated the same procedures for building the elevator moving up and down detector. We found y-axis direction (the axis perpendicular to the floor) linear acceleration from imu sensor was the most informative. Fig. 4(a) shows the comparison between one random instance from elevator moving up event and one random instance from elevator moving down. We can see that `/imu_1/linear_acceleration/y` and `/imu_2/linear_acceleration/y` topics have exactly opposite patterns. Fig. 4(b) shows the comparison between one random instance from elevator moving up event and another random instance from elevator moving down. We can see that `/imu_1/linear_acceleration/y` and `/imu_2/linear_acceleration/y` topics have very similar patterns. The results matched with our expectations.

5 Discussion and Future Work

We are aware of the fact that our results do not include any user study. However, since we built our tool, couple robotics students in our lab has been using the tool and we made some informal observations from them. All the students were be able to learn rqt_bag_diff quickly ($<$ 10min) and use it without difficulty–we suspect that this is because they are already familiar with ROS. However, some of them preferred using rqt_bag to using our tool. Those users mentioned that it was because 1) some of rqt_bag_diff features were not fully compatible with existing ROS tools and 2) they were already too familiar with rqt_bag. We also observed that nobody used the "Advanced Event Filter" feature. The "Advanced Event Filter" feature required the user to understand how the tool works internally, which prevented them to use it.

One important feature that was not fully supported by rqt_bag_diff was support for time information stored in bag files. The rqt_bag is heavily depends on saved time information in bag files. In rqt_bag_diff, time information from bag files is ignored and uses reasonable proxy time information. This heck was used to play sensor data from two different timepoints at the sametime in synchronized manner.

In future, we would like to run carefully designed user studies. We want to know which added feature is helpful and measure how much our tool is helping users to do sensor data comparison tasks. In terms of new feature, we want to implement time warped playback to synchronize playback of two unequal length event time intervals. In addition, we would like to one provide classifier analysis supports from our tool. We believe the user must be able to analyze the data and build classifier within ROS. We are considering how to visualize choice of classifiers and its parameters, and its performances.

References

[1] J. D. Hunter. Matplotlib: A 2D graphics environment. *Computing In Science & Engineering*, 9(3):90–95, 2007.

[2] Apple iMoive. `http://www.apple.com/mac/imovie/`, 2014. [Online; accessed 2-24-2014].

[3] E. Jones, T. Oliphant, P. Peterson, et al. SciPy: Open source scientific tools for Python, 2001–.

[4] M. Kipp. Multimedia annotation, querying and analysis in anvil. *Multimedia information extraction*, 19, 2010.

[5] MATLAB. *version 8.02.0 (R2013b)*. The MathWorks Inc., Natick, Massachusetts, 2013.

[6] R Core Team. *R: A Language and Environment for Statistical Computing*. R Foundation for Statistical Computing, Vienna, Austria, 2013. ISBN 3-900051-07-0.

[7] M. Romero, J. Summet, J. Stasko, and G. Abowd. Viz-a-vis: Toward visualizing video through computer vision. *Visualization and Computer Graphics, IEEE Transactions on*, 14(6):1261–1268, 2008.

[8] ROS rosbag package. `http://wiki.ros.org/rosbag`, 2014. [Online; accessed 2-24-2014].

[9] ROS rqtbag package. `http://wiki.ros.org/rqt_bag`, 2014. [Online; accessed 2-24-2014].

[10] ROS rqtplot package. `http://wiki.ros.org/rqt_plot`, 2014. [Online; accessed 2-24-2014].

[11] ROS rviz package. `http://wiki.ros.org/rviz`, 2014. [Online; accessed 2-24-2014].

[12] H. Wickham. *ggplot2: elegant graphics for data analysis*. Springer New York, 2009.