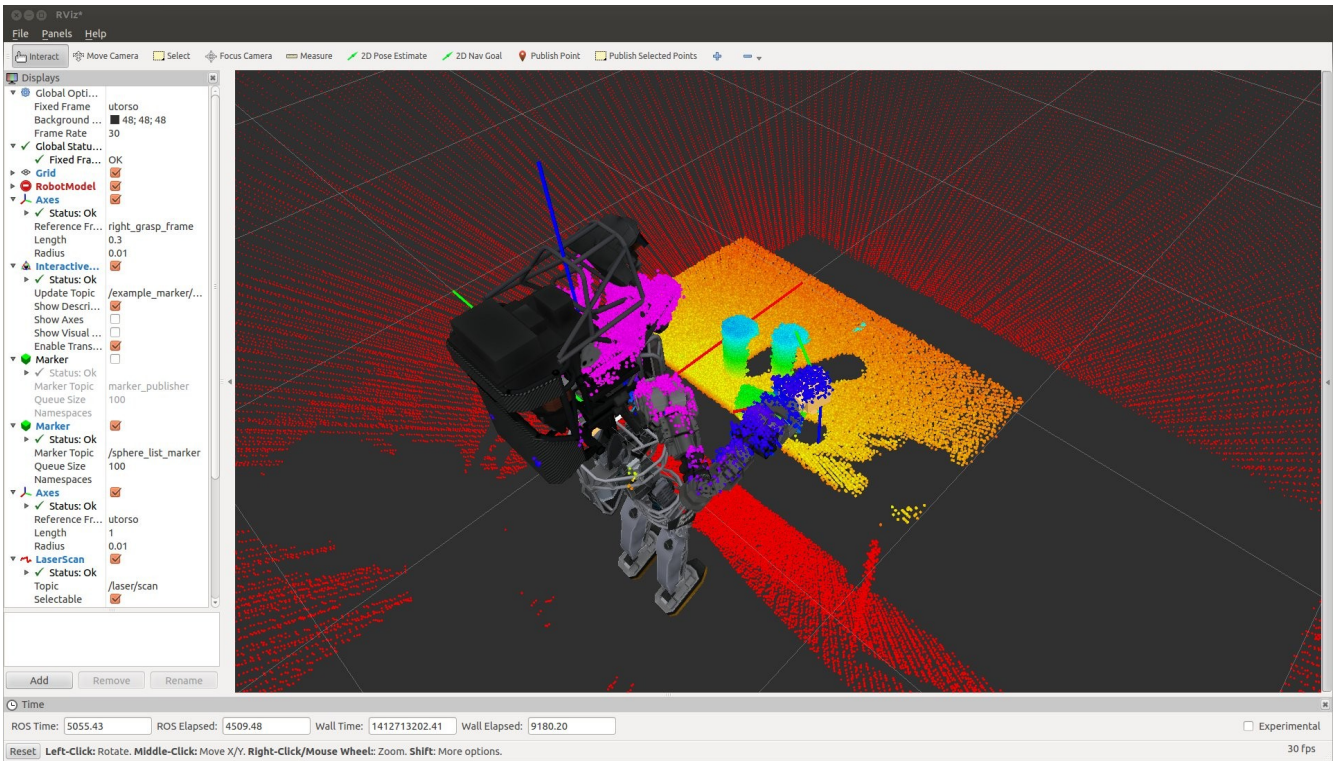
Interacting with Point Cloud Data in Rviz and Invoking Task Commands

Wyatt Newman
October, 2014

Use of the (spinning) LIDAR allows visualization of 3-D sensory data in Rviz. By rotation and translating the Rviz scene, a human operator can interpret Atlas's environment. The (typically, remote) human operator at an Operator Control Station (OCS) only has access to Atlas's sensor data—not a direct view of Atlas.

In the rviz scene below, we can make out a table and three items (cans) on the table.



The scene has many artifacts, including shadowing of points, noise, and coarse sampling. The cans only have LIDAR samples from line of sight, and thus this data is hard to align with a complete model of a can. Humans, though, are quite adept at understanding/interpreting a scene, in spite of considerable defects. We can easily visualize the tops of the cans in the above scene, and we can recognize the centers. Ideally, the remote operator would exploit human scene interpretation capability to give Atlas higher-level instructions. From experience, teleoperation of Atlas is ineffective—exacerbated by his severe kinematic constraints. Thus, the remote operator should issue commands based on task objectives (e.g., pick up the can from above) and let software compute what is achievable for Atlas.

Communicating goals to Atlas easily and abstractly is aided by having a narrowed context within a specific skill. An example useful skill is to approach an object with a hand from above, with palm facing down, and descend until touch (or nearly touch), setting up a pose suitable for grasp. In this context, the task command can be as simple as specifying desired coordinates of the top of the object of interest. With this information, inverse kinematics can be used to find sequences of joint angles that lead the hand to the desired goal pose. By implication, the desired hand orientation has its palm normal

pointing down. However (in this context), rotation of the hand about its palm normal is unconstrained, offering a greater range of inverse-kinematics solution opportunities.

Selecting Points in Rviz: The "Publish Selected Points" tool in Rviz offers the opportunity to select a patch of points via a mouse, and the points in the selected region are published to the topic /selected_points. The example program in the class package: example_pointcloud_selector (with source file of the same name) illustrates how one can operate on this data.

In this example program, a callback function responds to new publications on the topic /selected_points. The message type is sensor_messages::PointCloud2. A "point cloud" is merely an array of points. The message header contains both a time stamp and the reference frame for the coordinate data.

An open-source library, the "Point Cloud Library" (PCL, see http://pointclouds.org/) offers a variety of useful software for operating on point clouds. This library grew up independent of ROS, but it may be used with ROS by converting ROS messages into PCL-compatible objects.

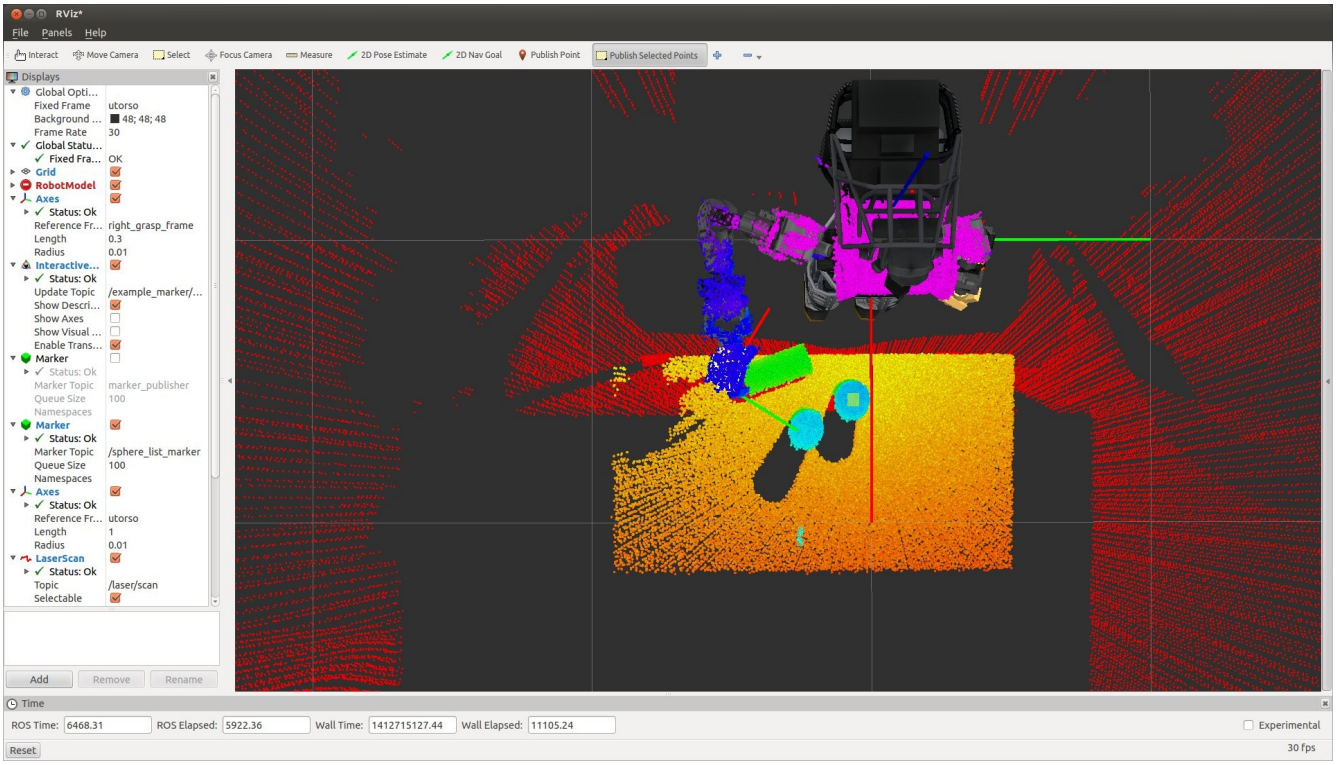
The example "example_pointcloud_selector.cpp" has its callback function respond to the ROS message for selected points, then it converts this message into a PCL type with the function:
pcl::fromROSMsg(*cloud, *pclSelect);

The pclSelect object is of type: pcl::PointCloud<pcl::PointXYZ>. This object is presented as an argument to the function computeCentroid(), which operates on the point-cloud data to compute a centroid of the set of points. It can access the points, e.g., in the style:

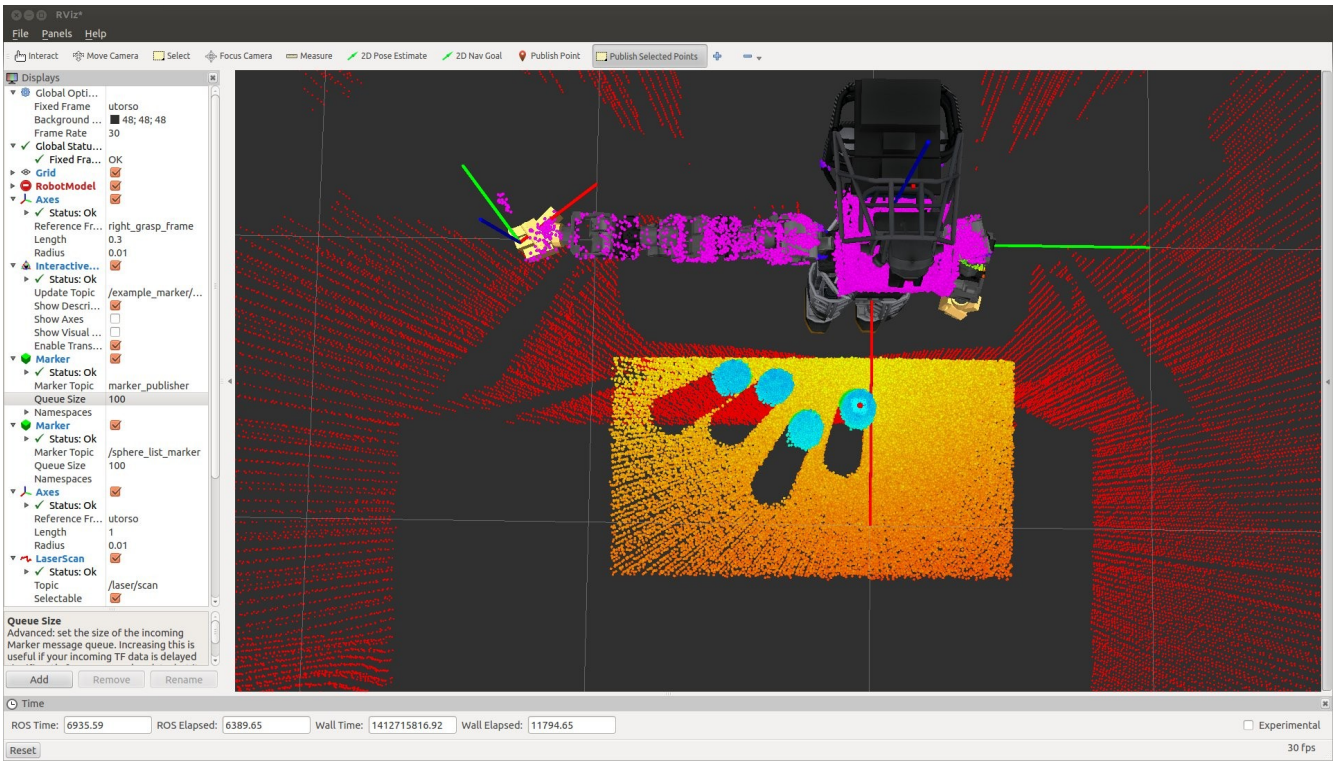
```
centroid.x += pcl_cloud->points[i].x;
```

The example computeCentroid() function is a simple illustration; it should be improved to reject outliers. In selecting points, it is easy to unintentionally include points very far from the patch of interest. The centroid function would be more effective, e.g., if it performed clustering, rejected outliers, and returned the centroid of the largest cluster.

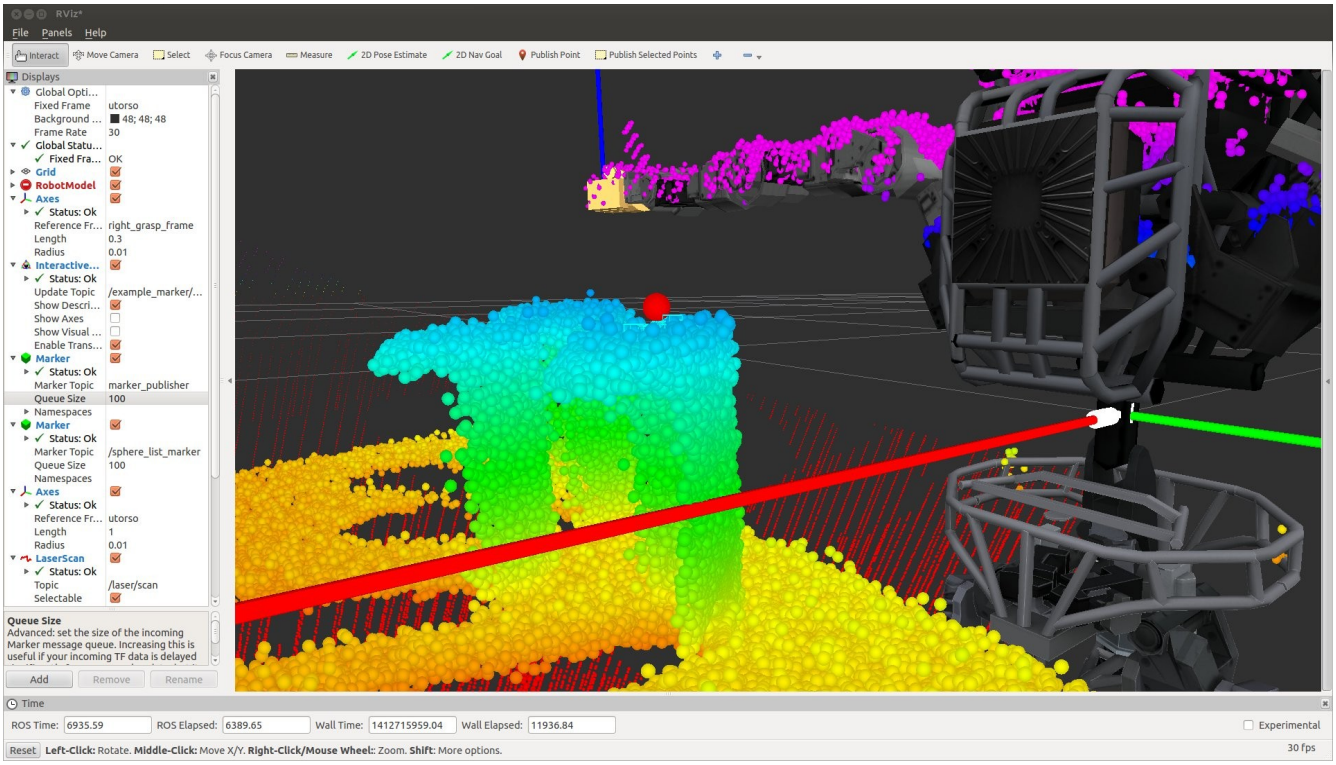
The screen-capture below shows a top-view scene. A small, yellow rectangle is visible on the top of the can closest to the robot. This region was selected with a mouse click/drag operation. Upon releasing the mouse button, the selection tool finds and publishes all of the 3-D points bounded by this rectangle (projected normal to the view).



The scene below shows an example of the effect of selecting a region, together with a marker publication of the centroid. (In this instance, the centroid was deliberately placed 2cm above the computed points, to allow a hand-approach clearance). The red dot illustrates the location of the centroid computed from the point-cloud selection.



A side view is shown below.

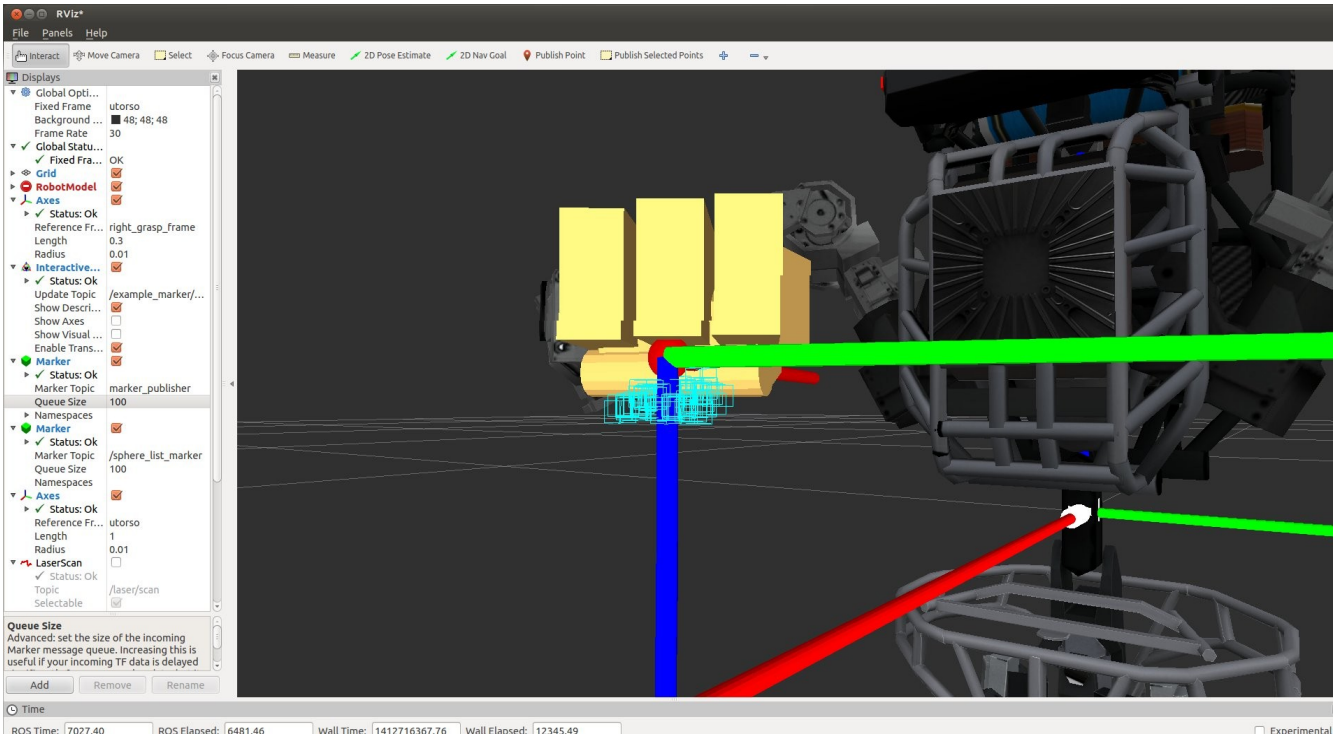


The red marker shows the computed (offset) centroid 2cm above the top surface of the can, as desired.

This is one, useful way that one can interact directly with sensory data to imply task specifications.

The computed centroid is republished by the node example_pointcloud_selector on the topic "/centroidStamped", and thus made available to additional nodes. A specific example is in the package: example_interactive_ik, with source file example_grab_from_above_v3.cpp. This node subscribes to the topic "/centroidStamped", and it uses this point plus a presumption of vertically-oriented palm normal to seek inverse-kinematic solutions (with an option for arbitrary rotation about the palm normal). This example node will send the arm to a valid solution, if one exists, by interacting as a client with the joint-trajectory action server.

The screenshot below shows convergence of the robot's right hand grasp frame to the selected point. (The red sphere marker, corresponding to the published centroid, is coincident with the grasp-frame origin). Also, the palm normal is pointing down, as desired.



This scenario can be recreated using the following set-up (which may be simplified with a launch file).

In one tab (terminal):

```
VRC_CHEATS_ENABLED=1 roslaunch hku_worlds beer_table.launch
```

This will bring up the scenario of the robot standing in front of a table with several cans present. Gazebo will be launched with an option to “pin” the pelvis of the robot, which is convenient for development (to keep Atlas from falling over).

in 2nd tab:

```
rostopic pub --once /atlas/mode std_msgs/String '{data: "pinned_with_gravity"}'
```

This command pins the pelvis.

Wait for this to complete, then re-use this window with:

```
roslaunch fc_bringup sim_task_4.launch
```

This will bring up the LLJC and the joint-trajectory action server (and multiple other nodes, not yet introduced).

in a 3rd tab:

```
rosrun sim_controller USER
```

This will inform the low-level joint controller that the user will take control of the joints (not a third-party behavior).

Wait for this to complete, then re-use the tab to:

rostopic pub --once /multisense_sl/set_spindle_speed std_msgs/Float64 '{data: 3.0}'
This command will start the spindle rotating, so the LIDAR data will be 3-D.
Wait for this to complete, then re-use the tab again to:

rosrun rviz rviz
which starts up Rviz.

Make sure the settings display the LIDAR to your liking, including LIDAR display, persistence, and colorization. Also, turn on the marker display on topic: marker_publisher.

in a 4th tab:

rosrun example_interactive_marker marker_listener_v2
This will start a node that listens for coordinates and causes a corresponding marker to display in rviz.

in a 5th tab:

rosrun example_pointcloud_selector example_pointcloud_selector
This node performs centroid computations on selected point-cloud points and publishes the result.

in the 6th tab:

rosrun example_interactive_ik example_grab_from_above_v3
This node interprets centroid publications as desired right-hand grasp-frame origins, with the assumption of down-ward facing palm normal. If solutions are found, the arm is commanded to go there.

Watch the output in tab 6; it will require user responses

In the rviz view, use the "PublishSelectedPoints" tool to select regions of interest;
Watch the output in tab 6. If the chosen point is reachable and approachable from above, the robot will try to move there.