YDLIDAR HP60C-SDK _ROS DEVELOPMENT AND USE MANUAL

CONTENTS

1 INTRODUCTION

1.1 Product Overview

as_camera_node is the node used by ROS for HP60C camera module. It can publish depth, point cloud, IR, PEAK, RGB and other functions.

1.2 Environmental Requirement

1.2.1 Operating Environment

ARM/AARCH64, X86_64 architecture.

Run the SDK sample provided by the need to install the following software:

sudo apt - get the install cmake git g++.

1.2.2 System Requirement

Ubuntu16.04/Ubuntu18.04/Ubuntu20.04/Ubuntu22.04

2 SDK INTEGRATION GUIDE

2.1 Packet Structure Specification

├── CHANGELOG	SDK update record
├── docs	SDK/ROS develop usage instructions and other information sheets
└── sample	
├── linux_ros	
│ ├── linux	Reference example of simple call in linux system
│ ├── ros	ROS1 reference example under ros system
│ │ └── src	
│ │ └── as_nodelet	nodelet example
│ │ └── ascamera	Node example
│ │ ├── configurationfiles	Camera parameter configuration file
│ │ ├── include	Demo header file
│ │ ├── launch	ROS parameter configuration file
│ │ ├── libs	
│ │ │ ├── include	Library header file

```
|   |       |   └── lib                Library versions for each compiler
|   |       |       ├── aarch64-linux-gnu
|   |       |       ├── arm-linux-gnueabihf
|   |       |       └── x86_64-linux-gnu
|   |       └── src                  Demo source document
|   └── ros2                         ROS2 reference example in ROS system
└── windows                          Reference example of windows system
```

Among them, the structure of Linux is referred to "SDK Development and Usage Specification.pdf", and the structure of ros2 is referred to the structure of ros1, and the usage methods of the two can be referred to each other.

Version changes: from v1.2.10 version, after the beginning of the Linux/ros/ros2 demo provided are hot-swappable mechanism to realize reference example, equivalent to the previous version with xxx_listener reference example.

2.2 Example Program Running Instructions

2.2.1 Camera Configuration File Description

Camera configurationfile in the directory "configurationfiles", you need to upload the configurationfile path when you call the "Open camera" interface.

2.2.2 Run Ubuntu (Linux) System Sample Program

Go to the sample/linux_ros/linux directory, where build.sh is the compilation script and run_camera.sh is the script for running the sample program. Firstly, execute build. sh for compilation, and then execute the run_camera. sh script to run the program. Executing programs requires root privileges, and the script will run in sudo mode.

Compile:

./build.sh

Run:

./run_ascamera.sh

2.2.3 Run ROS System ROS1 Example Program

Go to sample/linux_ros/ros directory, where build.sh is the compilation script and run.ascamera. sh is the script for starting the node. First, run build.sh to compile and then run run_ascamera_node.sh to start the node. For details about how to use nodelet, see the ascamera node. Starting a node requires root permission, which is run in sudo mode in the script.

Compile:

./build.sh

Start node:

./run_ascamera_node.sh

Start nodelet:

./run_ascamera_nodelet.sh

2.2.4 Run ROS/ROS2 System Sample Program

Go to the sample/linux_ros/ros directory, where build.sh is the compilation script and run_camera_mode.sh is the script for starting the node. Firstly, execute build. sh for compilation, and then root execute the run_camera_dode. sh script to start the node. Starting a node requires root privileges, and the script will run in sudo mode.

Compile:
./build.sh
Start node:
./run_ascamera_node.sh

3 TOPIC DESCRIPTION

The node published the following topics, mainly including camera info, image, pointcoud, etc

3.1 TF Tree Topic

By default, TF tree topics are published, and users can set the pub_tfTree parameter in the launch configuration file/ros/src/ascamera/launch/xxx.launch. ROS1 refers to hp60c. launch and ROS2 refers to hp60c. launch. py.

When publishing tf tree topics, use RVIZ to view the image Fixed Frame and select the coordinates nodeNameSpace_camera_link_x from the dropdown menu nodeNameSpace_depth_x, nodeNameSpace_color_x, among them, nodeNameSpace_colorx is only applicable to RGBD cameras; When not posting tf topics, please fill in nodeNameSpace-ascamera_x for Fixed Frame. Where x corresponds to the camera's serial number, nodeNameSpace is the namespace of the current node.

Instructions for using RGBD cameras:

1) If the output color data is not aligned with the depth data, the camera coordinate system is "nodeNameSpace_camera_link_x", the depth data coordinate is "nodeNameSpace_depth_x", and the color data coordinate is "nodeNameSpace_color_x", which can be seen through the topic;
2) If the output color data is aligned with the depth data, the camera coordinate system is "nodeNameSpace_camera_link_x", and the coordinates of the depth data and color data are both "nodeNameSpace_color_x", that is, the depth data has been internally aligned with the color data, which can be seen through the topic.

3.2 Depth Image Camera Information Topic

Topic name:/xxx/depthx/camera info

The "x" in depthx represents the serial number of the camera found, such as "0", "1", and will not be repeated below.

Type: sensor_msgs: CameraInfo

3.3 Depth Image Topic

Topic name: /xxx/depthx/points
Type: sensor_msgs::Image

3.4 Point Cloud Topic Corresponding To Depth

Topic name: /xxx/depthx/points
Type: sensor_msgs::PointCloud2

3.5 RGB Image Information Topic

Topic name: /xxx/rgbx/camera_info
Type: sensor_msgs:CameraInfo

3.6 RGB Image Topic

Topic name: /xxx/rgbx/camera_info
Type: sensor_msgs::Image

3.7 IR Image Information Topic

Topic name: /xxx/irx/image
Type: sensor_msgs::Image

3.8 IR Image Topic

Topic name: /xxx/peakx/camera_info
Type: sensor_msgs:CameraInfo

3.9 PEAK Image Topic

Topic name: /xxx/peakx/image
Type: sensor_msgs::Image

4 LAUNCH FILE DESCRIPTION

4.1 ROS Launch File

Launch filename	Parameter	parameter specification
ascamera.launch (Default universal launch file)	confiPath	The path to the configuration file directory can be found by running ascamera node. sh
HP60C.launch	confiPath	Same as the general launch confiPath parameter
	depth_width	Set the width of camera depth data
	depth_height	Set the height of camera depth data
	rgb_width	Set the width of camera RGB data
	rgb_height	Set the height of camera RGB data
	fps	Set the frame rate of the camera
	color_pcl	Set whether to enable the color point cloud function
	pub_mono8	Set whether to post Mono8 topics
	pub_tfTree	Set whether to publish tf tree topics

[Note] To open multiple camera nodes in a launch file, refer to hp60c.launch and set the node name to different, you can receive the topic published by the corresponding node Deinitializate SDK.

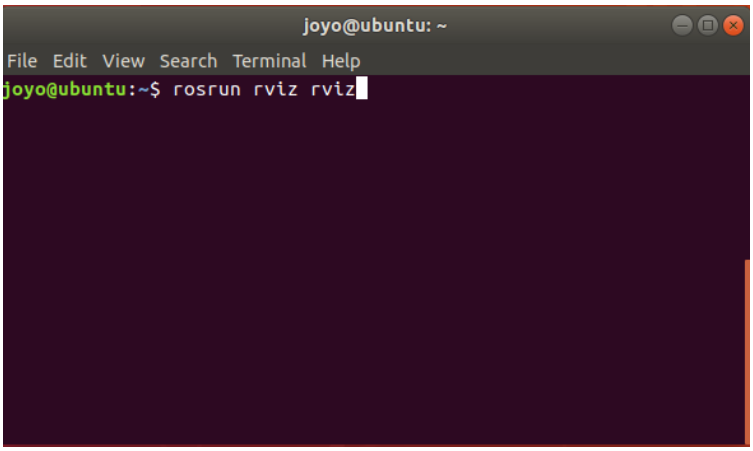
4.2 ROS2 Launch File

Launch filename	Parameter	parameter specification
ascamera.launch.py (Default universal launch file)	confiPath	The path to the configuration file directory
HP60C.launch.py	confiPath	The path to the configuration file directory
	color_pcl	Set whether to enable the color point cloud function
	usb_bus_no	Set to open the camera with the specified USB bus number (-1 is unrestricted)
	usb_path	Set to open the camera with the specified USB port number (null is unrestricted)
	depth_width	Set the width of camera depth data
	depth_height	Set the height of camera depth data
	rgb_width	Set the width of camera RGB data
	rgb_height	Set the height of camera RGB data
	fps	Set the frame rate of the camera

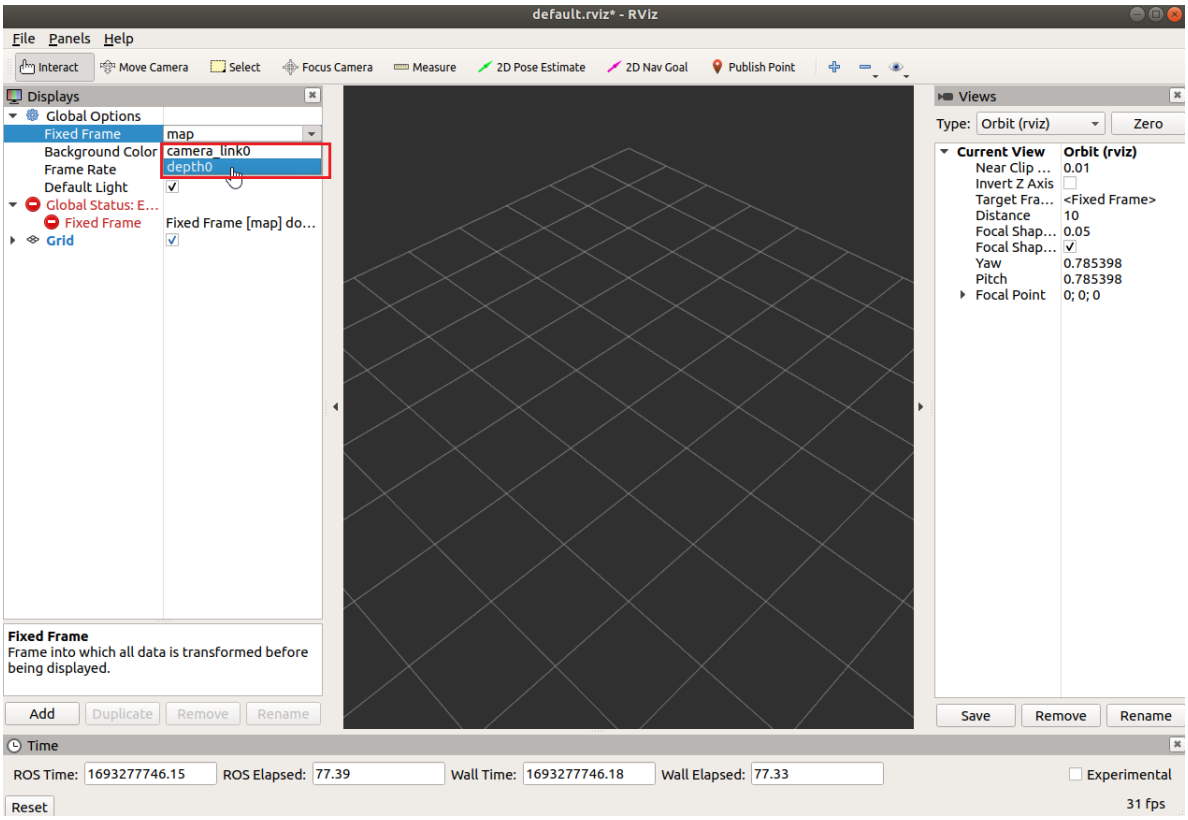
5 VIEW IMAGES USING RVIZ

Firstly, compile and run the ROS node according to the steps in 2.2.2. After the node is started, open rviz to receive and view images. The steps are as follows:

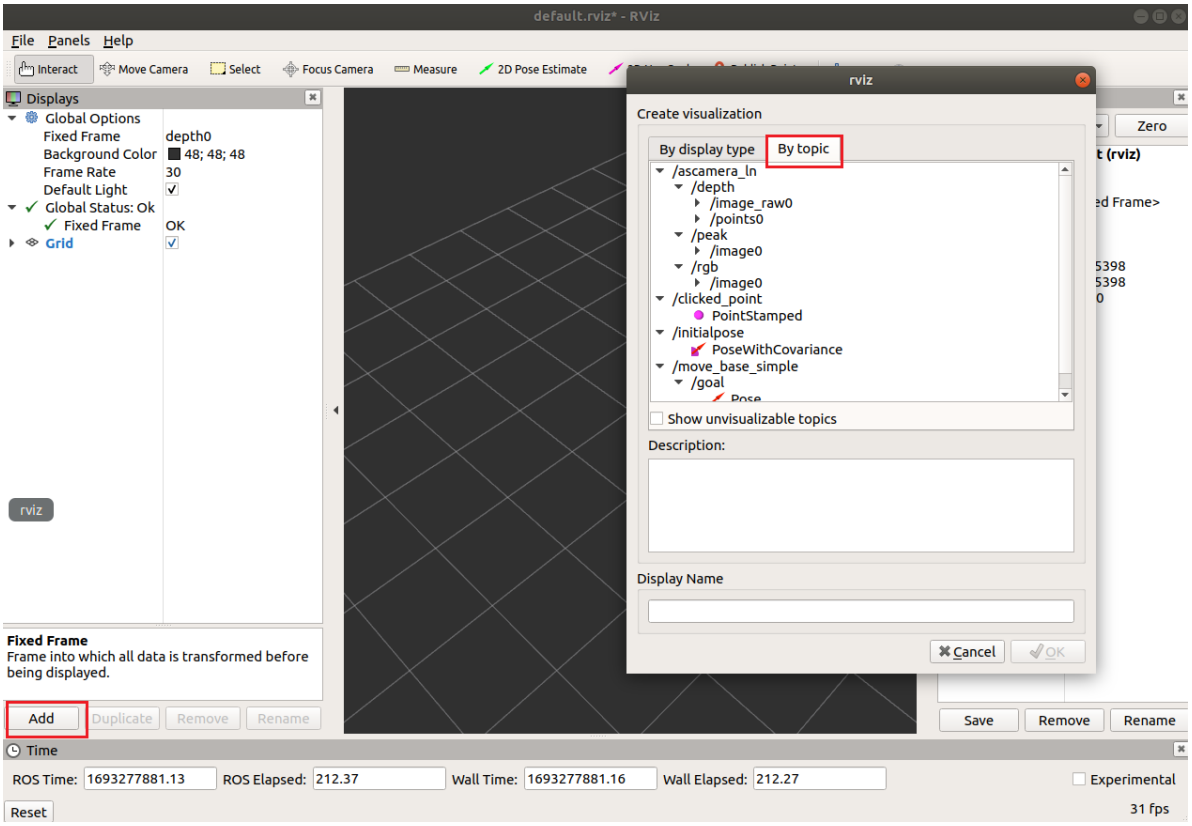
1) ROS1 starts the rviz node with the following command: (Under ROS2, run the command: ros2 run rviz2 rviz2)



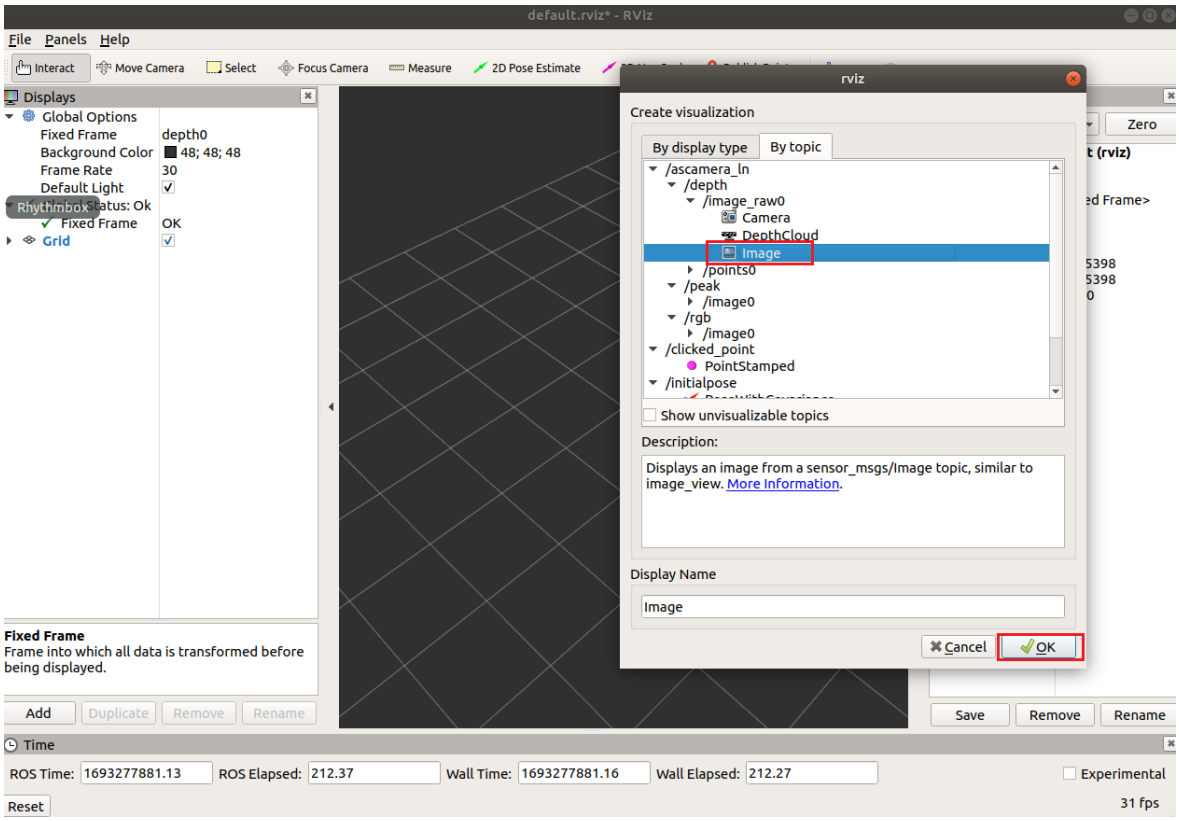
2) In the RVIZ after startup, select the Fixed Frame you need from the Global Options option in Display, as shown in the following figure:



3) To Add display options, click Add in the lower left corner of RVIZ and select By topic in the pop-up window to add the topic you want to display.

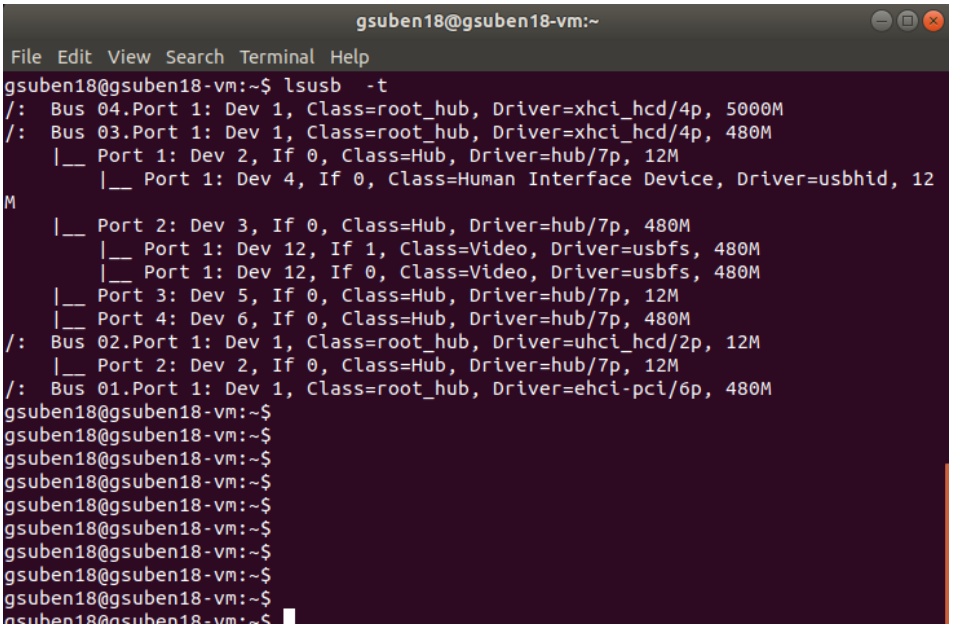


4) Taking viewing depth as an example, select Image under/xxx/depth/imageraw0 and click OK to add this topic to the display window.



6 OBTAIN DEVICE PATH INFORMATION

After the device is mounted, run lsusb -t to view node information. As shown in the picture below:



Among them, Bus corresponds to usd_busyno in the launch file, and the port at the next level of Bus corresponds to usd_path in the launch file. It should be noted that there are several layers of ports that need to be written completely. For example, in the above image, when the camera is turned on, usd_bus_2o is set to "3" and usd_path is set to "2.1".

7 ERROR CODE SPECIFICATION

Error code	Description
-80	Libuvc initialization failed
-81	Failed to obtain camera list
-82	Opening camera failed, the camera does not exist
-83	Parameter input error, camera handle does not exist
-84	Closing camera failed, camera not initialized or not present
-85	Opening image stream failed, unsupported image type
-86	Open image stream failed, Libuvc open stream error
-87	Failed to close image stream, Libuvc failed to close stream
-88	XU communication failed, setup request failed
-89	XU communication failure, acquisition request failed
-90	Failed to obtain USB device information

8 FAQ

8.1 How To Run Ros Nodes Without Root Privileges

In Linux systems, accessing devices requires root privileges. For programs that operate devices by operating device nodes in the/dev directory, one way is to modify the permissions of device nodes through the chmod command. However, this is only temporary and cannot be changed through chmod for programs that do not operate devices through the/dev device node. At this point, device permissions can be managed through Linux's udev and rules rules rules.

For Linux systems using the HP60C camera module, access to the HP60C camera can be achieved through running the ROS node with normal permissions by executing the create_udev_rules.sh script in the sample/linux_ros/ros/src/ascamera/scripts directory. Modify the run_camera'node.sh script in the sample/linux_ros/ros/directory to annotate the statement that checks and applies for root permission. As shown in the following figure:

```
18
19  CUR_DIR="$(dirname "$(realpath "${BASH_SOURCE[0]}")")"
20  # check for whitespace in $CUR_DIR and exit for safety reasons
21  grep -q "[[:space:]]" <<<"${CUR_DIR}" && { echo "\"${CUR_DIR}\" contains whitespace. Not supported. Aborting." >&2 ; exit 1 ; }
22
23  # if [ $EUID -ne 0 ]; then
24  #     echo -e "${RED}---This script requires root privileges, trying to use sudo${NORMAL}"
25  #     sudo "$CUR_DIR/run_ascamera_node.sh" "$@"
26  #     exit $?
27  # fi
28
29  if [ -f /opt/ros/melodic/setup.bash ]; then
30      source /opt/ros/melodic/setup.bash
31  elif [ -f /opt/ros/kinetic/setup.bash ]; then
32      source /opt/ros/kinetic/setup.bash
33  elif [ -f /opt/ros/noetic/setup.bash ]; then
34      source /opt/ros/noetic/setup.bash
35  else
36      echo "Error,Can't not found ros in /opt/"
37  fi
38
39  if [ "$(ps -aux | grep rosmaster | grep -v grep | wc -l)" -eq "0" ]; then
40      roscore &
41      sleep 3
42  fi
```

8.2 How To Run Ros2 Nodes Without Root Privileges

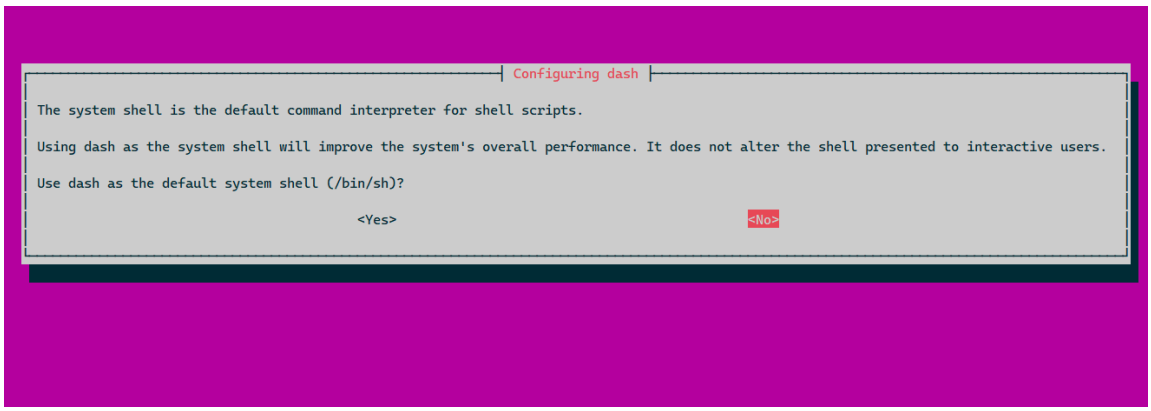
In Linux systems, accessing devices requires root privileges. For programs that operate devices by operating device nodes in the/dev directory, one way is to modify the permissions of device nodes through the chmod command. However, this is only temporary and cannot be changed through chmod for programs that do not operate devices through the/dev device node. At this point, device permissions can be managed through Linux's udev and rules rules rules.

For Linux systems using the HP60C camera module, the HP60C can be accessed by running the create_udev_rules.sh script in the sample/linux_ros/ros2/src/scamera/scripts directory with normal permissions by running the ROS2 node.

8.3 Executing Script Errors Provided By SDK

Execute the script provided by the SDK with errors such as Bad substitution or syntax error: redirect unexpected. This is because the shell script provided by the SDK is a bash shell, while the default shell parser of the Ubuntu system is dash. Bash is more powerful than Dash, and some Bash syntax may not be parsed by Dash. You can change the default shell of ubuntu to bash by following these steps.

Execute sudo dpkg configure dash on the terminal and select NO in the pop-up window.



8.4 Compilation Error On Ubuntu 20.04/22.04 ROS

The following error was reported during ROS (Noetic or higher) compilation in Ubuntu 20.04/22.04.

Due to the fact that our ROS node example program is developed based on the mellodic version of ROS on Ubuntu 18.04, when porting to ROS Noetic, the - std=c++11 in the ROS/src/ascamera/CMakeLists.txt file can be changed to - std=c++14 or - std=c++17.

```
                 from /opt/ros/noetic/include/ros/time.h:58,
                 from /opt/ros/noetic/include/ros/ros.h:38,
                 from /home/boocax/roswork/src/as_nuwacam/src/as_nuwacam_node.cpp:29:
/usr/include/boost/mpl/aux_/preprocessed/gcc/minus.hpp:68:8: note:   'boost::mpl::minus'
   68 | struct minus
      |        ^~~~~
In file included from /usr/include/pcl-1.10/pcl/point_types.h:44,
                 from /usr/include/pcl-1.10/pcl/common/impl/copy_point.hpp:41,
                 from /usr/include/pcl-1.10/pcl/common/copy_point.h:58,
                 from /usr/include/pcl-1.10/pcl/common/impl/io.hpp:45,
                 from /usr/include/pcl-1.10/pcl/common/io.h:586,
                 from /usr/include/pcl-1.10/pcl/io/file_io.h:41,
                 from /usr/include/pcl-1.10/pcl/io/pcd_io.h:44,
                 from /opt/ros/noetic/include/pcl_conversions/pcl_conversions.h:70,
                 from /home/boocax/roswork/src/as_nuwacam/src/as_nuwacam_node.cpp:38:
/usr/include/pcl-1.10/pcl/point_types.h:698:1: error: 'minus' is not a member of 'pcl::traits'
  698 | POINT_CLOUD_REGISTER_POINT_STRUCT (pcl::_PointDEM,
      | ^~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~
/usr/include/pcl-1.10/pcl/point_types.h:698:1: note: suggested alternatives:
In file included from /usr/include/c++/9/string:48,
                 from /usr/include/c++/9/bits/locale_classes.h:40,
                 from /usr/include/c++/9/bits/ios_base.h:41,
                 from /usr/include/c++/9/ios:42,
                 from /usr/include/c++/9/ostream:38,
                 from /usr/include/c++/9/iostream:39,
                 from /home/boocax/roswork/src/as_nuwacam/src/as_nuwacam_node.cpp:21:
/usr/include/c++/9/bits/stl_function.h:177:12: note:   'std::minus'
  177 |     struct minus : public binary_function<_Tp, _Tp, _Tp>
      |            ^~~~~
```

sample > ros > src > ascamera > CMakeLists.txt

```
1   cmake_minimum_required(VERSION 3.0.2)
2   project(ascamera)
3
4   ## Compile as C++11, supported in ROS Kinetic and newer
5   add_compile_options(-std=c++11)
6   add_definitions(-std=c++11)
7   # get gcc -v target
```

8.5 Using RVIZ2 On ROS2, No Published Topics Found

Reason: When the 8.2 method is not executed, the script will run the program with root privileges, resulting in the inability to subscribe to the topics published by the program when running rviz2 without root privileges on Ubuntu 22.04 or some versions.

Solution: Follow the method in 8.2 above.

8.6 Depth Camera Matching Exception Problem In Virtual Machine

Due to the HP60C camera being composed of 2 USB devices combined into 1 camera (1 USB communication device+1 UVC device), matching is required. When running on a virtual machine, USB devices may not pair properly due to incorrect device topology. Therefore, when using a virtual machine to run a program, only one HP60C camera can be connected, and more than one camera may cause matching failure due to the above reasons.

8.7 Multiple Cameras Run Abnormal Flow

One possible reason for this issue is that the system kernel parameter usbfs_memory_mb is set too small. Usbfs_memory_mb is a kernel parameter used to specify the memory size used by the USB file system (usbfs). USBFS is a virtual file system used to transfer USB device data between user space and the kernel. This parameter specifies the memory size, in MB, allocated by usbfs to the USB device communication buffer. By default, this value is 16 MB. By increasing or decreasing this value, you can adjust the memory size of the USB device communication buffer.

Temporary modification method (will fail after system restart): Increase this kernel parameter to 64/128/512 or higher.

Command: echo "64" | sudo tee - a/sys/module/usbcore/parameters/usbfs memory_mb

The permanent modification method can be set by customers according to the system they are using.