1. ROS camera driver

1.1. What is ROS?

The Robot Operating System (ROS) is a set of software libraries and tools that help you build robot applications. From drivers to state-of-the-art algorithms and with powerful developer tools, ROS has what is needed for a robotics project. It is all open source (Source: ROS.org). For more details, also refer to ROS.org and ROS Wiki sensors.

1.2. Building the ROS driver

Extract zip file for example in **`~/projects/`** directory

Change current directory:

```
> cd ~/projects/cam660_driver
```

Build a project from the command:

```
> catkin_make
```

Change to the home directory and open the bash-file:

```
> cd ~
> gedit .bashrc
```

Insert the following line at the end of the bash-file:

```
source ~/projects/cam660_driver/devel/setup.bash
```

Save the file and exit editor.

Log-out and again log-in linux or execute command:

```
> source ~/projects/cam660_driver/devel/setup.bash
```

1.3. Running the ROS driver with launch file

Start the ROS with GUI in terminal mode with the following command:

```
roslaunch espros_cam660 camera.launch
```

The ROS tool opens with the different node windows and is ready to use.

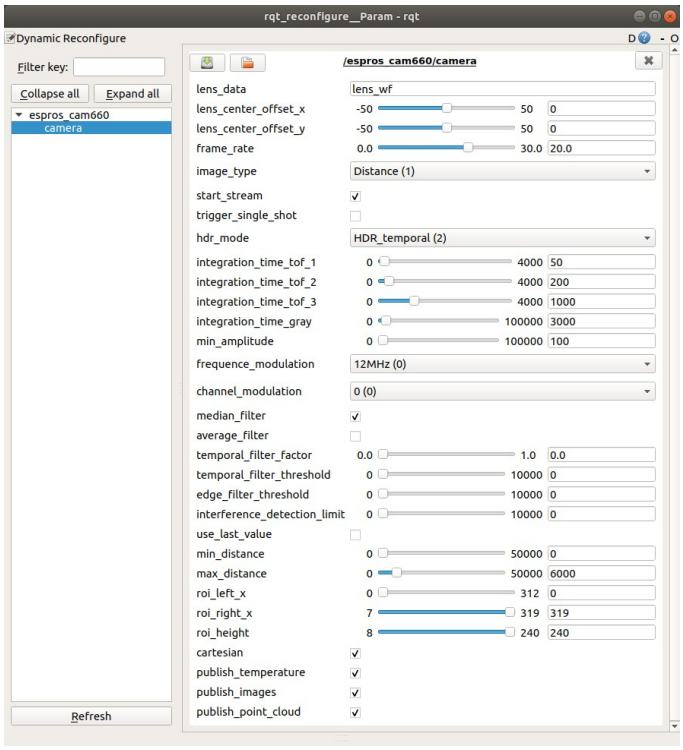

Figure 1: Example of the “dynamic reconfigure” node window

Start the camera operation by changing in the menu the parameter “start_stream” from false to true.

2. ROS API

This is the official driver for the camera ESPROS TOFcam660.

2.1. Start of the node

If you use in terminal mode the APIs only, without GUI:
Start the ROS operating system with the command: `roscore&`
Start the TOFcam660 with the command: `rosrun espros_cam660 cam660_node`

2.2. Published topics

Topic name	ROS msgs file	ROS message type	Function
camera/image_raw1	sensor_msgs	Image	Sends the grayscale or amplitude image according the selected image type parameter
camera/image_raw2	sensor_msgs	Image	Sends the distance image for image type parameters which include distance
camera/temperature	sensor_msgs	Temperature	Sends temperature of sensor
camera/points	sensor_msgs	PointCloud2	Sends the point cloud image for image type parameters which include distance

Table 1: ESPROS ROS topics

2.3. Lens distortion table

TOF cameras are measuring the distance between the camera (center of the lens' front-surface) and the object (surface reflected by Lambertian reflector) by estimating the time-of-flight (TOF) forth and back of the light emitted by the camera itself. Due to this principle, the obtained measurement data are given by polar coordinates, whereas distance D is the vector length, and the vector position in the sphere is given by the individual, effective view of each pixel through the lens, depending of the field of view (FOV).

In many cases, the user needs the distance information in a vector format which is locatable in a cartesian coordinate system.

Camera driver makes polar - cartesian transformation based on the lens data from the lens manufacturer. Lens data are written in lens distortion file which is in /lensData directory. The file name is optional and is written as string parameter ~lens_data.

Lens data file consist of two arrays and format is as follows:

size = 101
angle
0.00
0.74
1.48
2.22
2.95
…

radius
0.00
0.05
0.10
0.14
0.19
...

Where: size – the size of array

angle – in grad

radius – lens circle projection radius on the sensor mm.

2.4. Dynamically reconfigurable parameters

Refer for details on the dynamically reconfigurable parameters to the enclosed “dynamic_reconfigure package” or to http://wiki.ros.org/dynamic_reconfigure.

Parameter	Function	Data format	Default	Reference
~lens_data	Sets the lens data file name for point cloud cartesian transformation. This file must be in /lensData directory.	string	0	n/a
~lens_center_offset_x	Sets the offset of lens center relative to sensor center Range: -50 … 50 pixels	int	0	n/a
~lens_center_offset_y		int	0	n/a
~image_type	Sets the image acquisition type 0: Grayscale 1: Distance 2: Distance and amplitude	int	1	n/a
~start_stream	Enables image streaming	bool	True	n/a
~integration_time_tof_1	Sets the integration time for distance measurements in microseconds. Range: 1 … 4'000 µs	int	50	n/a
~integration_time_tof_2		int	400	
~integration_time_gray	Sets the integration time for grayscale measurements in microseconds. Range: 0 … 50'000 µs	int	3000	n/a
~offset	Sets distance offset. Range: -10000 … 10000 mm	int	0	n/a
~min_amplitude	Sets the amplitude limits. Range 0 … 2'047 LSB	int	100	
~frequency_modulation	Sets camera frequency modulation. Range: 24 … 0.75 MHz	int	0	n/a
~channel	Sets camera frequency modulation offset.	int	0	n/a
~median_filter	Enables the spatial median filter for distance filtering	bool	False	n/a
~average_filter	Enables the spatial average filter for distance filtering	bool	False	n/a
~temporal_filter_factor	Sets the factor 'k' of the temporal filter (Kalman). Range: 0.0 ... 1.0	double	0	n/a
~temporal_filter_threshold	Sets the threshold of the temporal filter (Kalman). Range: 0 ... 10000 mm			
~edge_filter_threshold	Spatial edge filter threshold. Range: 0 … 10000 mm	int	0	n/a
~temporal_edge_filter_threshold_low	Temporal edge filter low threshold. Range: 0 … 10000 mm	int	0	n/a
~temporal_edge_filter_threshold_high	Temporal edge filter high threshold. Range: 0 … 10000 mm	int	0	n/a
~interference_detection_limit	Interference detection threshold. Range 0... 10000 mm	int	0	n/a
~use_last_value	Enables interference detection last value	bool	False	n/a
~enable_cartesian	Enables point cloud cartesian transformation (false = spheric)	bool	True	n/a
~enable_point_cloud	Activates pointCloud2Publisher node to send information (camera/points)	bool	True	Table1

Table 2: ROS parameter table