ROS Support for Astra Series

Table of Contents

Overview

ROS (Robot Operating System) is a popular robotics middleware. As the world's leading RGBD manufacturer, Orbbec also has ROS environment SDK for developers based on OpenNI2. This document mainly explains how to compile, run and get the data of Astra series camera under ROS for further development. Please visiting http://wiki.ros.org/Sensors/OrbbecAstra for more information. This document assumes that readers understand and are familiar with the basic operations of Linux and ROS. All operations are based on ROS Indigo system. For readers using other version, please replacing the version code of indigo to the proper one. For example, in kinetic version, change “ros-indigo-astra-camera” to “ros-kinetic-astra-camera”.

ROS System Installation

Visiting http://wiki.ros.org/ROS/Installation for detail steps about installing and configurating ROS environment. Astra only support Indigo and newer ROS system version. Correct Ubuntu versions that introduced in wiki should be utilized when installing ROS system. For example: ROS Indigo -> Ubuntu 14.04, ROS Kenetic -> Ubuntu 16.04.

http://wiki.ros.org/indigo/Installation/Ubuntu

http://wiki.ros.org/ROS/Tutorials/InstallingandConfiguringROSEnvironment

ROS SDK

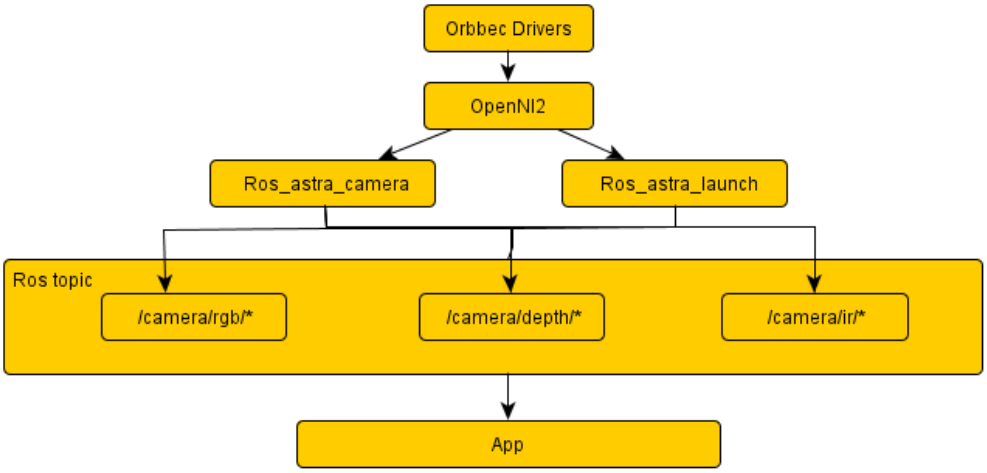
1. Using apt command to install for Ubuntu Indigo and newer version.
   $ sudo apt-get install ros-indigo-astra-camera
   $ sudo apt-get install ros-indigo-astra-launch

2. Another method is to compile source code.

1) GitHub: https://github.com/orbbec/ros_astra_camera and https://github.com/orbbec/ros_astra_launch
2) Installing ros package needed by SDK.
   $ sudo apt-get install ros-indigo-rgbd-launch
   $ sudo apt-get install ros-indigo-camera-info-manager
3) Compiling ros astra sdk and registration system environment.
   $catkin_make --pkg astra_camera
   $source devel/setup.bash
4) Installing UDEV file.
   $ roscd astra_camera && ./scripts/create_udev_rules

Please referring readme in GitHub for more compiling information.

ROS SDK Basic Structure



ROS SDK Execution and Data Collection

1. Astra Execution

1) Roslaunch to start Astra camera:

$ roslaunch astra_launch astra.launch

```
etting /run_id to c6b0043c-6eaa-11e7-9370-000c298ed66e
rocess[rosout-1]: started with pid [6538]
tarted core service [/rosout]
rocess[camera/camera_nodelet_manager-2]: started with pid [6549]
rocess[camera/driver-3]: started with pid [6556]
rocess[camera/rgb_rectify_color-4]: started with pid [6557]
rocess[camera/depth_rectify_depth-5]: started with pid [6558]
rocess[camera/depth_metric_rect-6]: started with pid [6577]
rocess[camera/depth_metric-7]: started with pid [6591]
rocess[camera/depth_points-8]: started with pid [6595]
rocess[camera/register_depth_rgb-9]: started with pid [6610]
rocess[camera/points_xyzrgb_sw_registered-10]: started with pid [6625]
rocess[camera/depth_registered_sw_metric_rect-11]: started with pid [6633]
rocess[camera_base_link-12]: started with pid [6646]
rocess[camera_base_link1-13]: started with pid [6654]
rocess[camera_base_link2-14]: started with pid [6670]
 INFO] [1500706542.181969785]: Initializing nodelet with 4 worker threads.
rocess[camera_base_link3-15]: started with pid [6688]
 INFO] [1500706542.342949269]: Device "2bc5/0401@3/7" found.
arning: USB events thread - failed to set priority. This might cause loss of da
a...
```

2) Running the node to show Astra information:

$ rosrun astra_camera astra_list_devices

```
orbbec@localhost:~$ rosrun astra_camera astra_list_devices
[ INFO] [1500706596.935980289]: Device "2bc5/0401@3/7" found.
Found 1 devices:

Device #0:
Uri: 2bc5/0401@3/7 (Vendor: Orbbec, Name: Astra, Vendor ID: 2bc5, Product ID: 401)

Warning: USB events thread - failed to set priority. This might cause loss of data...
Serial number: 16030310030
```

2. Data Acquisition

1) After running $ roslaunch astra_launch astra.launch, Astra diver will be started. It will output topic according to ROS rules. Using rostopic to check if the data is normal:
   $ rostopic list

```
orbbec@localhost:~$ rostopic  list
/camera/camera_nodelet_manager/bond
/camera/depth/camera_info
/camera/depth/image
/camera/depth/image/compressed
/camera/depth/image/compressed/parameter_descriptions
/camera/depth/image/compressed/parameter_updates
/camera/depth/image/compressedDepth
/camera/depth/image/compressedDepth/parameter_descriptions
/camera/depth/image/compressedDepth/parameter_updates
/camera/depth/image/theora
/camera/depth/image/theora/parameter_descriptions
/camera/depth/image/theora/parameter_updates
/camera/depth/image_raw
/camera/depth/image_raw/compressed
/camera/depth/image_raw/compressed/parameter_descriptions
/camera/depth/image_raw/compressed/parameter_updates
/camera/depth/image_raw/compressedDepth
/camera/depth/image_raw/compressedDepth/parameter_descriptions
/camera/depth/image_raw/compressedDepth/parameter_updates
/camera/depth/image_raw/theora
/camera/depth/image_raw/theora/parameter_descriptions
/camera/depth/image_raw/theora/parameter_updates
/camera/depth/image_rect
/camera/depth/image_rect/compressed
/camera/depth/image_rect/compressed/parameter_descriptions
```

2) image_raw is released by Astra, and others are released after modifying rgbd_launch. Please focus on the three topics below to receive original camera information.
   /camera/rgb/image_raw
   /camera/depth/image_raw
   /camera/ir/image_raw
   According to the limitation of openni2, rgb and ir cannot output at the same time
   Please access to the two topic below to receive point cloud data.
   /camera/depth/points
   /camera/depth_registered/points
3) Checking camera information
   $ rostopic echo /camera/depth/camera_info

```
Cheader:
 seq: 226
 stamp:
   secs: 1500707483
   nsecs: 704247229
 frame_id: camera_depth_optical_frame
eight: 480
idth: 640
istortion_model: plumb_bob
: [0.0, 0.0, 0.0, 0.0, 0.0]
: [570.3422241210938, 0.0, 314.5, 0.0, 570.3422241210938, 235.5, 0.0, 0.0, 1.0]
: [1.0, 0.0, 0.0, 0.0, 1.0, 0.0, 0.0, 0.0, 1.0]
: [570.3422241210938, 0.0, 314.5, 0.0, 0.0, 570.3422241210938, 235.5, 0.0, 0.0,
0.0, 1.0, 0.0]
inning_x: 0
inning_y: 0
oi:
 x_offset: 0
 y_offset: 0
 height: 0
 width: 0
 do_rectify: False
--
```

4) Checking topic release rate

$ rostopic hz /camera/depth/image_raw

```
orbbec@localhost:~$ rostopic hz /camera/depth/image_raw
subscribed to [/camera/depth/image_raw]
average rate: 28.326
        min: 0.015s max: 0.050s std dev: 0.00682s window: 21
average rate: 27.338
        min: 0.015s max: 0.100s std dev: 0.01116s window: 48
average rate: 26.411
        min: 0.015s max: 0.100s std dev: 0.01197s window: 74
average rate: 26.985
```

5) Checking depth data（ir/rgb are similar）

$ rostopic echo /camera/depth/image_raw

```
 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0
 159, 2, 156, 2, 156, 2, 156, 2, 155, 2, 132, 2, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
4, 9, 161, 10, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0
 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 235, 5, 235, 5, 229, 5, 229, 5,
22, 5, 222, 5, 216, 5, 216, 5, 216, 5, 209, 5, 209, 5, 209, 5, 209, 5, 209, 5,
09, 5, 203, 5, 203, 5, 203, 5, 203, 5, 203, 5, 203, 5, 203, 5, 203, 5, 203, 5,
03, 5, 203, 5, 203, 5, 203, 5, 203, 5, 203, 5, 203, 5, 203, 5, 203, 5, 203, 5,
03, 5, 196, 5, 196, 5, 196, 5, 196, 5, 196, 5, 190, 5, 190, 5, 190, 5, 190, 5,
84, 5, 184, 5, 184, 5, 184, 5, 184, 5, 184, 5, 184, 5, 184, 5, 184, 5, 184, 5,
84, 5, 184, 5, 184, 5, 184, 5, 184, 5, 184, 5, 184, 5, 177, 5, 177, 5, 177, 5,
77, 5, 177, 5, 177, 5, 171, 5, 171, 5, 171, 5, 171, 5, 171, 5, 171, 5, 165, 5,
65, 5, 165, 5, 165, 5, 159, 5, 159, 5, 159, 5, 153, 5, 153, 5, 153, 5, 153, 5,
53, 5, 153, 5, 153, 5, 153, 5, 153, 5, 153, 5, 159, 5, 159, 5, 159, 5, 159, 5,
59, 5, 159, 5, 159, 5, 159, 5, 159, 5, 165, 5, 165, 5, 165, 5, 159, 5, 159, 5,
59, 5, 159, 5, 159, 5, 159, 5, 159, 5, 159, 5, 153, 5, 153, 5, 153, 5, 147, 5,
47, 5, 147, 5, 147, 5, 141, 5, 141, 5, 141, 5, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
0, 0, 0, 0, 0, 0, 0, 0, 0, 115, 11, 140, 11, 140, 11, 166, 11, 166, 11, 166, 11
 166, 11, 166, 11, 192, 11, 192, 11, 192, 11, 192, 11, 192, 11, 192, 11, 192, 1
, 192, 11, 192, 11, 192, 11, 219, 11, 219, 11, 219, 11, 0, 0, 219, 11, 219, 11,
```

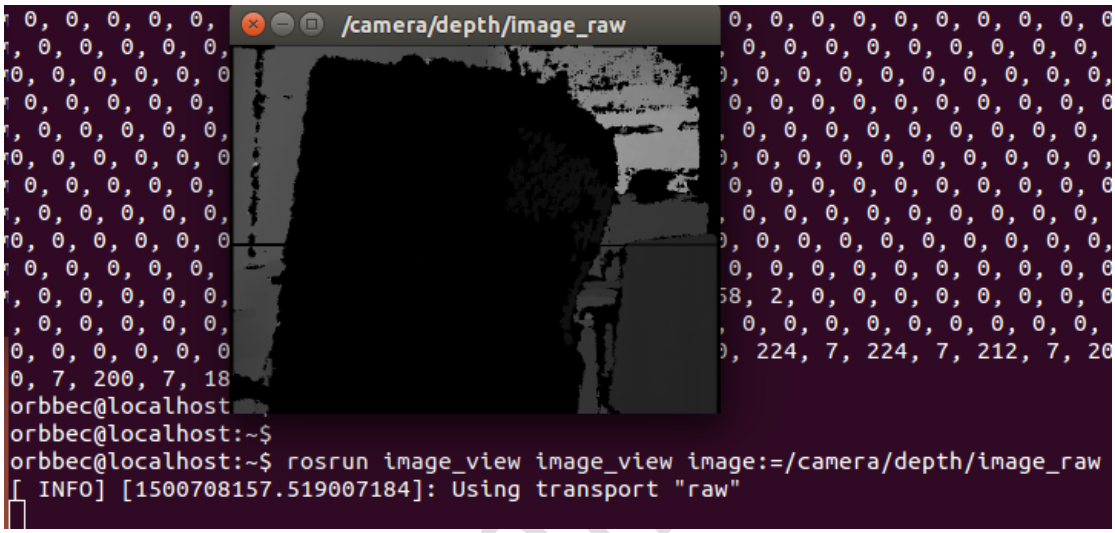
3. Image_view

Visually viewing the data released by camera by installing the image_view tool.

$ sudo apt-get install ros-indigo-image-view

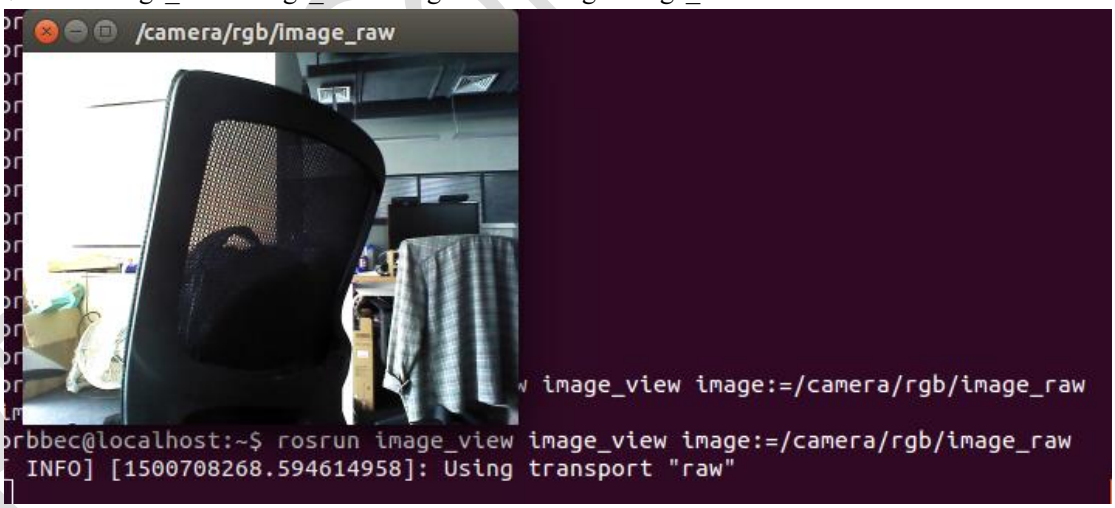
1) View depth map

$rosrun image_view image_view image:=/camera/depth/image_raw



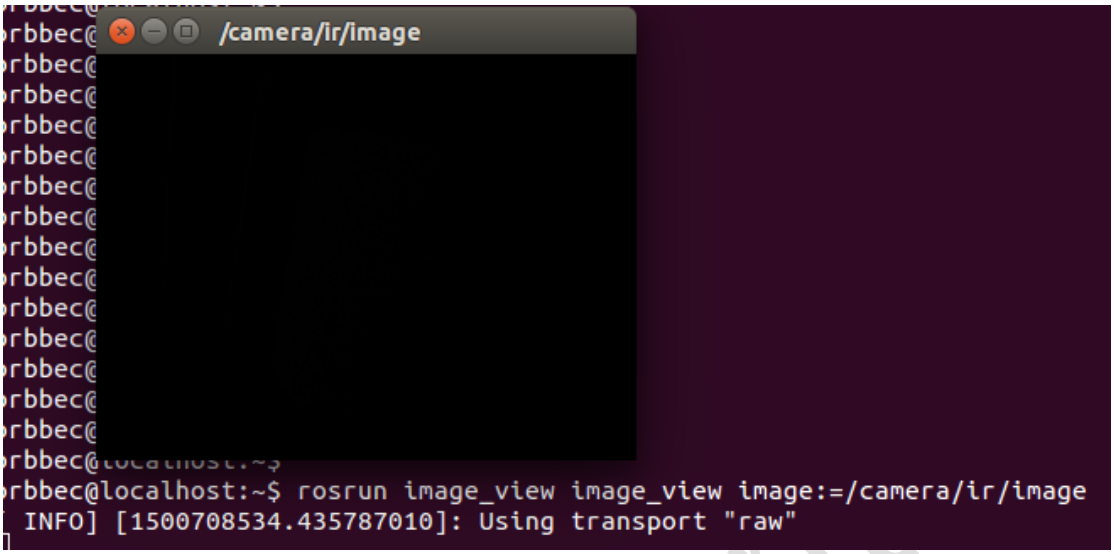
2) View color map

$rosrun image_view image_view image:=/camera/rgb/image_raw



3) View IR map

$rosrun image_view image_view image:=/camera/ir/image

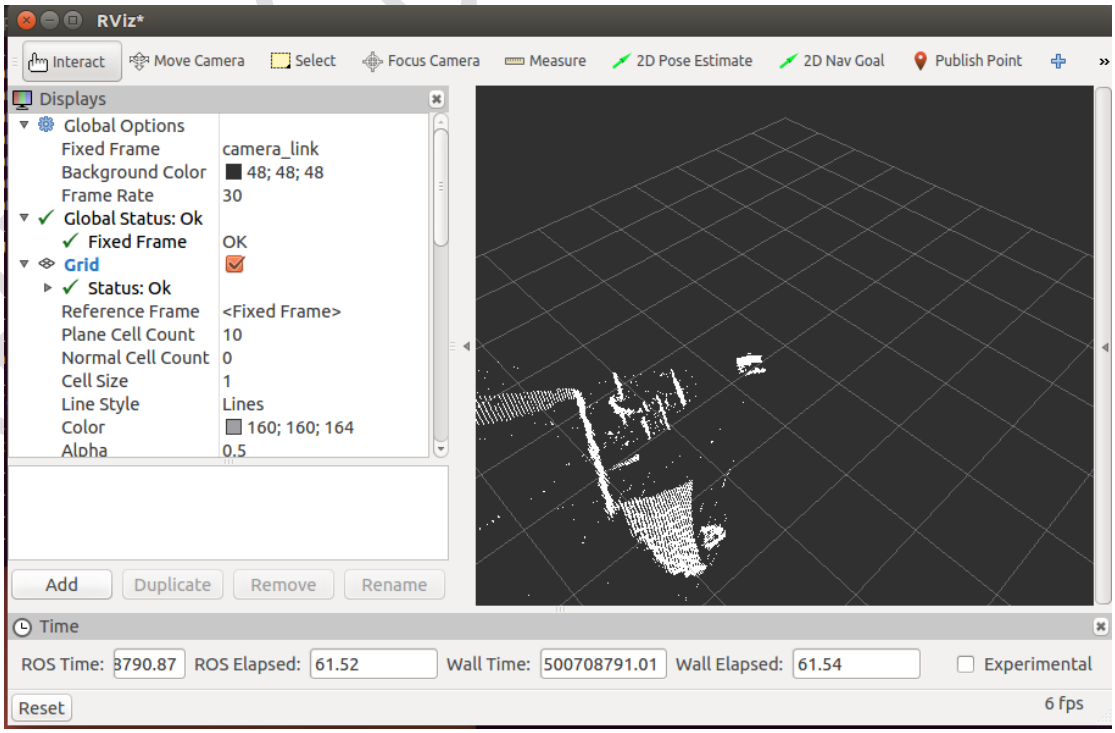


4. RVIZ Displays Depth Data

Rviz is a visual development tool under ROS. Rviz can load the image or camera of the previous step to realize visual analysis of image data. It can also display point cloud data directly.

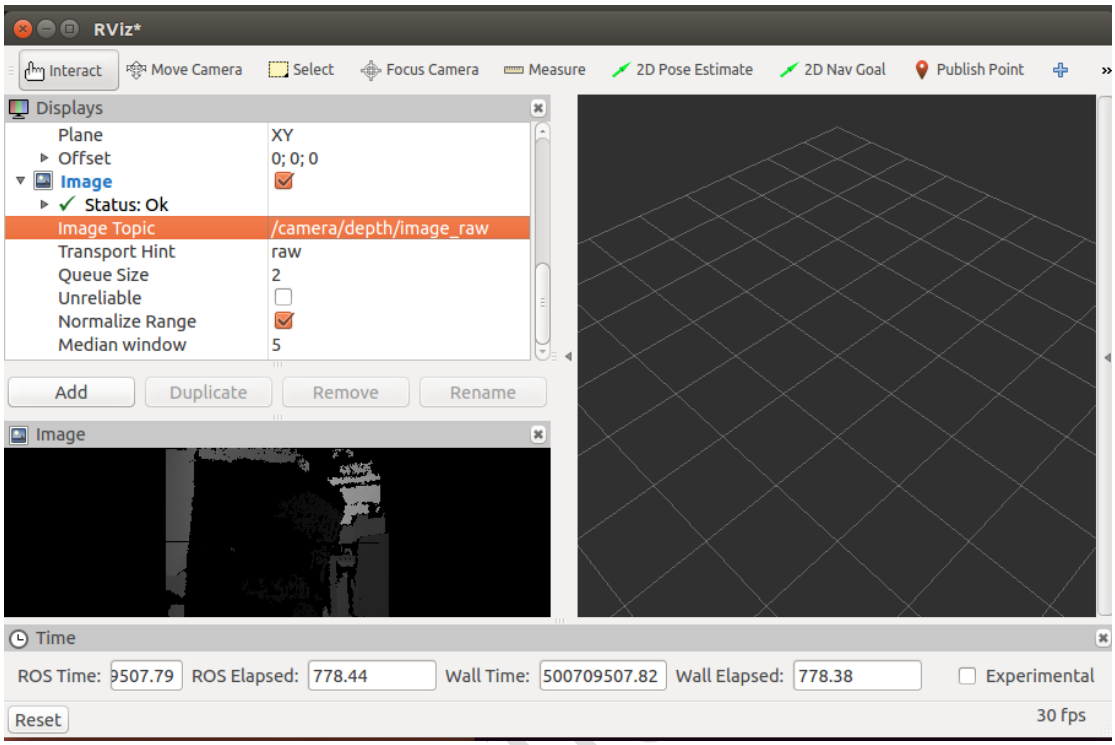
1) Run rviz

   $ rviz

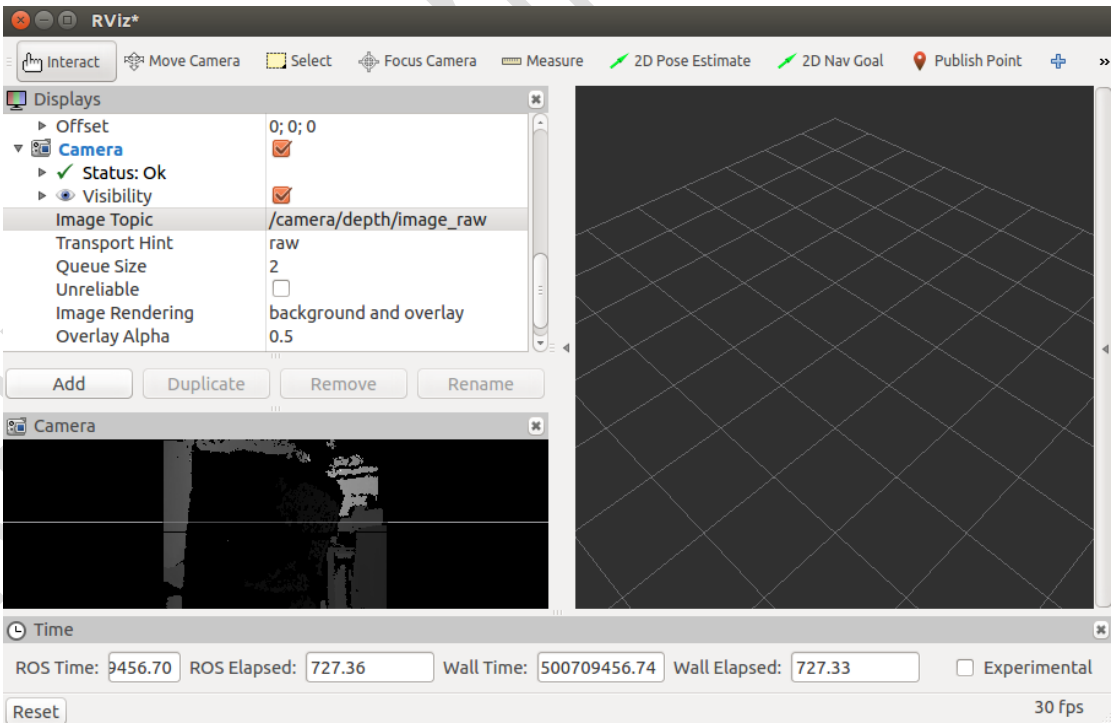


2) Add an image: Clicking Add -> Image; Specifying the topic as /camera/depth/image_raw in the image property page. Image window will show depth map. Similar method can be used to show color or IR map (set corresponding

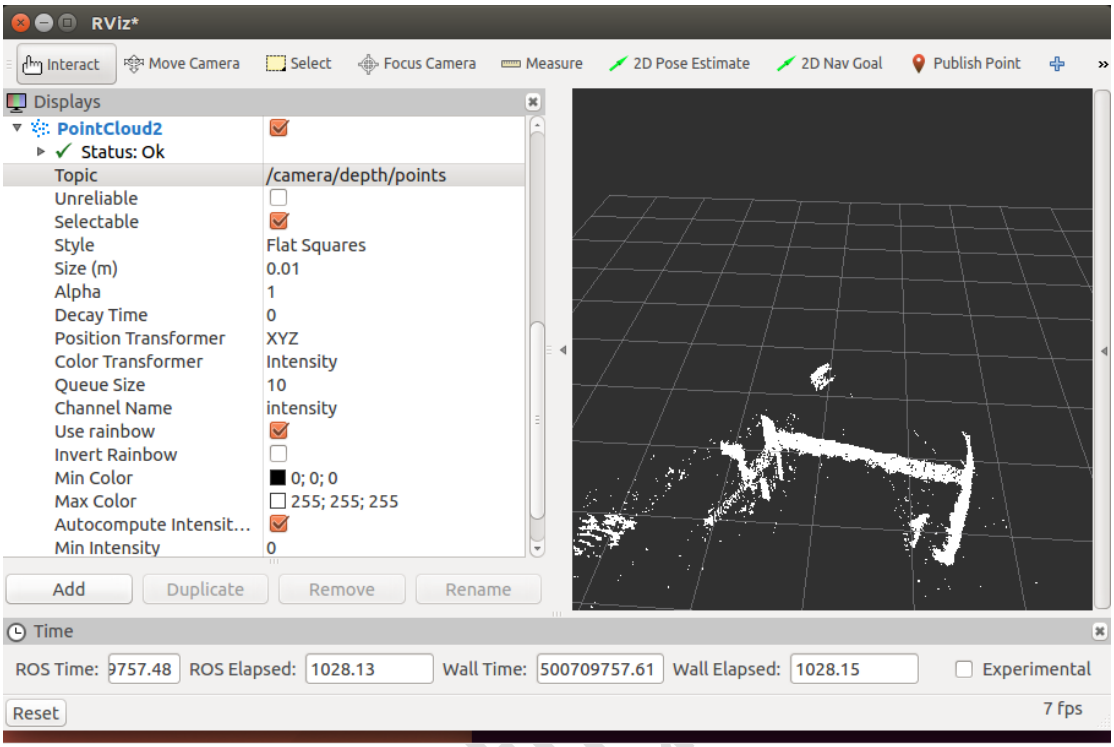
topic).



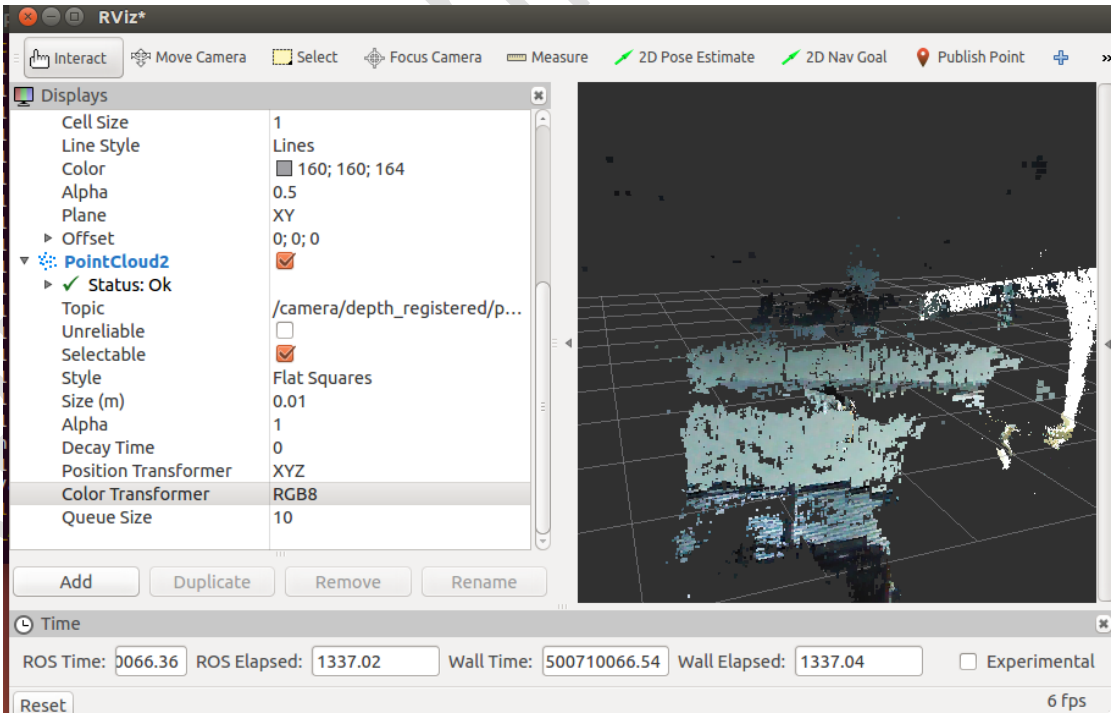
3) Camera is same as image, but some parameters of camera can be adjusted.



4) Show point cloud: Clicking Add -> PointCloud2, Setting topic to /camera/depth/points. Another Depthcloud in Add comes by conversing /camera/depth/image_raw from rviz. It's display effect is similar to PointCloud2.



5) Show color point cloud: Setting the topic of PointClound2 to /camera/depth_registered/points, and setting color transformer to rgb8.



5. Getting Color Data from Astra Pro Series

Astra pro series color camera obtain data by standard UVC interface instead of OpenNI interface. Since ROS Astra SDK cannot output UVC data directly, other methods should be

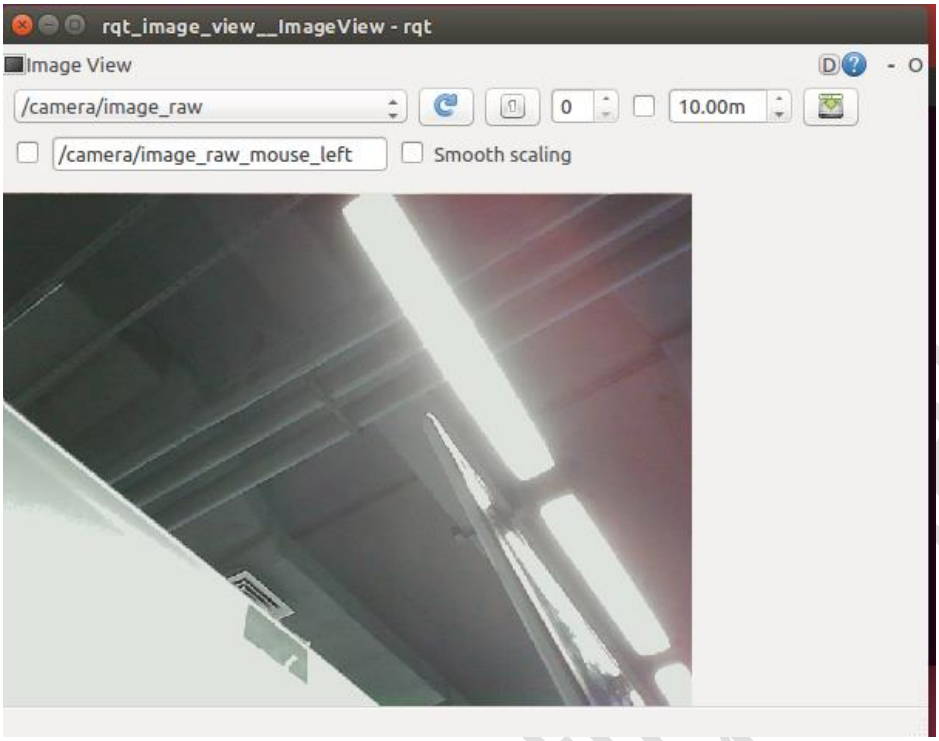
utilized to obtain color camera's data of Pro series. Here is a method which is based on ros of libuvc to implement libuvc_camera. (http://wiki.ros.org/libuvc_camera)

1) Implement the order below to view vid and pid for UVC devices.
   $ sudo apt-get install usbutils //install lsusb
   $ lsusb -v
2) Start UVC node
   $ sudo -E rosrun libuvc_camera camera_node _vendor:=0x2bc5 _product:=0x0501
3) Or obtain the USB access permission by modifying the UDEV file so that using the root to execute the permission is not necessary. For example: Add /etc/udev/rules.d/99-uvc.rules file and add the content below to ensure UVC device of vid=2bc5, pid=0501 can be visit by all users without root permission.
   # UVC cameras
   SUBSYSTEMS=="usb", ENV{DEVTYPE}=="usb_device",
   ATTRS{idVendor}=="2bc5", ATTRS{idProduct}=="0501", MODE="0666"
   # ^ Change the vendor and product IDs to match your camera.
4) Or start UVC camera through roslaunch.

```
<launch>
 <group ns="camera">
 <node pkg="libuvc_camera" type="camera_node" name="mycam">
 <!-- Parameters used to find the camera -->
 <param name="vendor" value="0x2bc5"/>
 <param name="product" value="0x0501"/>
 <param name="serial" value=""/>
 <!-- If the above parameters aren't unique, choose the first match: -->
<param name="index" value="0"/>
 <!-- Image size and type -->
 <param name="width" value="640"/>
 <param name="height" value="480"/>
 <!-- choose whichever uncompressed format the camera supports: -->
<param name="video_mode" value="uncompressed"/> <!-- or yuyv/nv12/mjpeg -->
<param name="frame_rate" value="30"/>


<param name="timestamp_method" value="start"/> <!-- start of frame -->
 <param name="camera_info_url" value="file:///tmp/cam.yaml"/>
<param name="auto_exposure" value="3"/> <!-- use aperture_priority auto exposure -->
 <param name="auto_white_balance" value="false"/>
 </node>
</group>
</launch>
```

5) Normally, data topic of UVC can be obtained through the above setting, and displayed by rqt_image_view or image_view.



6) If the UVC image cannot be obtained, libuvc.so may incompatible with libuvc_camera. In this case, please try to manually compile libuvc.so and libuvc_camera to solve the incompatible problem.
Libuvc : https://github.com/ktossell/libuvc
Libuvc_camera: https://github.com/ros-drivers/libuvc_ros