DreamVu

PAL Ethernet Documentation

To contact DreamVu, please drop an email at:
For general queries - info@dreamvu.com
For technical support - support@dreamvu.com

DreamVu Inc.
3675 Market Street, Suite 200, Philadelphia, PA, 19104
www.dreamvu.com

Table of Contents

Revision History

Revision	Author	Date	Status and Description
1	Piyush	10/03/2020	• Updated Firmware APIs to v1.2 • Updated Explorer Section

1. Introduction and Background

DreamVu's PAL is the world's first 360° stereo and depth sensor capable of providing real time situational awareness to autonomous machines. Its innovative binocular mirrored optics enables the capture of 360° environment in stereo. You can capture the following information in real time using the APIs provided in this document:

- Stereo panoramas (left and right views of a stereo system)
- Disparity Image
- Depth-map
- Point cloud

The objective of this document is to serve as an all in one guide for the end user. This document explains installation procedure, includes tutorials on how to use the API, provides an API reference and documents the known issues & troubleshooting procedures.

Note: Unless stated otherwise, all the relative paths are with respect to the directory where PAL_SDK is extracted.

Figure 1.2: Front view of PAL Ethernet

2. System Requirements

The PAL SDK is available for Linux and is supported by OpenCV and ROS platforms. Please ensure you have the following system specifications in order to run the SDK smoothly:

2.1 Hardware Requirements:

- Processor: Intel/Arm based architecture
- Ram: 2 GB
- Gigabit Ethernet port

2.2 Software Requirements:

- Operating System: Ubuntu 18.04 64 bit.
- OpenCV 3.4.4 libraries.
- Gstreamer & Curl libraries

3. Installation Procedure

3.1 Installing the PAL libraries

- Extract the PAL SDK compressed file.
- Open a terminal with the extracted folder as the current working directory.
- Run the following commands

```
cd installations
chmod +x ./*.sh
sudo ./install.sh
```

3.2 Installing Dependencies

PAL API depends on OpenCV libraries

- Follow **opencv.sh** script present in the *installations* folder of SDK.

3.3 Establishing Ethernet Connection between PAL and Host

- As part of the above install script a static ip ethernet connection named **DirectConnect** will be created which will be used to communicate with PAL.

Note: Please ensure that whenever PAL is connected to your system *DirectConnect* connection is activated.

4. Quick Start

4.1 Explorer Application

Once the Host SDK is installed following the steps in the previous section, you can quickly get started with exploring the features of the camera by using the Explorer Application provided in the Explorer folder `./Explorer/`. This application allows you to preview the live feed of the stereo panoramas & disparity map, record the videos and access or change the resolution, and camera parameters . You can also use this application to check if the camera is connected properly and is being detected by the API backend.

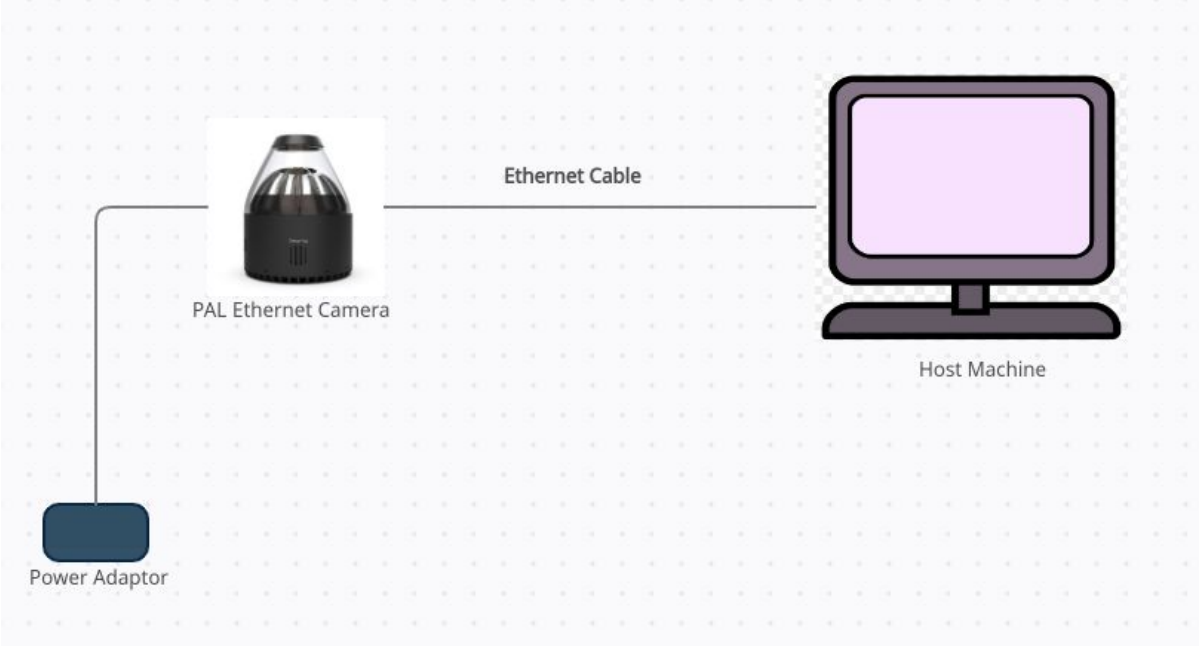


Figure 4.1: PAL Ethernet Camera setup

- Connect PAL Ethernet Camera to power through the provided adapter and

then Plug it in to an Ethernet port on the host computer. Ensure that the active network connection is switched to **DirectConnect** on the host system.

- Please wait at least for 2-3 minutes for the camera resources to be initialized.
- Open Explorer directory and launch Explorer Application. See below sections for more details about the application.
- Explorer when loaded for the first time initialises the sensor with default properties which can be changed in the application. Please set sensor and camera properties in the application to appropriate values which are best suited for your scene.

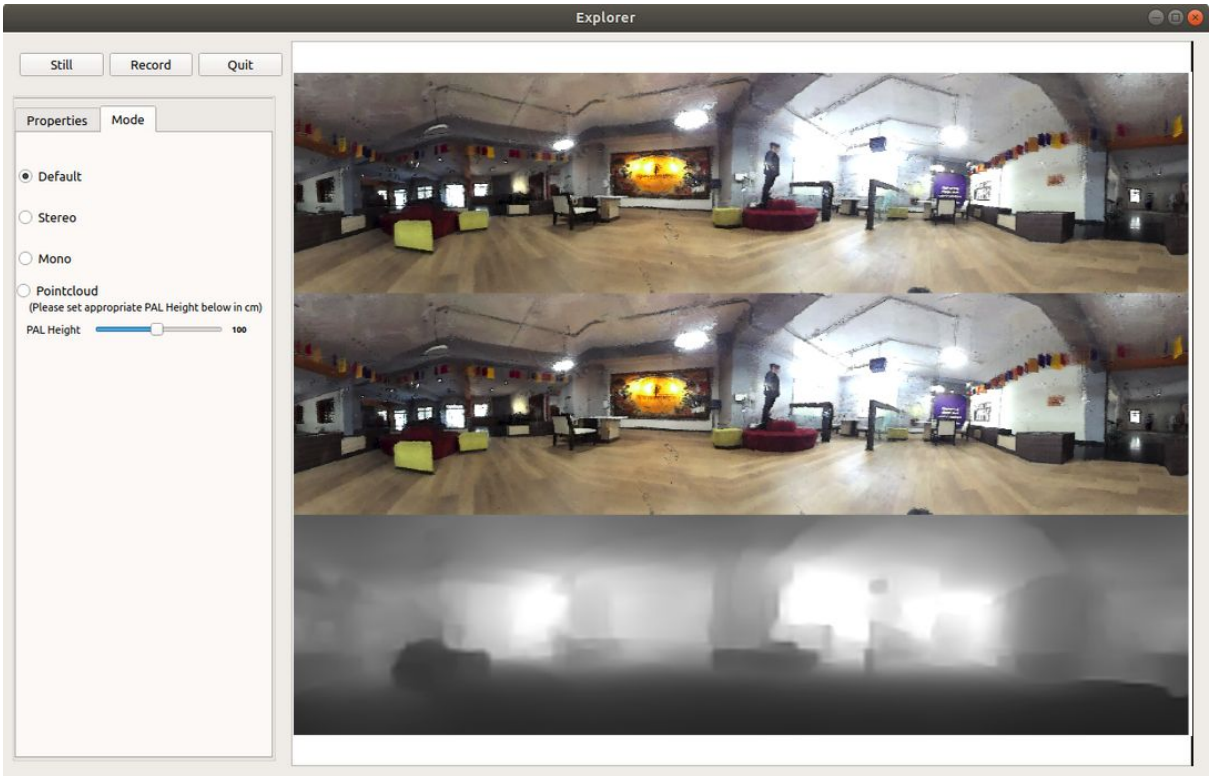


Figure 4.2: Default view of the explorer

Let us explore the features of the Explorer in the following sections in more detail.

4.1.1 Viewing modes

There are four viewing modes in the Explorer to switch between them you first

click on the “Mode” tab highlighted in the image below:

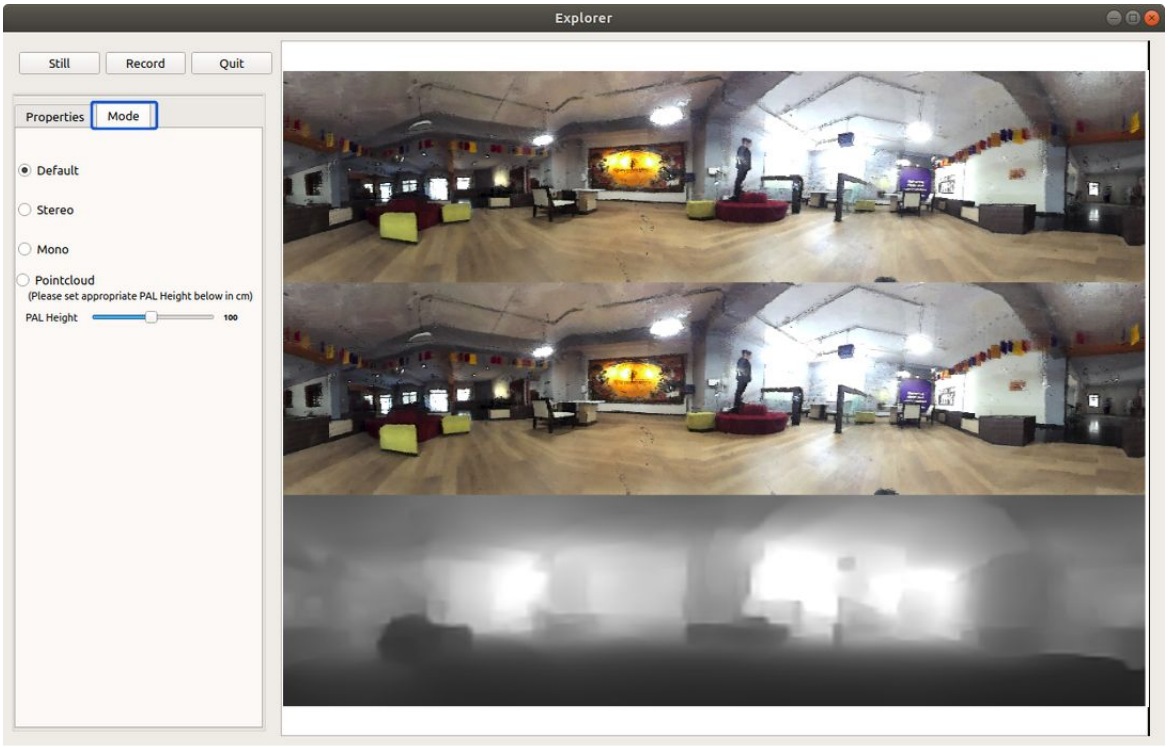


Figure 4.1.1: Clicking the mode tab

This will show the mode pane where you can choose between the following modes:

1. Default mode
2. Stereo mode
3. Mono mode
4. Pointcloud mode

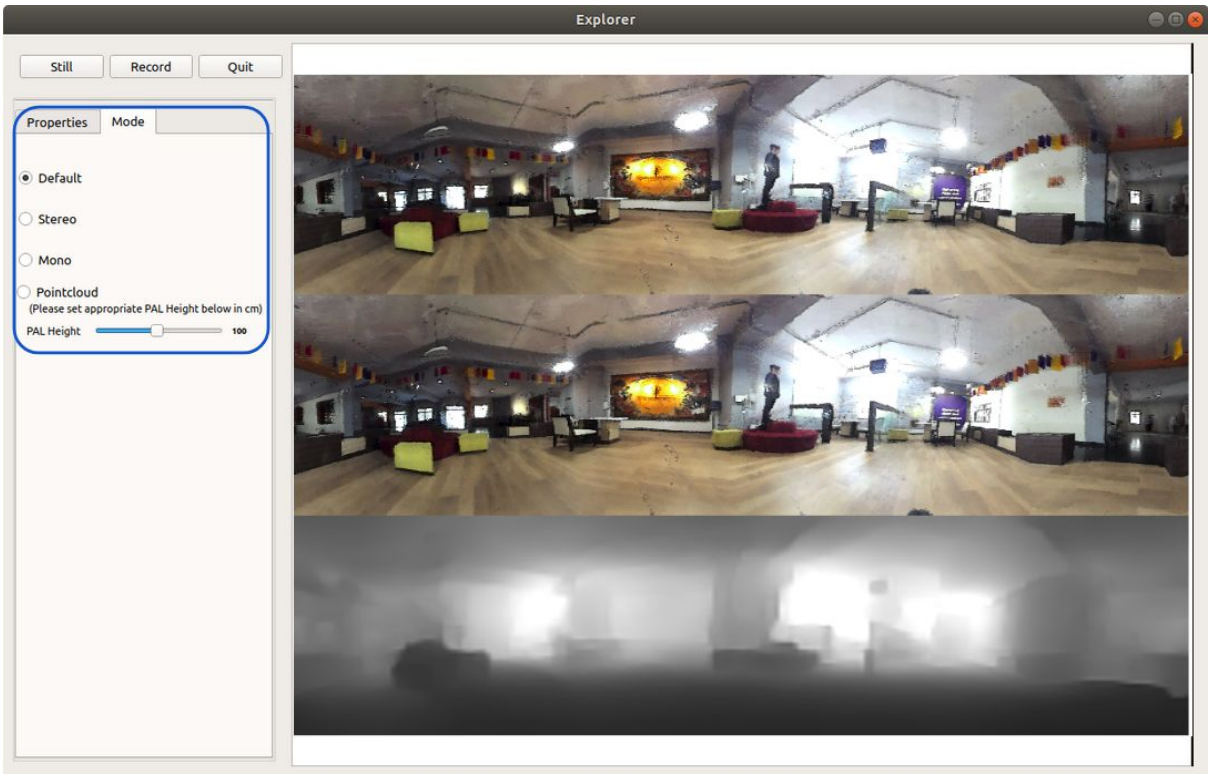


Figure 4.1.3: Mode pane highlighted

In the default mode, you will be shown the left, right and disparity panoramas in the order shown below:

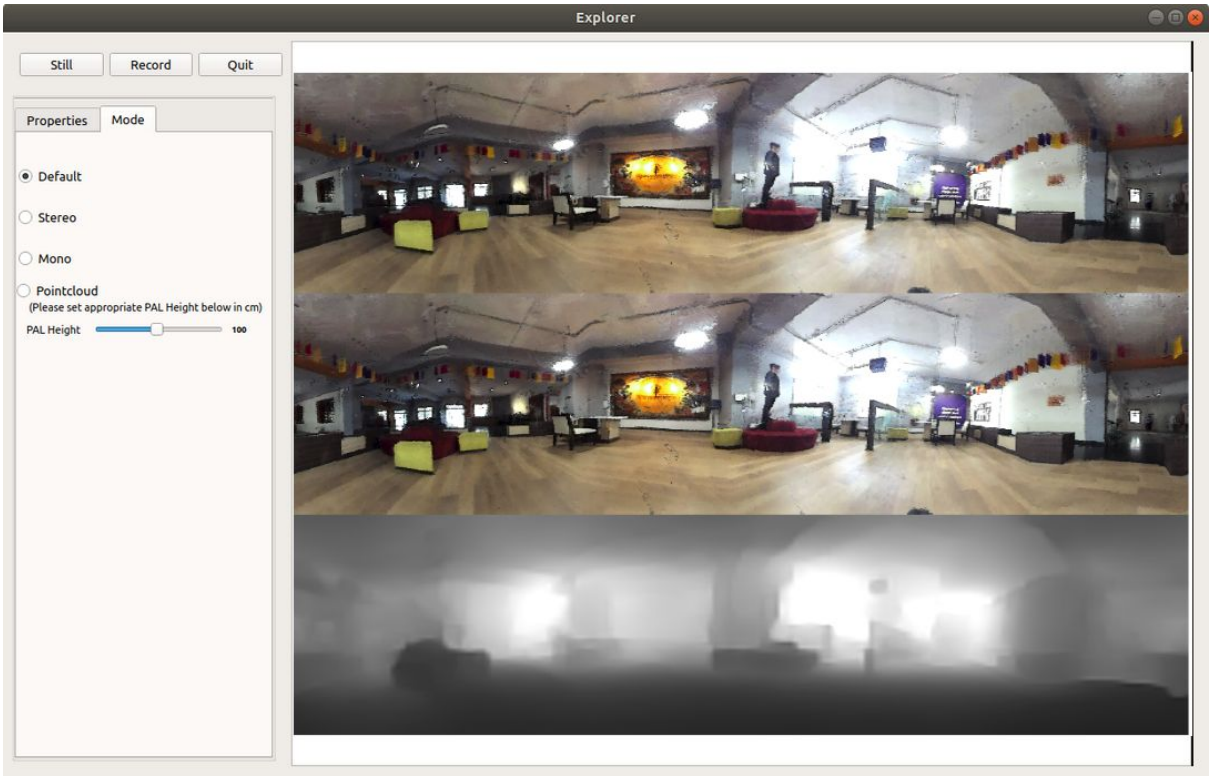


Figure 4.1.4: default mode

In the Stereo mode, the left and right panoramas are displayed as shown below:

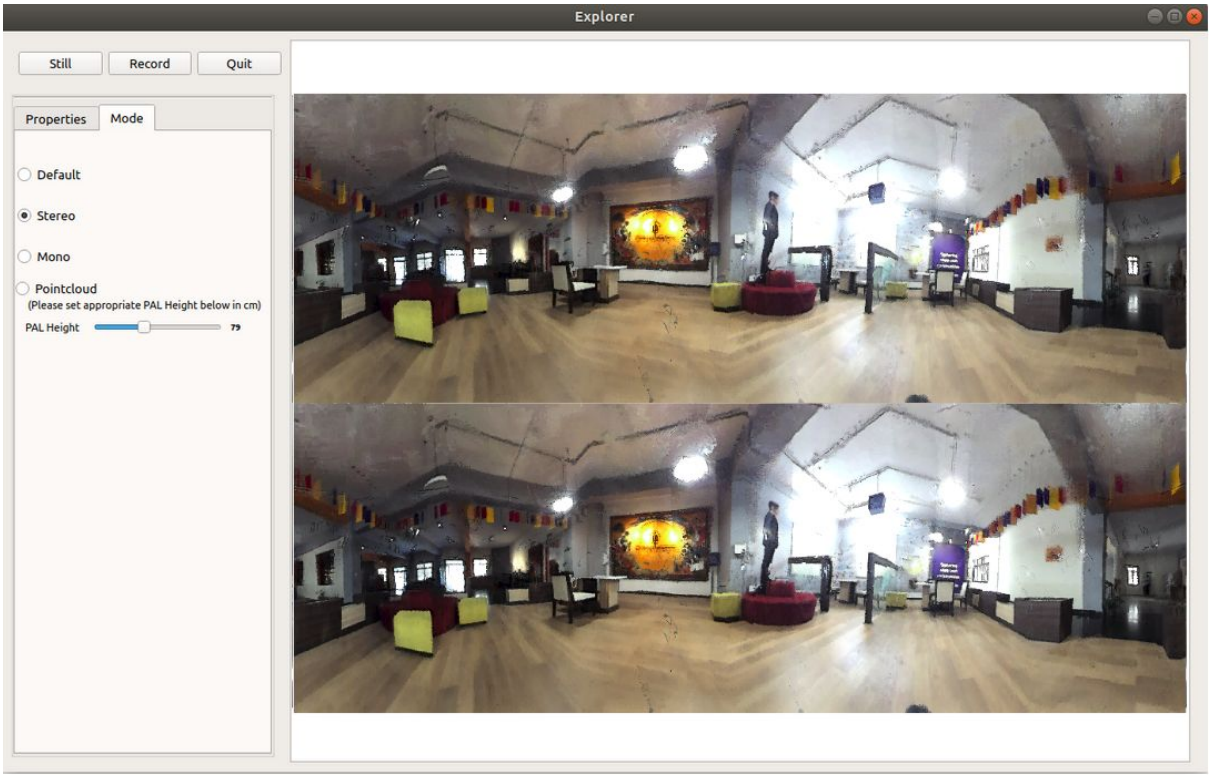


Figure 4.1.5: Stereo mode

In the Mono mode just the left panorama is displayed as shown below:

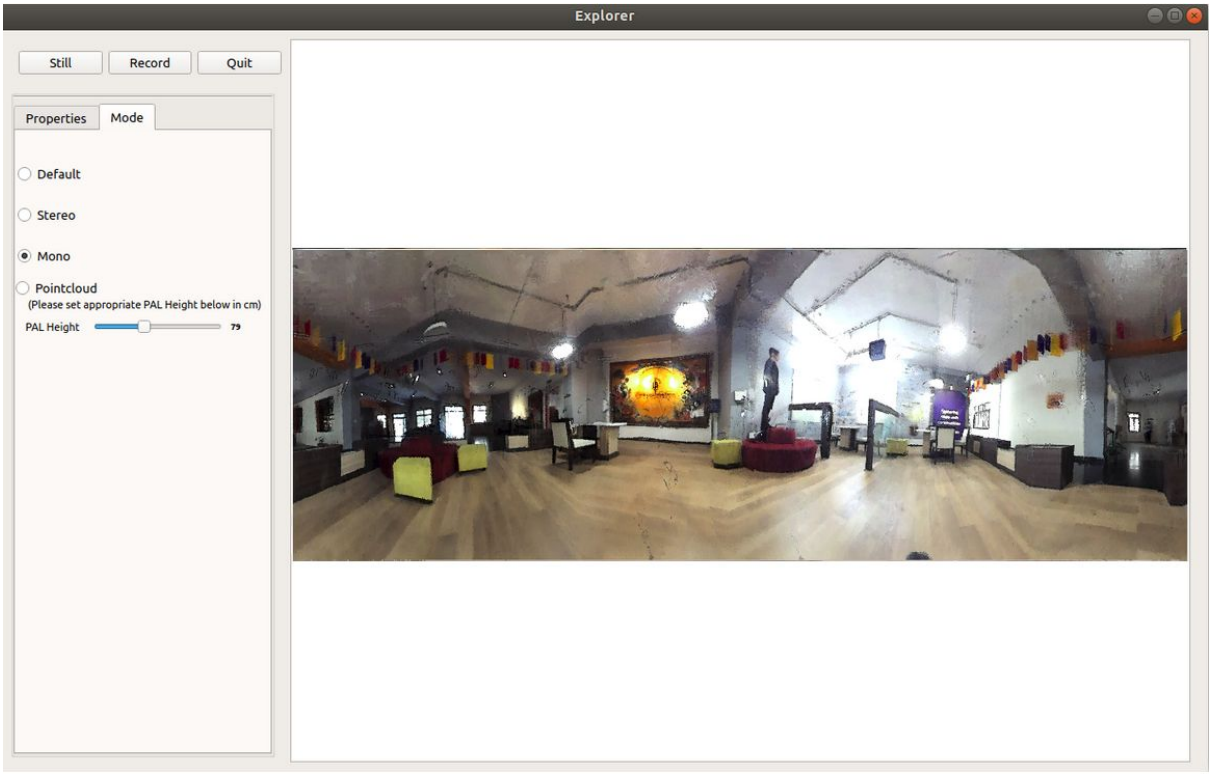


Figure 4.1.6: Mono mode

Finally, in Pointcloud mode, the pointcloud is drawn in 3D space.

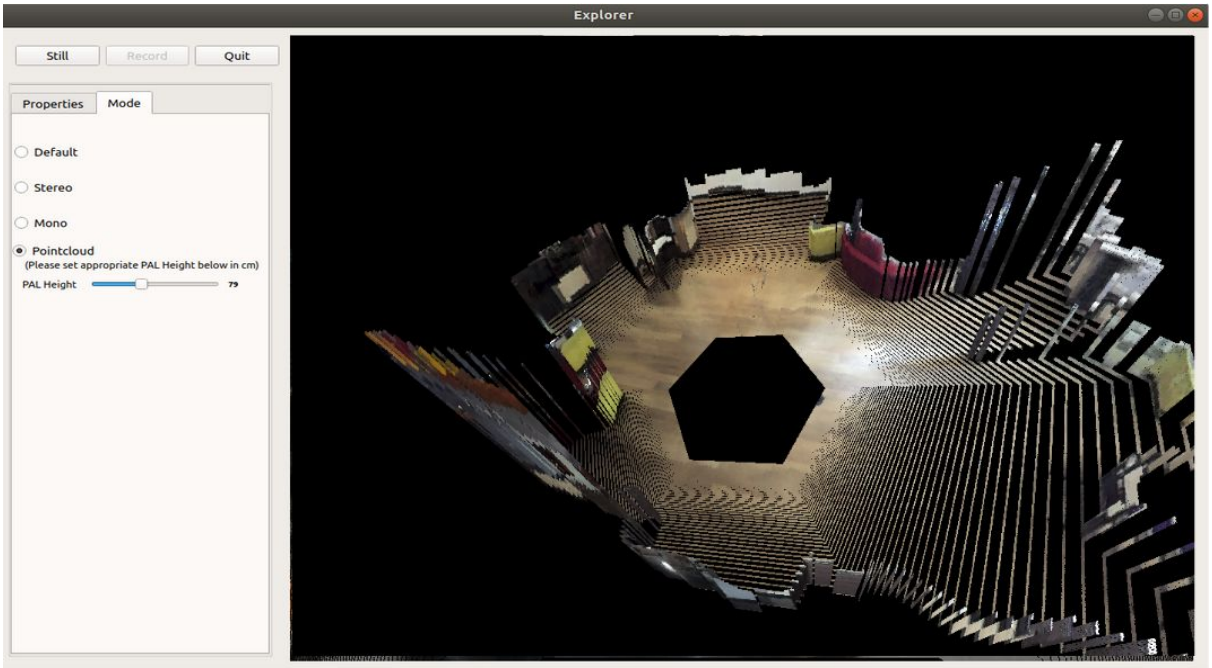


Figure 4.1.7: Pointcloud mode

4.1.2 Changing Properties

PAL properties can be configured in the properties pane. This pane is exposed by clicking on the properties tab highlighted in the image below:

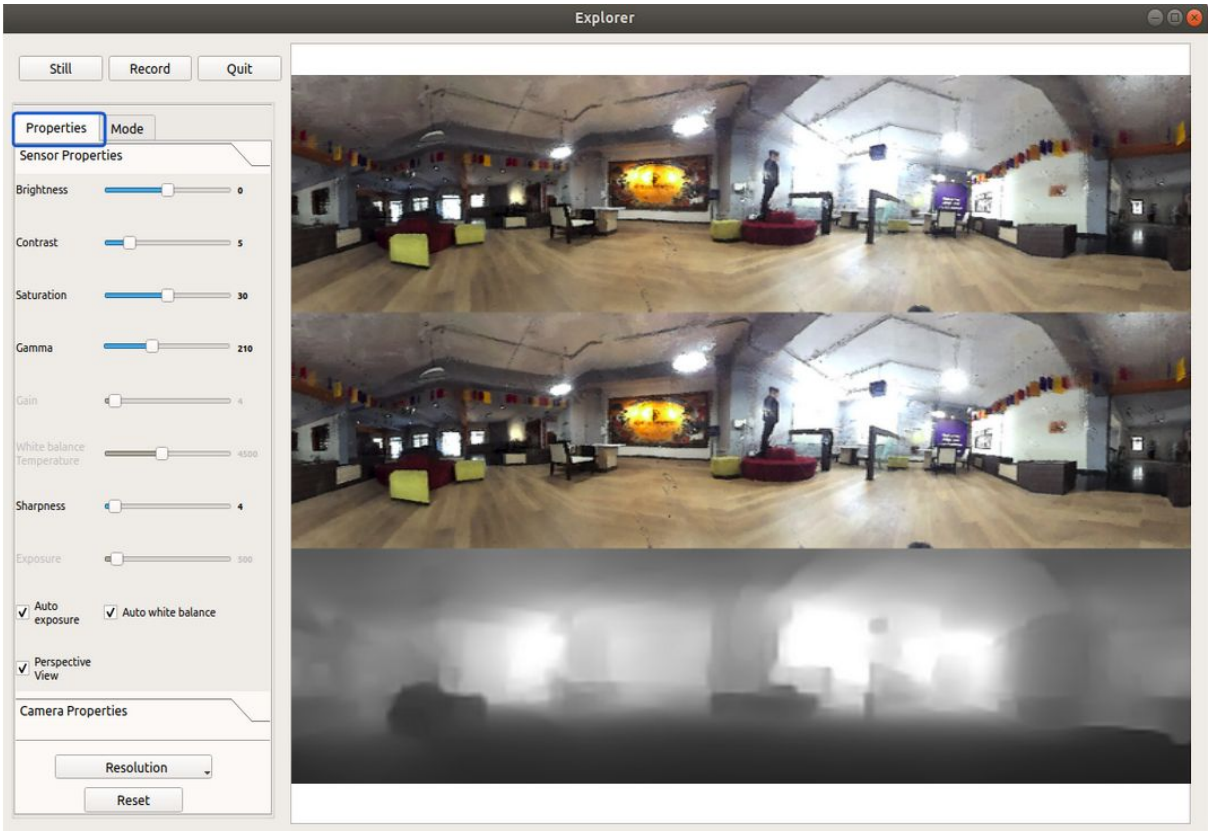


Figure 4.1.8: Clicking on Properties tab

The properties are divided into two sections:

- Sensor properties
- Camera properties

In the default view, the camera properties are hidden and they can be accessed by clicking the corresponding section header.

4.1.3.1 Sensor Properties

This grouping of properties is associated with the imaging sensor used in PAL. All the properties in this section have corresponding members in the API CameraProperties structure.

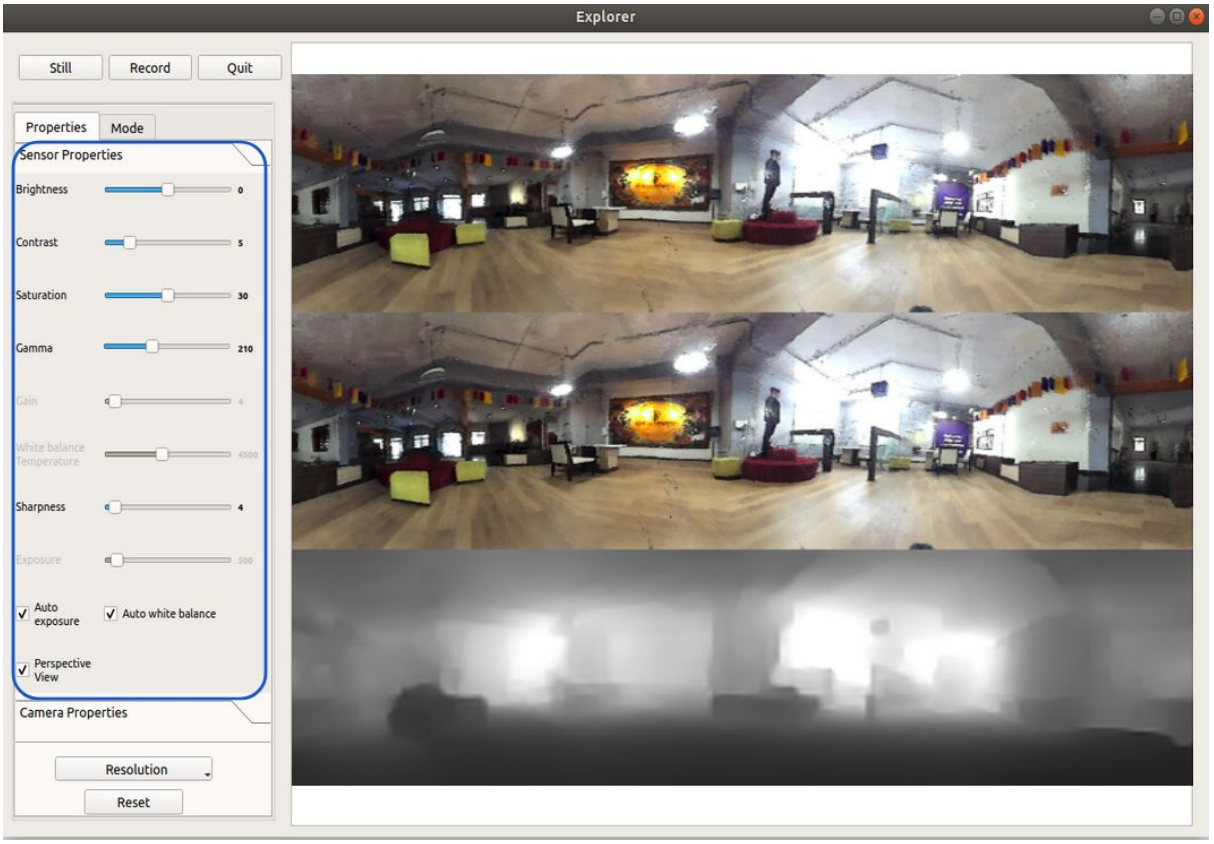


Figure 4.1.9: Sensor Properties

Brightness, Contrast, Saturation, Gamma, Gain, White balance temperature, Sharpness and Exposure can be modified by dragging the corresponding sliders.

Auto Exposure checkbox can be checked or unchecked to enable or disable

automatically adjusting the exposure of PAL respectively.

- *Note:* When Auto Exposure is checked the Gain and Exposure sliders are disabled.

Auto White Balance checkbox can be checked or unchecked to enable or disable automatically adjusting the white balance temperature of PAL respectively.

- *Note:* When Auto White Balance is checked White Balance Temperature slider is disabled.

Perspective View checkbox can be checked or unchecked to switch between perspective and equirectangular projection.

4.1.3.2 Camera Properties

This grouping of properties is associated with the PAL Ethernet Camera itself. Most of the properties in this section have corresponding members in the API CameraProperties structure. Exceptions are called out in the description.

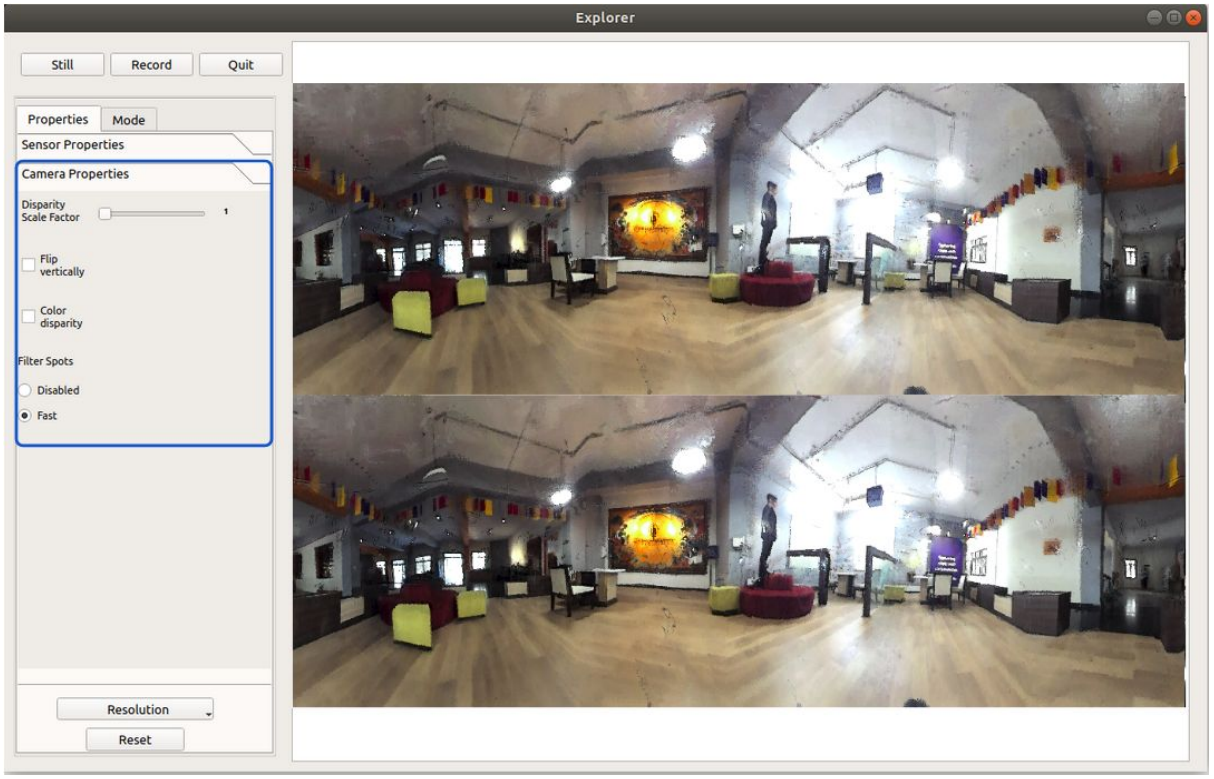


Figure 4.1.10: Camera Properties

Disparity Scale Factor is a property exclusive to the Explorer application to allow

users to scale and visualize the disparity computed by the API backend. It is a simple scalar multiplier applied to disparity values output by the API. It allows users to visually perceive disparity of the scene better over a larger depth range.

Flip Vertically checkbox can be checked / unchecked to enable / disable vertically flipping the output data frames respectively. This is particularly useful if the device is being used in a top-down orientation.

Filter Disparity checkbox can be checked or unchecked to view filtered disparity or raw disparity respectively.

Filter Spots checkbox can be checked or unchecked to enable / disable filtering of bright spots (Eg. specular reflections) respectively.

Color Disparity checkbox can be checked or unchecked to apply or not apply a color map to disparity respectively. This property is exclusive to the Explorer application.

Disparity Computation can be used to select one of the available modes of computing disparities and depth mode.

4.1.3.3 Ungrouped elements

This section describes ungrouped properties and UI elements in the properties pane.

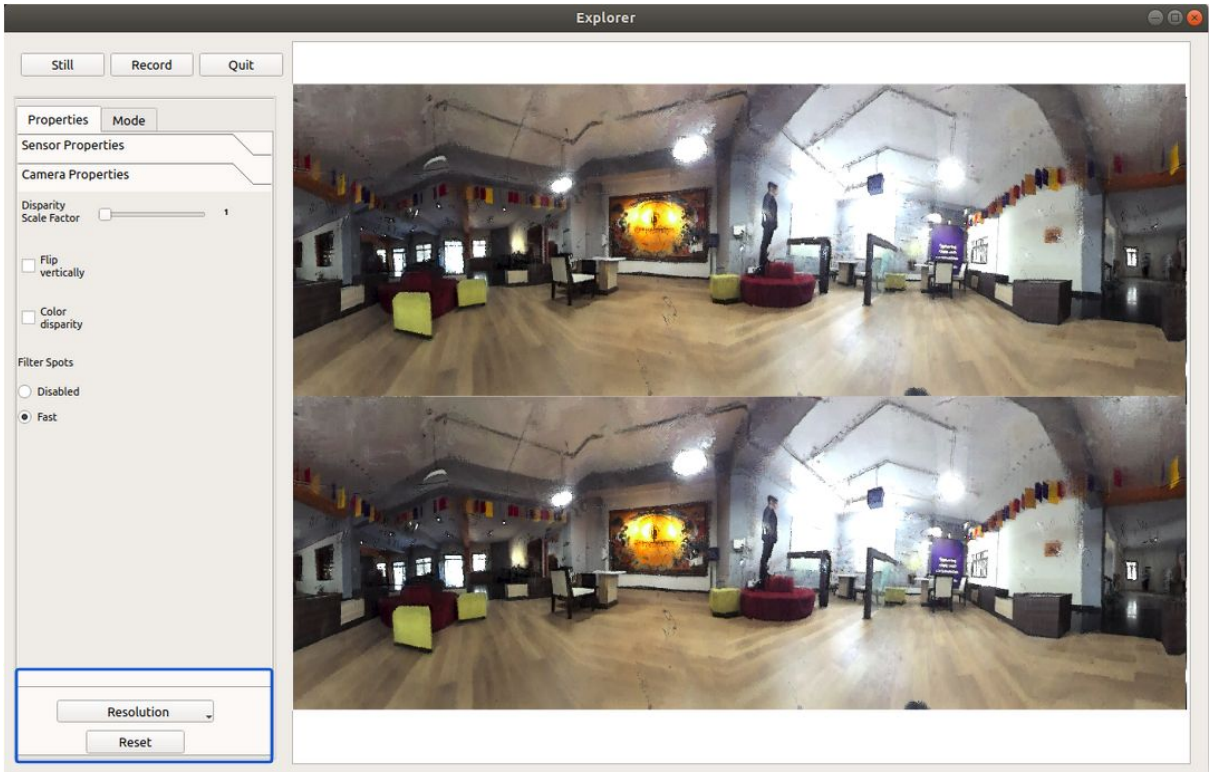


Figure 4.1.11: Ungrouped properties & UI elements

Resolution Dropdown can be used to switch resolutions dynamically.
Note: It is not possible to switch resolutions when video recording is in progress.

Reset Button:

The Reset button (at the bottom of the Properties pane) resets the camera properties to what they were at the beginning of the session i.e. it undoes the changes made in the current session.

Note: To reset camera properties to default values, close the Explorer application, delete the `SavedPalProperties.txt` file and restart the application.

4.1.4 Capturing Images

Capturing images is as simple as clicking the Still button highlighted in the image below:

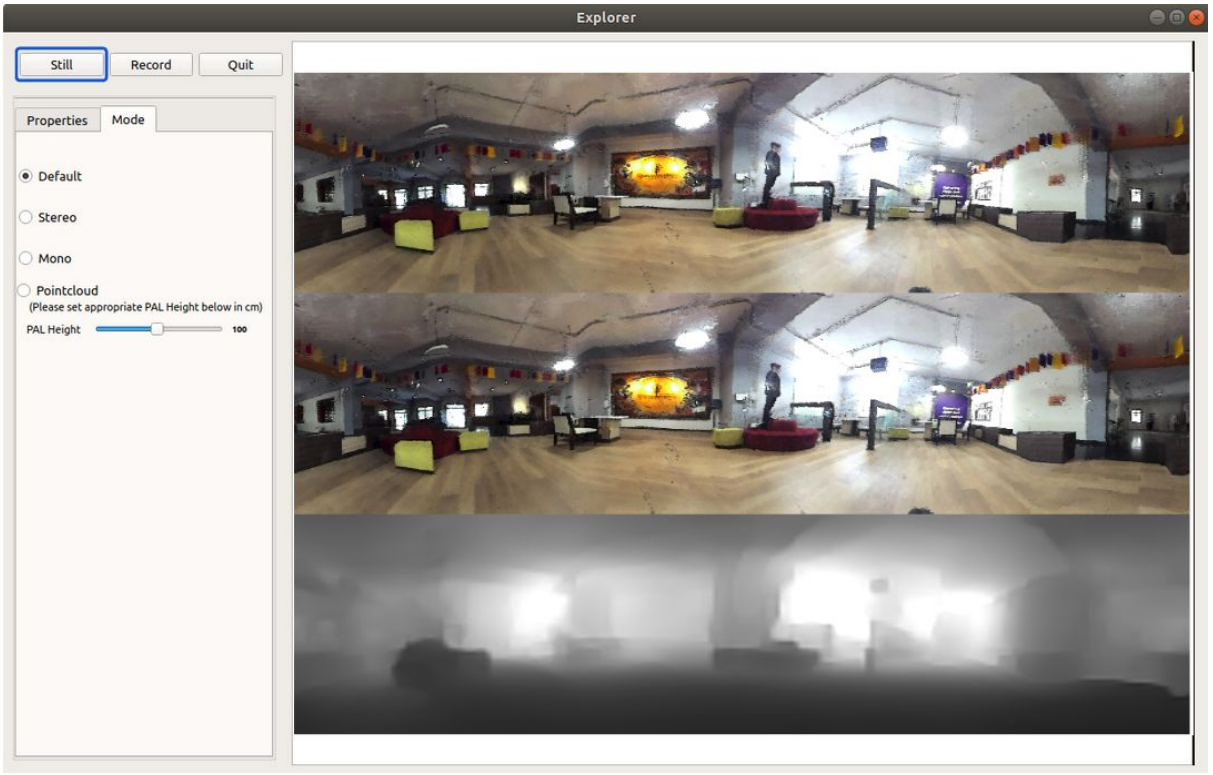


Figure 4.1.12: Still button highlighted

The captured image is saved in the `./Explorer/images/` folder. The image is saved uncompressed in a .tga format. The image captured corresponds to the mode selected i.e. whatever panoramas are displayed on the screen in the current mode are captured as a single image

4.1.5 Recording Video

To begin recording a video click the Record button highlighted in the image below:

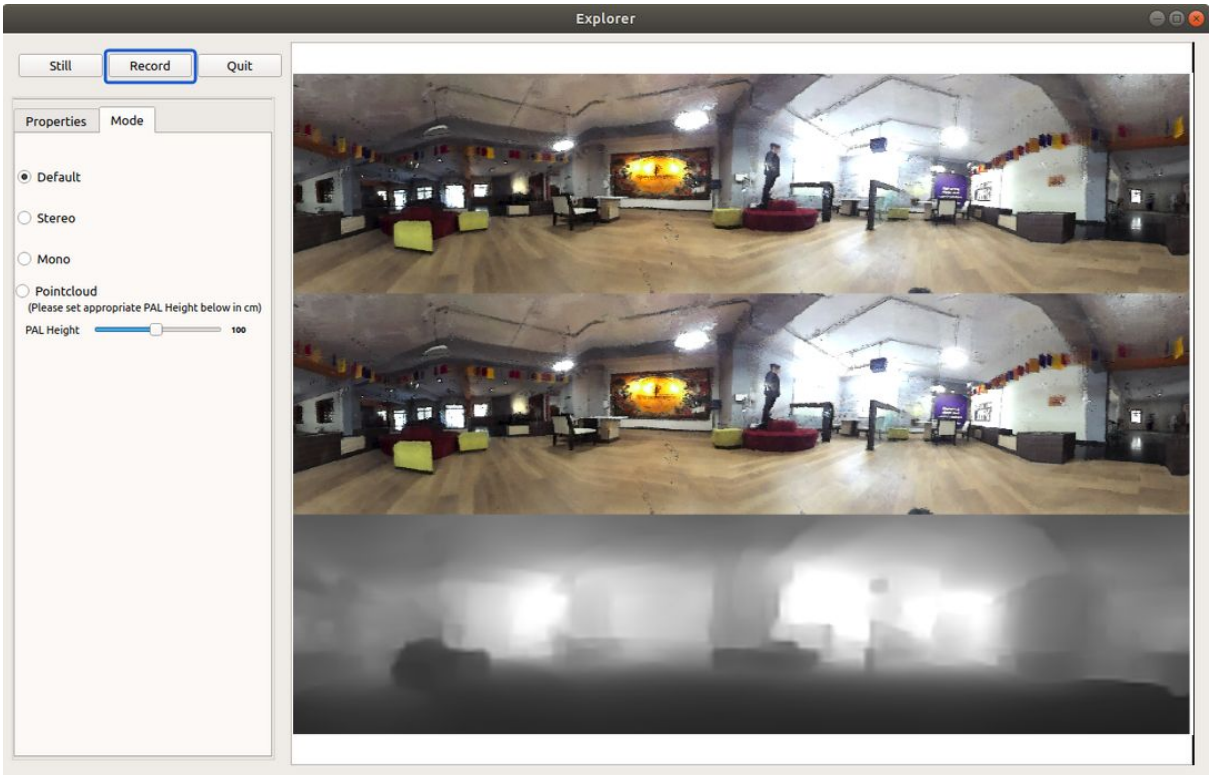


Figure 4.1.13: Record button highlighted

Once it is clicked, the Record button changes state to the Stop button highlighted in the figure below:

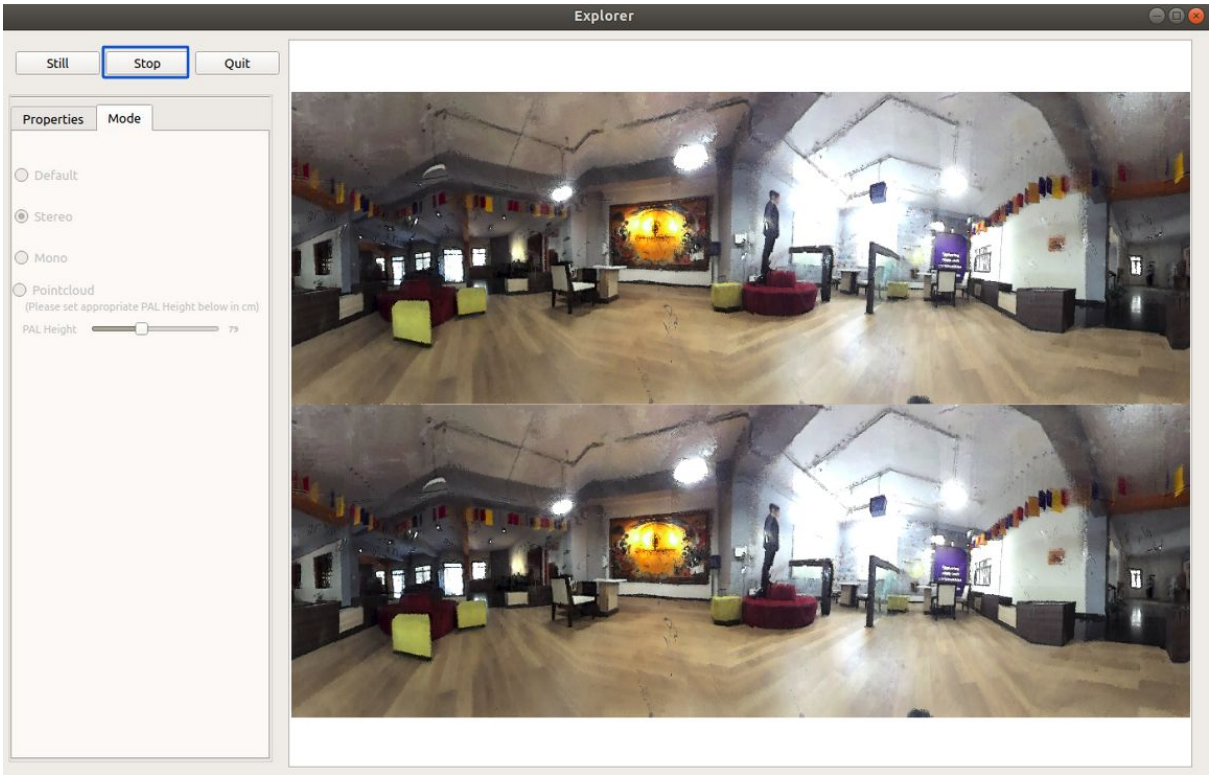


Figure 4.1.14: Stop button highlighted

You can click the Stop button to stop recording the video. It is saved to an automatically named file in the `./Explorer/videos/` folder. The video captured corresponds to the mode selected i.e. whatever panoramas are displayed on the screen in the current mode are captured in the video. The video is recorded compressed using MJPEG format in an .avi file container.
***Note**:* When recording a video, changing modes or resolution is not allowed.

4.1.6 Quit Application

You can quit the application by clicking the quit button highlighted in the following image:

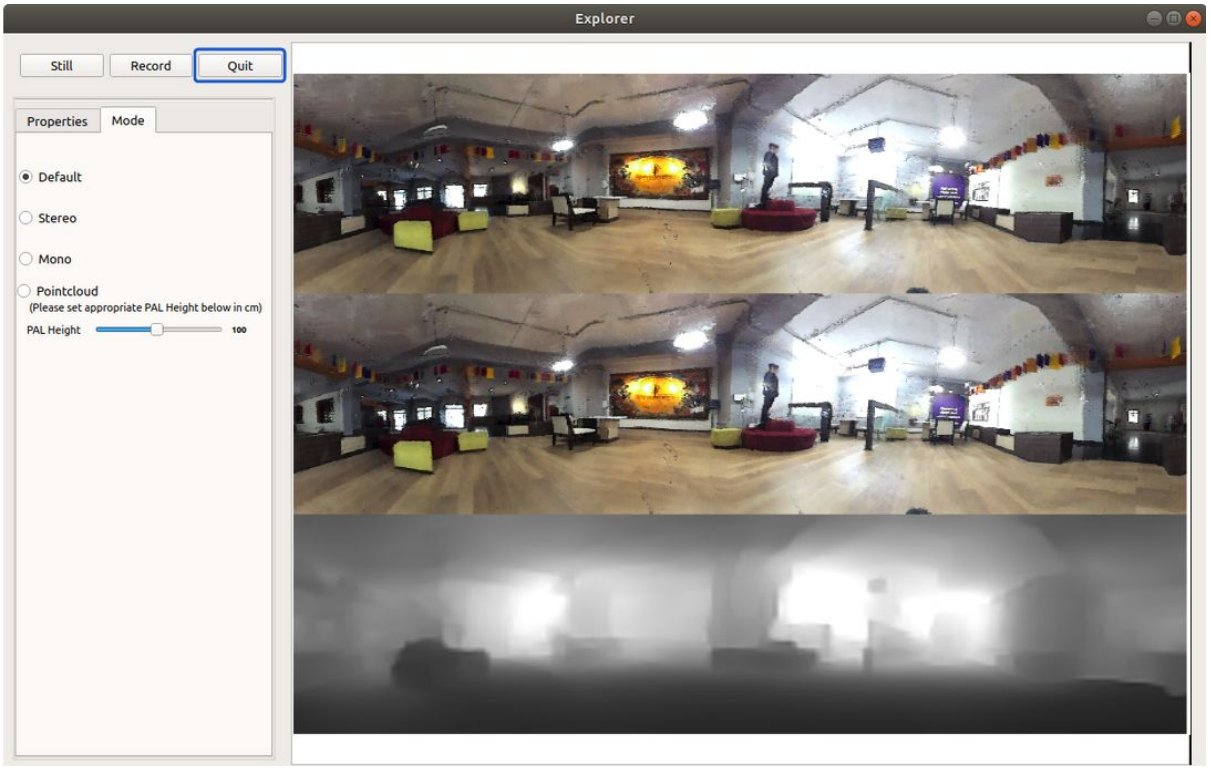


Figure 4.1.15: Quit button highlighted

Note:

- When the explorer is quit it saves the current properties configuration to a `SavedPalProperties.txt` file in the same directory as the application. This file can be used to initialize the ROS pal_camera_node node or as part of the LoadProperties call.
- When the Explorer is launched, it checks if the `SavedPalProperties.txt` file exists, if it does, it loads those parameters, if not, it loads the default properties as discussed in the API section.

4.2 Sample Applications

As part of the SDK users are provided with source code for sample applications. They are grouped into "tutorials" and "code samples". Tutorials are simple and complete examples, provided to quickly get started with the PAL API. They can be found in the `./tutorials/` folder. Code samples are slightly more complex and can be found in the `./code_samples/` folder.

4.2.1 Running Sample Applications

In this section we will show you how to compile and run a sample application - specifically, the `001_cv_png.cpp` tutorial.

First, let us change our working directory to the tutorials folder.

```
cd tutorials/
```

To compile all the tutorials you can run the compile script:

```
./compile.sh
```

To compile just the first tutorial, run the following command:

```
g++ 001_cv_png.cpp ../lib/libPAL.so `pkg-config --libs --cflags opencv`
-g -o 001_cv_png.out -I../include/ -lv4l2 -lpthread
```

Notice that as part of the compilation libPAL.so, OpenCV, pthread and V4l2 libraries are being linked. These are the only libraries the API backend depends upon.
Checkout the first tutorial by opening `001_cv_png.cpp`, read the code and inline comments to get started. You can run the corresponding executable as follows:

```
./001_cv_png.out
```

When the executable is successfully executed, it should create 2 different files: 'left.png' and 'right.png' in the same folder. The output looks like:

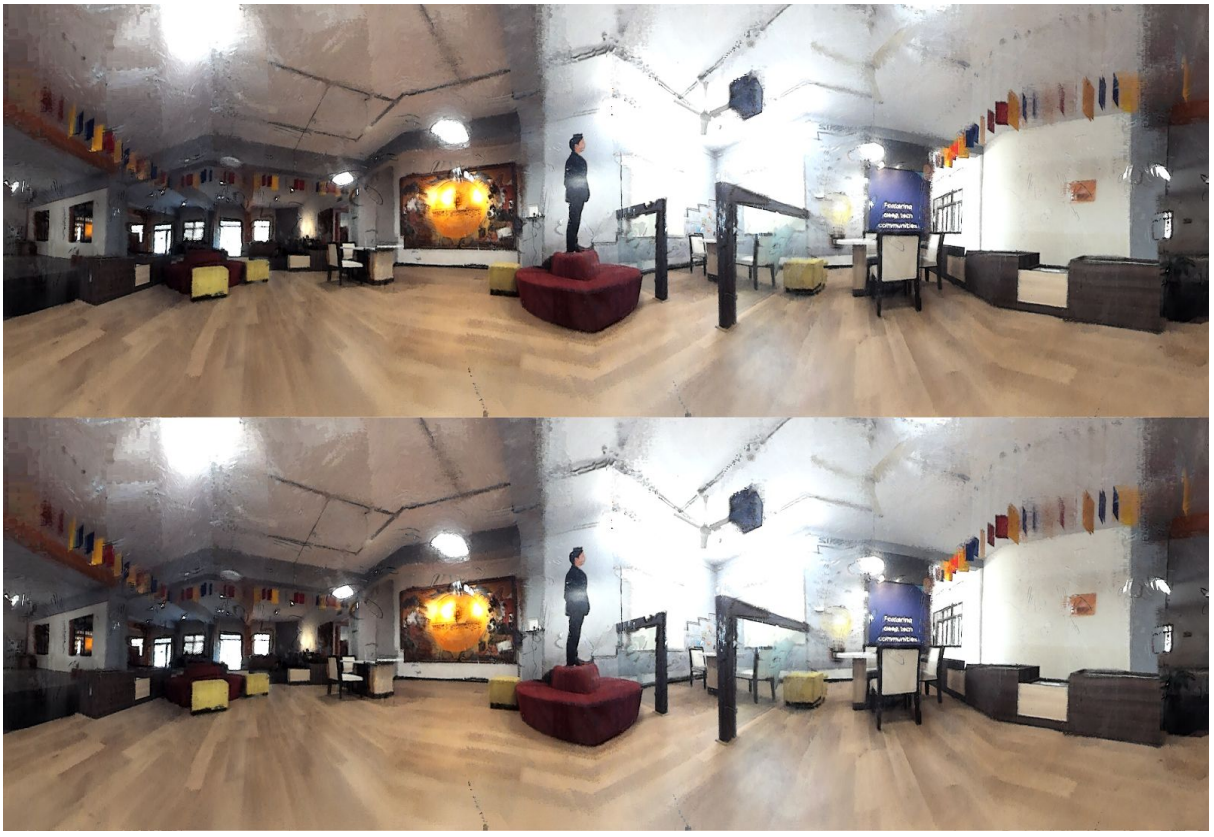


Figure 4.1: Left & right panoramas saved to the file when first tutorial is run

You can follow a similar process for the other sample applications to begin your understanding of the PAL API.

5. PAL API Reference

5.1 Terminology

Let us define some terminology to help you with understanding the rest of this section.

Callee	The function being invoked
Caller	The function invoking the Callee
Write Argument	The Callee writes data into this argument, the Caller reads it.

Read Argument	The Callee reads data from this argument, the Caller provides it.
Optional Argument	This argument can be optionally not provided. When not provided it will take on a default value.

5.2 Initialization and Destruction

There are two important calls related to managing the resources associated with the camera: `Init` and `Destroy`.

5.2.1 Init Call

```
PAL::Acknowledgement Init(int& panoramaWidth, int& panoramaHeight);
```

Overview:

Initializes the PAL API backend by allocating the necessary resources and interfacing with the camera hardware. This function should be called before any other API function call.

Note: As part of the initialization CameraProperties are set to default values. You can change these camera parameters with the help of APIs discussed in section 5.4.8 and 5.4.12.

Arguments:

Name	`panoramaWidth`
Type	Write
Optional	No
Description	Width of the panoramas returned in *GrabFrames Call.*

Name	`panoramaHeight`

Type	Write
Optional	No
Description	Height of the panoramas returned in *GrabFrames*.

Return

PAL::SUCCESS	If initialization is successful.
PAL::FAILURE	If initialization failed. A common cause for this is that the camera is not connected.
PAL::IGNORED	If Init call is made multiple times without a Destroy call.

5.2.2 Destroy Call

```
PAL::Destroy();
```

Overview:

Releases all the resources that were initialized as part of program execution in the API backend.

It is not required to call this function usually. The API implementation takes care of calling this function automatically at the time of application exit. However, if the user needs to call `PAL::Init` multiple number of times in the same application, then `PAL::Destroy` should be called before the next `PAL::Init` is called.

Arguments: None
Return: Void

5.3 Capture Image

This section talks about the APIs associated with image capture from the camera. First let us take a look at the `Image` structure associated with these calls.

5.3.1 Image Structure

```
struct Image
{
	union RawData
	{
		void* data;
		unsigned char* u8_data;
		unsigned short* u16_data;
		float* f32_data;
	}Raw;

	int rows;
	int cols;
	int channels;
	int bytesPerChannel;
	int stride;
	int size;
	timeval timestamp;
};
```

This structure encapsulates an image and the associated metadata. The members of the structure are explained below:

Raw	Allows users to access the start of the raw pixel memory and interprets it as different data types depending on the type of image stored
rows	Height of the image
cols	Width of the image
channels	Number of channels (for eg., 3 for RGB, 1 for depth and disparity)
bytesPerChannel	Number of bytes required to represent one channel of a single pixel (for e.g., 1 for RGB, 4 for depth)
stride	Number of bytes occupied by a single row of pixels

size	Overall size of the image in bytes
timestamp	[NOT SUPPORTED IN THIS VERSION. RESERVED FOR FUTURE] Represents the time at which the data was captured. *Note 1*: This time instant corresponds to the moment when the first byte of data is transferred from PAL to the host system. It is represented as epoch time. *Note 2*: The time difference between the real world event and the time returned in this structure will vary based on lighting conditions, camera properties, etc. which affects the exposure time. *Note 3*: For information about timeval structure please this link.

5.3.1.1 SetDimensions Method

```
void SetDimensions(int width, int height, int channelCount,
int bytesPerChannel)
```

This method allows us to set the dimensions of the image like width, height, number of channels and the number of bytes each channel takes per pixel. It also computes and sets the `stride` and `size` members of the `Image` structure from the provided arguments.

5.3.1.2 Set Method

```
void Set(void* ptr, int width, int height, int channelCount = 3,
int bytesPerChannel = 1)
```

This method is same as the `SetDimensions` method except that additionally the memory location of the raw pixel data can be set with the ptr argument.

5.3.1.3 Create Method

```
void Create(int width, int height, int channelCount = 3,
int bytesPerChannel = 1)
```

This method is same as the `Set` method except that it additionally allocates new memory corresponding to the image dimensions provided as arguments.

5.3.1.4 Destroy Method

```
void Destroy();
```

Releases the memory associated with the image structure.

5.3.1.5 Converting PAL::Image to cv::Mat

```
int width = 256;
int height = 256;
int channels = 3;
int bytesPerChannel = 1;

PAL::Image img;
img.Create(width, height, channels, bytesPerChannel);
cv::Mat m = cv::Mat(rgb.rows, rgb.cols, CV_8UC3, rgb.Raw.data);
```

Note: Please make sure that the channel depth, number of channels of the input image and the selected OpenCV image format match.

5.3.1.6 Converting from cv::Mat to PAL::Image

```
cv::Mat m = cv::Mat::zeros(256, 256, CV_16SC1);
PAL::Image img;
int channels = 1;
int bytesPerChannel = 2;
img.Set(m.data, m.cols, m.rows, channels, bytesPerChannel);
```

The set function of the `PAL::Image` structure can be used for the conversion as shown above.

Note: Please make sure that the channel depth, number of channels of the input

image and the selected OpenCV image format match.

5.3.2 GrabFrames Call

```
PAL::Acknowledgement GrabFrames(PAL::Image* left,
                                PAL::Image* right,
                                PAL::Image* depth = 0,
                                PAL::Image *disparity = 0,
                                bool normalize = false,
                                bool asynchronous = true);
```

Overview:

This function helps us retrieve the latest available left, right, depth and disparity panoramas.

Arguments:

Name	`left`
Type	Write
Optional	No (But, it can be NULL)
Description	Panorama image as seen by the left view in a stereo pair. This will be a 8 bit, 3 channel image. Equivalent to CV_8UC3 of OpenCV.

Name	`right`
Type	Write
Optional	No (But, it can be NULL)
Description	Panorama image as seen by the right view in a stereo pair. This will be a 8 bit, 3 channel image. Equivalent to CV_8UC3 of OpenCV.

Name	`depth`
Type	Write
Optional	Yes, the default value is 0
Description	Panorama image representing the depth perceived by the stereo system. This image will be 32 bit (float), 1 channel. Equivalent to CV_32FC1 of OpenCV. The unit of measurement is centimeter.

Name	`disparity`
Type	Write
Optional	Yes, the default value is 0
Description	Panorama image representing the disparity perceived by the stereo system. This image will be: 8 bit, 1 channel when normalize argument is set to true. Equivalent to CV_8UC1 of OpenCV. 16 bit, 1 channel when normalize argument is set to false. Equivalent to CV_16SC1 of OpenCV. *Note:* When normalization is enabled, internally the 16 bit unnormalized information is mapped to 8 bits, such that 0 and 255 are represented by the minimum and maximum values in the current depth image respectively. As a consequence of this, it should be noted that disparity values for objects at same depth across multiple GrabFrames calls may not be the same (when normalization is enabled).

Name	`normalize`
Type	Read
Optional	Yes, the default value is false
Description	This argument enables / disables disparity normalization when it is set to true / false respectively.

Name	`asynchronous`
Type	Read
Optional	Yes, the default value is true
Description	[NOT SUPPORTED IN THIS VERSION. RESERVED FOR FUTURE] When this argument is set to true, the call immediately returns and the last successfully computed image information is returned i.e all the images returned will not be in sync (from a single instant of time) because of differing computation speeds of individual images. When this argument is set to false, the call blocks till the current frame is completely processed and then returns the image information i.e. all the images returned are in sync (from a single instant of time).

Note: The memory location corresponding to raw pixel data written to the `left`, `right`, `depth` & `disparity` `PAL::Image` structures will contain the address to the same memory locations across multiple `GrabFrames` calls. If you wish to operate on data from more than one frame simultaneously please copy the data as required.

Return:

PAL::SUCCESS	If the call succeeds.
PAL::FAILURE	If the call fails (eg. when the Ethernet is loose/removed).
PAL::ERROR_CAMERA_NOT_INITIALIZED	If this call is made before an Init call.
PAL::IGNORED	This can happen in the following cases: 1. If all the 4 Image arguments are null.

Note: The left and right panoramas returned by this call are always the latest, while the disparity and depth panorama are not. This is because they are computed at a different speed. The last computed depth / disparity panorama is returned at the time a `GrabFrames` call is made.

5.4 Camera Properties

This section details API calls related to setting and getting different camera properties. First let us take a look at some relevant structures and enums.

5.4.1 Resolution Structure

```
struct Resolution
{
	int width;
	int height;
};
```

This structure encapsulates the width and height of the panorama image.

5.4.2 Projection Enum

```
enum Projection
{
	EQUI_RECTANGULAR = 0,
	PERSPECTIVE = 1,
```

```
};
```

This enum can be used to indicate if the PAL Ethernet camera should interpret the captured panorama as an image captured using perspective projection - or using equi-rectangular projection. When the projection is set to equi-rectangular mode, the panoramas appear as if it is wrapped around a sphere. When the perspective projection is used, the panoramas appear as if they are formed by combining multiple perspective view frustums.

5.4.3 DisparityComputation Enum

```
enum DisparityComputation
{
      FAST = 0,
};
```

This enum can be used to indicate the computation mode used for disparity and depth calculation.

5.4.4 CameraProperties Structure

```
struct CameraProperties
{
      int brightness;
      int contrast;
      int saturation;
      int gamma;
      int gain;
      int white_bal_temp;
      int sharpness;
      int exposure;
      bool  auto_white_bal;
      bool  auto_exposure;

      Resolution resolution;

      bool  vertical_flip;
      bool  filter_disparity;
      bool  filter_spots;
```

```
        Projection projection;
        DispartyComputation computation;

        int camera_height;
};
```

This structure encapsulates the various properties of PAL. The following sections go into more detail about each of the camera properties.

Note: Please refer to the table below for the range of possible values of the camera properties.

Property	Minimum Value	Maximum Value	Valid Step Size	Default Value
Brightness	-15	15	1	0
Contrast	0	30	1	4
Saturation	0	60	1	30
Gamma	40	500	1	220
Gain	1	100	1	4
White balance temperature	10	10000	10	4500
Sharpness	0	127	1	0
Exposure	1	10000	1	290

5.4.4.1 Brightness

Represents the brightness of the left and right panoramas. The minimum, maximum and default values for this property can be accessed as follows:

```
PAL::CameraProperties::MIN_BRIGHTNESS
PAL::CameraProperties::MAX_BRIGHTNESS
PAL::CameraProperties::DEFAULT_BRIGHTNESS
```

Valid values for this property should lie between these limits.

5.4.4.2 Contrast

Represents the contrast of the left and right panoramas. The minimum, maximum and default values for this property can be accessed as follows:

```
PAL::CameraProperties::MIN_CONTRAST
PAL::CameraProperties::MAX_CONTRAST
PAL::CameraProperties::DEFAULT_CONTRAST
```

Valid values for this property should lie between these limits.

5.4.4.3 Saturation

Represents the colour saturation of the left and right panoramas. The minimum, maximum and default values for this property can be accessed as follows:

```
PAL::CameraProperties::MIN_SATURATION
PAL::CameraProperties::MAX_SATURATION
PAL::CameraProperties::DEFAULT_SATURATION
```

Valid values for this property should lie between these limits.

5.4.4.4 Gamma

The minimum, maximum and default values for this property can be accessed as follows. Valid values for this property should lie between these limits.

```
PAL::CameraProperties::MIN_GAMMA
PAL::CameraProperties::MAX_GAMMA
PAL::CameraProperties::DEFAULT_GAMMA
```

5.4.4.5 Gain

The minimum, maximum and default values for this property can be accessed as follows and valid values for this property should lie between these limits.

```
PAL::CameraProperties::MIN_GAIN
PAL::CameraProperties::MAX_GAIN
PAL::CameraProperties::DEFAULT_GAIN
```

Note: When `auto_exposure` is enabled, `gain` is ignored. Also, when `auto_exposure` is disabled the `gain` is reset to the last successfully set value.

5.4.4.6 White Balance Temperature (white_bal_temp)

The minimum, maximum and default values for this property can be accessed as follows and valid values for this property should lie between these limits.

```
PAL::CameraProperties::MIN_WHITE_BAL_TEMP
PAL::CameraProperties::MAX_WHITE_BAL_TEMP
PAL::CameraProperties::DEFAULT_WHITE_BAL_TEMP
```

Note: When `auto_white_bal` is enabled, `white_bal_temp` is ignored. Also, when `auto_white_bal` is disabled the `white_bal_temp` is reset to the last successfully set value.

5.4.4.7 Sharpness

The minimum, maximum and default values for this property can be accessed as follows and valid values for this property should lie between these limits.

```
PAL::CameraProperties::MIN_SHARPNESS
PAL::CameraProperties::MAX_SHARPNESS
PAL::CameraProperties::DEFAULT_SHARPNESS
```

5.4.4.8 Exposure

The minimum, maximum and default values for this property can be accessed as follows and valid values for this property should lie between these limits.

```
PAL::CameraProperties::MAX_EXPOSURE
PAL::CameraProperties::MIN_EXPOSURE
PAL::CameraProperties::DEFAULT_EXPOSURE
```

Note 1: When `auto_exposure` is enabled, `exposure` is ignored. Also, when `auto_exposure` is disabled the `exposure` is reset to the last successfully set value.

Note 2: Exposure units are 100 microseconds i.e. a setting of 200 corresponds to 200 * 100 microseconds which is 20 milliseconds.

5.4.4.9 Auto White Balance (auto_white_bal)

When this property is set to `true`, the White Balance Temperature of the camera is automatically adjusted. The default value can be accessed as follows:

```
PAL::CameraProperties::DEFAULT_AUTO_WHITE_BAL
```

Note: When enabled, the value set for `white_bal_temp` is ignored (while setting properties). When a `GetCameraProperties` call is made and `auto_white_bal` is enabled the last successfully set value is returned for `white_bal_temp`.

5.4.4.10 Auto Exposure (auto_exposure)

When the value of this property is set to `false` the Exposure of the left and right panoramas are automatically adjusted. The default value can be accessed as follows:

```
PAL::CameraProperties::DEFAULT_AUTO_EXPOSURE
```

Note: When enabled, the value set for `exposure` is ignored (while setting properties). When a `GetCameraProperties` call is made and `auto_exposure` is enabled the last successfully set value is returned for `exposure`.

5.4.4.11 Resolution

The resolution of the panoramas returned in `GrabFrames`. This property affects the processing that follows like disparity, depth & point-cloud. This property is reserved for future versions. Any attempt to change the resolution will be ignored.
The default value of resolution can be accessed as follows:

```
PAL::CameraProperties::DEFAULT_RESOLUTION
```

5.4.4.12 Vertical Flip (vertical_flip)

This property will flip the output of the camera vertically if set to `true`. The default value can be accessed as follows:

```
PAL::CameraProperties::DEFAULT_VERTICAL_FLIP
```

5.4.4.13 Filter Disparity (filter_disparity)

This property will enable / disable filtering of raw disparity when set to `true` or `false` respectively. Filtering is done to produce a smoother disparity output. The default property value can be accessed as follows:

```
PAL::CameraProperties::DEFAULT_FILTER_DISPARITY
```

5.4.4.14 Filter Spots (filter_spots)

This property will enable / disable the reduction in spurious bright-spots in the

output RGB images when set to `true` / `false` respectively. The default property value can be accessed as follows:

```
PAL::CameraProperties::DEFAULT_FILTER_SPOTS
```

Setting this property to `true` will have the following effects:

1. Post-processing is performed on the regions identified with spurious bright spots (eg: specular reflections) to minimize them. This step is performed for each output image frame as long as `filter_spots` setting is set to true.

5.4.4.15 Projection

This property allows users to change the projection used when panoramas are accessed from PAL. It can be set to any of the Projections. The default property value can be accessed as follows:

```
PAL::CameraProperties::DEFAULT_PROJECTION
```

5.4.4.16 DispartyComputation

This property allows users to change the computation mode used when disparity and depth panoramas are computed. It can be set to any of the Computations. The default property value can be accessed as follows:

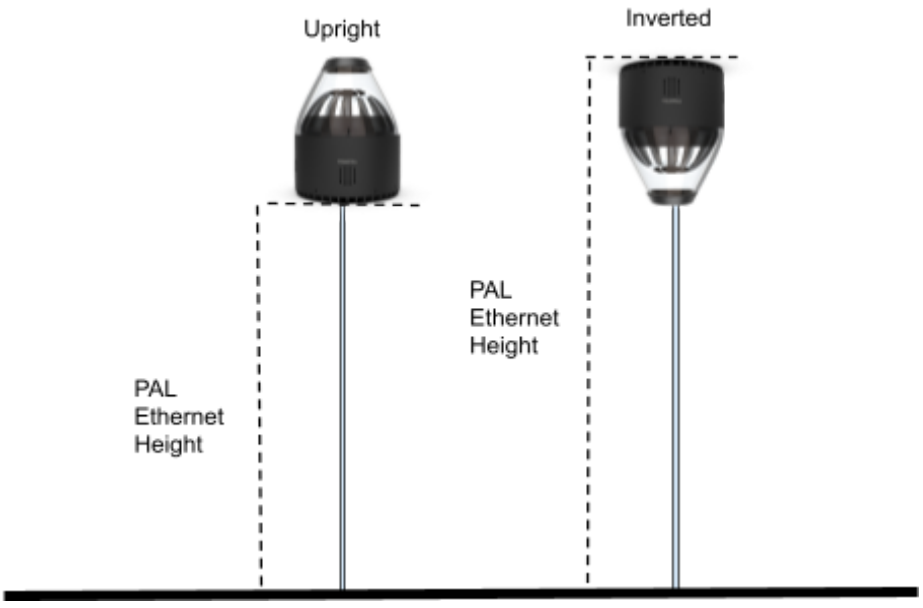
```
PAL::CameraProperties::DEFAULT_COMPUTATION
```

5.4.4.17 Camera Height (camera_height)

This property allows users to change the height of the camera from the floor. Height can be measured from the base of the PAL Ethernet camera to the floor in **centimeters**. The default property value can be accessed as follows:

```
PAL::CameraProperties::DEFAULT_CAMERA_HEIGHT
```

The height of the PAL Ethernet camera can be measured in centimeters as shown below:



5.4.7 CameraPropertyFlags Enum

This enum is used for selecting some or all the camera properties using bit flags. This is used in the `SetCameraProperties` call to specify which properties need to be set.

```
enum CameraPropertyFlags
{
	BRIGHTNESS = 0x1,
	CONTRAST = 0x2,
	SATURATION = 0x4,
	GAMMA = 0x8,
	GAIN = 0x10,
	WHITE_BAL_TEMP = 0x20,
	SHARPNESS = 0x40,
	EXPOSURE = 0x80,
	AUTO_WHITE_BAL = 0x100,
	AUTO_EXPOSURE = 0x200,
	RESOLUTION = 0x400,
	VERTICAL_FLIP = 0800,
	FILTER_DISPARITY = 0x1000,
	FILTER_SPOTS = 0x2000,
	PROJECTION = 0x4000,
	DISPARITY_COMPUTATION = 0x8000,
	CAMERA_HEIGHT = 0x10000,
	ALL = 0x1FFFF,
};
```

Note:

1. If you want to learn more about bit flags see this thread.
2. If you want to see how the enum is used in the API see SetCameraProperties section.

5.4.8 SetCameraProperties Call.

```
PAL::Acknowledgement
SetCameraProperties(PAL::CameraProperties* properties,
unsigned int* flags = 0);
```

Overview:

Allows the user to set different camera properties.

Arguments:

Name	properties
Type	Read
Optional	No
Description	Contains the values of the properties to be set.

Name	flags
Type	Read & Write
Optional	Yes, the default value is 0. When this argument is not specified, all the properties are set.

Description	This argument is read by the function to determine the selected properties that are to be set to the provided values in `properties` argument. If any of the provided values for the selected properties are invalid, then this argument is modified by the function to indicate which properties are invalid. This argument is to be used like a bit mask created using the CameraPropertyFlags enum. Read the following subsections for examples.

Return:

`PAL::SUCCESS`	If the call succeeds.
`PAL::FAILURE`	If the call fails.
`PAL::INVALID_PROPERTY_VALUED`	If any of the camera properties are invalid this is returned and the flags are updated to indicate the invalid properties. The current camera Properties remain unchanged in this case.
`PAL::ERROR_CAMERA_NOT_INITIALIZED`	If this call is made before an Init call.

Note: The API backend computes the left and right panoramas first, then the depth panorama and lastly the point cloud. Each stage feeds into the next. Therefore, the values you set for the camera properties here will affect the entire pipeline.

5.4.8.1 Setting the flags argument

Selecting all the properties:

```
unsigned int flags = PAL::ALL
```

For selecting a single property, simply set it to its corresponding

`CameraPropertyFlags` enum value. For example, to select the `saturation` property:

```
unsigned int flags = PAL::SATURATION
```

For selecting multiple properties, simply bitwise OR the corresponding `CameraPropertyFlags` enum values. To select `saturation` and `gain`:

```
unsigned int flags = PAL::SATURATION | PAL::GAIN
```

5.4.8.2 Reading the flags argument

To check if a particular property is invalid you just have to bitwise AND with the flags argument after the call returns a `PAL::INVALID_PROPERTY_VALUE` response. For example, to check if `saturation` is invalid:

```
bool isSaturationInvalid = flags & PAL::SATURATION
```

5.4.8.3 SetCameraProperties Call Examples

An example for setting all camera properties at once

```
PAL::CameraProperties p;
p.brightness = -10;
p.contrast = 10;
p.saturation = 10;
p.gamma = 50;
p.gain = 50;
p.white_bal_temp = 1500;
p.sharpness = 100;
p.exposure = 100;
p.auto_white_bal = true;
p.auto_exposure = true;
p.resolution = PAL::CameraProperties::DEFAULT_RESOLUTION;
p.vertical_flip = true;
p.filter_disparity = false;
p.filter_spots = true;
p.projection = PAL::PERSPECTIVE
PAL::SetCameraProperties(&p);
```

An example for setting only saturation and gamma:

```
PAL::CameraProperties properties;
```

```
int flags = PAL::SATURATION | PAL::GAMMA;
properties.saturation = 10;
properties.gamma = 50;
PAL::SetCameraProperties(&properties, &flags);
```

An example for reading invalid flags:

```
PAL::CameraProperties properties;
properties.saturation = 2;
properties.gain = 12345;
properties.white_bal_temp = 500;

int flags = PAL::GAIN | PAL::SATURATION | PAL::WHITE_BAL_TEMP;
PAL::SetCameraProperties(&properties, &flags);
```

In the code above `gain` and `white_bal_temp` are being set to invalid values.

Therefore, after the `SetCameraProperties` call the value of the `flags` argument will be 48. Which is a combination of the bit flags `PAL::WHITE_BAL_TEMP` (32) and `PAL::GAIN`(16). This means that the following statements are also correct:

- `flags` equals `(PAL::WHITE_BAL_TEMP | PAL::GAIN)`
- `(flags & PAL::WHITE_BAL_TEMP)` will be true
- `(flags & PAL::GAIN)` will be true
- Bitwise AND operation on `flags` and any other enum of `CameraPropertyFlags` will be false.

5.4.9 SetDefaultCameraProperties Call

```
PAL::Acknowledgement
SetDefaultCameraProperties(PAL::CameraProperties* properties = 0);
```

Overview:

Resets the camera properties to default values. Optionally, the users can provide a pointer to a `CameraProperties` structure as argument to retrieve the default camera property values.

Arguments:

Name	`Properties`
Type	Read
Optional	Yes, default value is NULL
Description	When the value is not NULL, the default property values are written to the CameraProperties structure pointed to by this argument.

Return:

`PAL::SUCCESS`	If the call succeeds.
`PAL::FAILURE`	Mostly if the cable is loose/removed.
`PAL::ERROR_CAMERA_NOT_INITIALIZED`	If this call is made before an Init call.

5.4.10 GetCameraProperties Call

```
PAL::Acknowledgement
GetCameraProperties(PAL::CameraProperties* properties);
```

Overview:

Allows users to retrieve the current camera properties.

Arguments:

Name	`Properties`
Type	Write
Optional	No
Description	The retrieved camera properties are written into this structure.

Return:

`PAL::SUCCESS`	If the call succeeds.
`PAL::FAILURE`	If the call fails.
`PAL::ERROR_CAMERA_NOT_INITIALIZED`	If this call is made before an Init call.

5.4.11 SaveProperties Call

```
PAL::Acknowledgement SaveProperties(const char*fileName);
```

Overview:

Saves the current camera properties into a text file.

Arguments:

Name	`fileName`
Type	Read
Optional	No
Description	The file into which the current camera properties are to be saved. *Note 1:* If the file path does not exist, the call fails. *Note 2:* If the file exists, it is overwritten.

Return:

`PAL::SUCCESS`	If the call succeeds.
`PAL::FAILURE`	If the file can't be created with write access.

`PAL::ERROR_CAMERA_NOT_INITIALIZED`	If this call is made before an Init call.

5.4.12 LoadProperties Call

```
PAL::Acknowledgement LoadProperties(const char* fileName,
PAL::CameraProperties* data = 0);
```

Overview:

This call reads an existing properties file saved with the `SaveProperties` call, then loads the values into memory and also sets them on the camera by calling `SetCameraProperties` internally.

Arguments:

Name	`fileName`
Type	Read
Optional	No
Description	The file from which camera properties are to be read.

Name	`data`
Type	Write
Optional	No
Description	The structure into which the camera parameters read from file are loaded.

Return:

`PAL::SUCCESS`	If the call succeeds.
`PAL::FAILURE`	If the properties file is not found.
`PAL::INVALID_PROPERTY_VALUE`	If data in the properties file is corrupted.
`PAL::ERROR_CAMERA_NOT_INITIALIZED`	If this call is made before an Init call.

5.4.13 GetAvailableResolutions Call

```
std::vector<PAL::Resolution> GetAvailableResolutions();
```

Overview:

This call is used to access the list of valid resolutions PAL can support at runtime.

Arguments: None

Return: A `std::vector` of `Resolution` structures representing the valid resolutions that can be set on the `resolution` camera property.

5.5 Point Cloud

This section details the API calls related to point clouds. First let us take a look at the point structure.

5.5.1 Point Structure

```
struct Point
{
      float x,y,z;
      unsigned char r, g, b, a;
};
```

This structure is used to represent a point in the point cloud.

x, y & z members represent the coordinates in the x, y & z dimension respectively. Together they represent the location of a 3D point (corresponding to a 2D panorama pixel, the values will be returned in centimeters).

Note: The points more than 20m away from PAL are discarded.

r, g, b & a members represent the red, blue, green and alpha (transparency) colour values respectively.

Frame of reference: The point cloud is computed with the following frame of reference:

- X-axis points towards the first column of the left panorama image and this implies the center of the left panorama points towards the negative

direction of the X-axis.

- Y-axis is perpendicular to the base of the camera and pointing upwards.
- Z-axis can be determined by the right-hand rule.

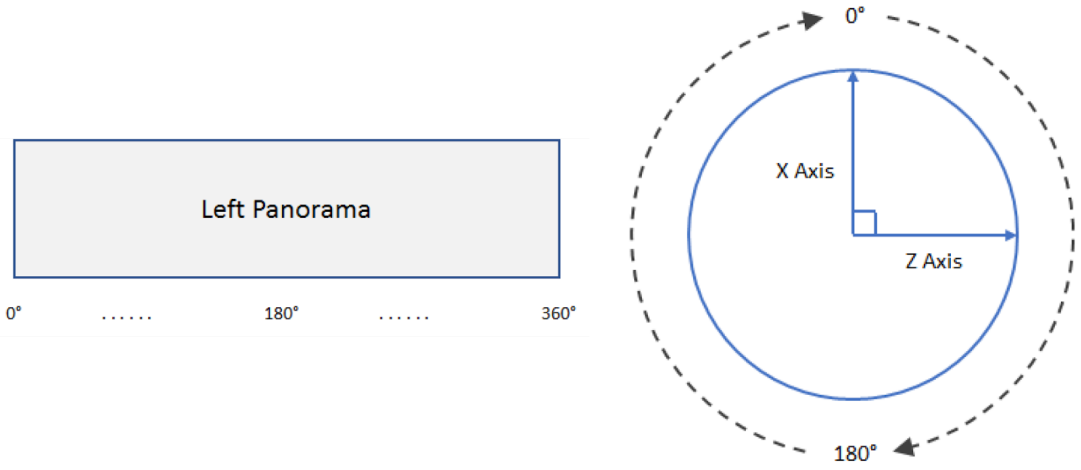


Figure 5.1: Left – Mapping columns of left panorama image into 360° horizontal field of view in the real world. Right – Top view of the left panorama wrapped around a cylinder. It also shows the X-axis mapping to 0° (i.e. start / end of the left panorama) in the real world.

5.5.2 GetPointCloud Call

```
PAL::Acknowledgement GetPointCloud(std::vector<PAL::Point> *pc,
      timeval *timestamp = 0, PAL::Image *left = 0, PAL::Image *right =
      0, PAL::Image *depth = 0, PAL::Image *disparity = 0);
```

Overview:

Allows users to retrieve the point cloud information along with optional left, right, disparity and depth panoramas. For every pixel in the panorama, its corresponding location in the 3D world the (x,y,z) position and (r,g,b) information is calculated.

It calls `GrabFrames` internally and computes the point cloud information from the last known images. It returns after the point cloud computation is finished for the latest frames.

These points would be pushed back into the provided vector. If the vector has prior information, that information stays intact. If you don't want the data to be accumulated, clear the vector before sending it as the argument.

Note: The Point member 'a' representing "alpha" value is set to 255 (fully opaque) for all returned points in the current implementation of the API.

Arguments:

Name	`pointCloud`
Type	Write
Optional	No
Description	Pointer to a valid std::vector<PAL::Point> into which the points will be pushed.

Name	`timestamp`
Type	Write
Optional	Yes, the default value is 0.
Description	Pointer to a valid timeval struct into which the capture time will be pushed. Time is represented as epoch time.

Name	`left`
Type	Write
Optional	Yes, the default value is 0
Description	Panorama image as seen by the left view in a stereo pair. This will be a 8 bit, 3 channel image. Equivalent to CV_8UC3 of OpenCV.

Name	`right`
Type	Write
Optional	Yes, the default value is 0
Description	Panorama image as seen by the right view in a stereo pair. This will be a 8 bit, 3 channel image. Equivalent to CV_8UC3 of OpenCV.

Name	`depth`
Type	Write
Optional	Yes, the default value is 0
Description	Panorama image representing the depth perceived by the stereo system. This image will be 32 bit (float), 1 channel. Equivalent to CV_32FC1 of OpenCV. The unit of measurement is centimeter.

Name	`disparity`
Type	Write
Optional	Yes, the default value is 0
Description	Panorama image representing the disparity perceived by the stereo system. This image will be 16 bit, 1 channel. Equivalent to CV_16SC1 of OpenCV.

Return:

`PAL::SUCCESS`	If the call succeeds.
`PAL::FAILURE`	It can happen in the following cases: 1. When the camera fails to capture the image. 2. When there's an internal error in point cloud computation.
`PAL::INVALID_PROPERTY_VALUE`	If this call is made before an Init call.

5.5.3 SavePointCloud Call

```
PAL::Acknowledgement SavePointCloud(const char* fileName,
std::vector<PAL::Point> *pointCloud = 0);
```

Overview:

Saves point cloud information (represented as an STL vector of Point) into a .ply file. These .ply files are recognized by 3D tools like MeshLab, Blender, etc.

Arguments:

Name	`fileName`
Type	Read
Optional	No
Description	The file into which the point cloud has to be saved. *Note 1:* The file name is appended with a .ply extension if not already provided by the user. *Note 2:* If the provided file path does not exist, then the corresponding directories are created and the file is saved in the provided path. *Note 3:* If the file already exists, it will be

	overwritten.
Name	`pointCloud`
Type	Read
Optional	Yes, default value is `NULL`
Description	The pointer to a vector of points that should be saved. When this argument is not provided or `NULL`, `GetPointCloud` function would be called internally and the latest point cloud will be saved.

Return:

`PAL::SUCCESS`	If the call succeeds.
`PAL::FAILURE`	If the file cannot be created with write access.
`PAL::IGNORED`	When the `pointCloud` vector provided is empty.

5.5.4 Point Cloud Example

```
std::vector<PAL::Point> pc;
if (PAL::GetPointCloud(&pc) == PAL::SUCCESS)
{
      PAL::SavePointCloud("point_cloud.ply", &pc);
}
```

This example demonstrates how to retrieve the latest computed point-cloud from the API and save it to a .ply file.

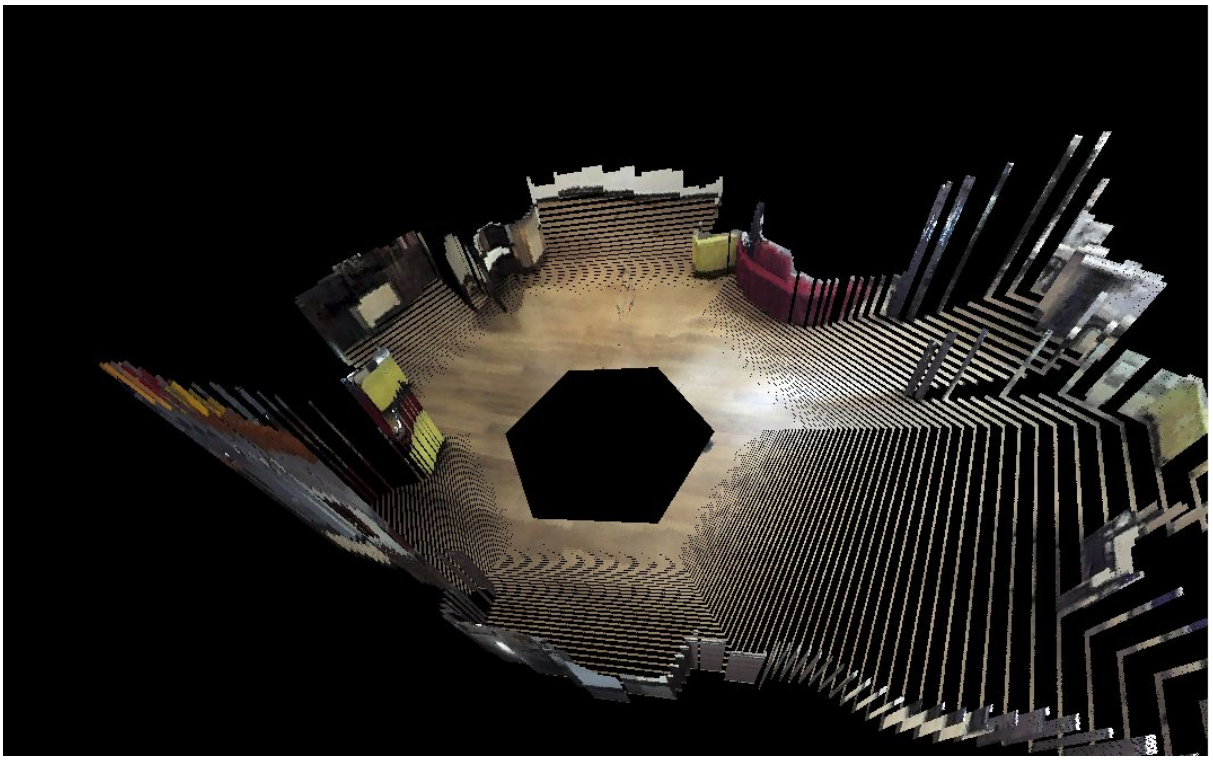


Figure 5.2: The .ply file is loaded in Meshlab and the point cloud is visualised

5.7 Acknowledgement Enum

```
enum Acknowledgement
{
	IGNORED,
	SUCCESS,
	FAILURE,
	INVALID_PROPERTY_VALUE,
	ERROR_CAMERA_NOT_INITIALIZED
};
```

This enum is used to indicate if the API call has been successfully executed or not. Ideally all the API Calls should return `PAL::SUCCESS`. It is recommended to check if the return value is `PAL::SUCCESS` or not – for every API function call.

6. ROS Support

The PAL SDK supports a wrapper package ./`dreamvu_pal_camera` folder which allows you to easily integrate with Robot Operating System (ROS). This section describes this wrapper in more detail.

6.1 Prerequisites

`dreamvu_pal_camera` has the following prerequisites:

- PAL SDK: Should be installed as described in Section 3.
- ROS Melodic: Should be installed as described here.
- Please make sure you install one of the desktop versions (they contain rviz).
- Once ROS is installed. You can set up your catkin workspace as described in section 3 here.

Note: `dreamvu_pal_camera` is a catkin package. It depends on following ROS packages: `cv_bridge`, `image_transport`, `roscpp`, `sensor_msgs`, `std_ msgs`, `message_generation`, `message_runtime`

6.2 Building the ROS package

Please follow the steps below for building `dreamvu_pal_camera`:

1. Ensure you've run the `./install.sh` script as part of PAL SDK installation. This updates the location of the PAL libraries and headers in the `CMakeLists.txt` file of `./dreamvu_pal_camera/ package`.
2. Copy or move the `dreamvu_pal_camera` folder into the `src/` folder of your catkin workspace. For the purpose of this document it is assumed to be in the home directory. The command for the same is as follows:

```
cp -r ./dreamvu_pal_camera ~/catkin_ws/src
```

3. Build the package by running the following code in a terminal.

```
cd ~/catkin_ws
catkin_make
source ./devel/setup.bash
```

6.3 Capture node

The `dreamvu_pal_camera` package has a node of `capture` type with the name `pal_camera_node`. It handles all of the publishing and subscribing responsibilities. The following table provides a list of ROS topics and the corresponding information being published and subscribed by it.

Data published	ROS Topic	Message Type
Left panorama	/dreamvu/pal/get/left	sensor_msgs::Image
Right panorama	/dreamvu/pal/get/right	sensor_msgs::Image
Depth panorama	/dreamvu/pal/get/depth	sensor_msgs::Image
PointCloud	/dreamvu/pal/get/pointcloud	sensor_msgs::PointCloud2
Data Subscribed	ROS Topic	Message Type
PAL Properties	/dreamvu/pal/set/properties	dreamvu_pal_camera::Properties

Table 7.1: List of ros topics published and subscribed by pal_camera_node

Note 1: The point cloud distance information (x, y & z data) is published in meters by the `pal_camera_node`.

6.3.1 Properties Message

The custom message `dreamvu_pal_camera::Properties` maps directly to the CameraProperties structure in the PAL API. The usage is straightforward and similar to the API. Usage of the flags property is described in sections 5.4.6.1 to 5.4.6.3

The deviations from the API are `color_space` which is interpreted as RGB / YUV444 when set to 0 / 1 respectively and `power_line_frequency` as AUTO /_50HZ / _60HZ when set to 0 / 1 / 2.

A sample node that publishes `dreamvu_pal_camera::Properties` messages to the `/dreamvu/pal/set/properties` topic is provided in `./dreamvu_pal_camera/src/` for convenience.

6.3.2 Launching the "capture" node

Once the package has been built as described in the previous section, to run the node, do the following. First, start the `roscore` process by running the following in your terminal:

```
roscore
```

Then, open another terminal, and source the bash setup file for that catkin workspace.

```
source ./devel/setup.bash
```

Note: This step will have to be repeated every time a new terminal is opened in the catkin workspace.

Now, run the following command to start the `pal_camera_node` of type `capture`:

```
rosrun dreamvu_pal_camera capture
```

Now, the node will advertise the list of topics mentioned in Table 7.1 to the master. Once any subscriber is registered to a topic, the node will start publishing messages corresponding to that topic.

6.4 Launching rviz preview

The `dreamvu_pal_camera` provides a launch file to preview the camera information being published in rviz. To launch rviz for previewing the camera feed run the following command in a terminal:

```
roslaunch dreamvu_pal_camera pal_rviz.launch
```

This should open an rviz window as shown below:

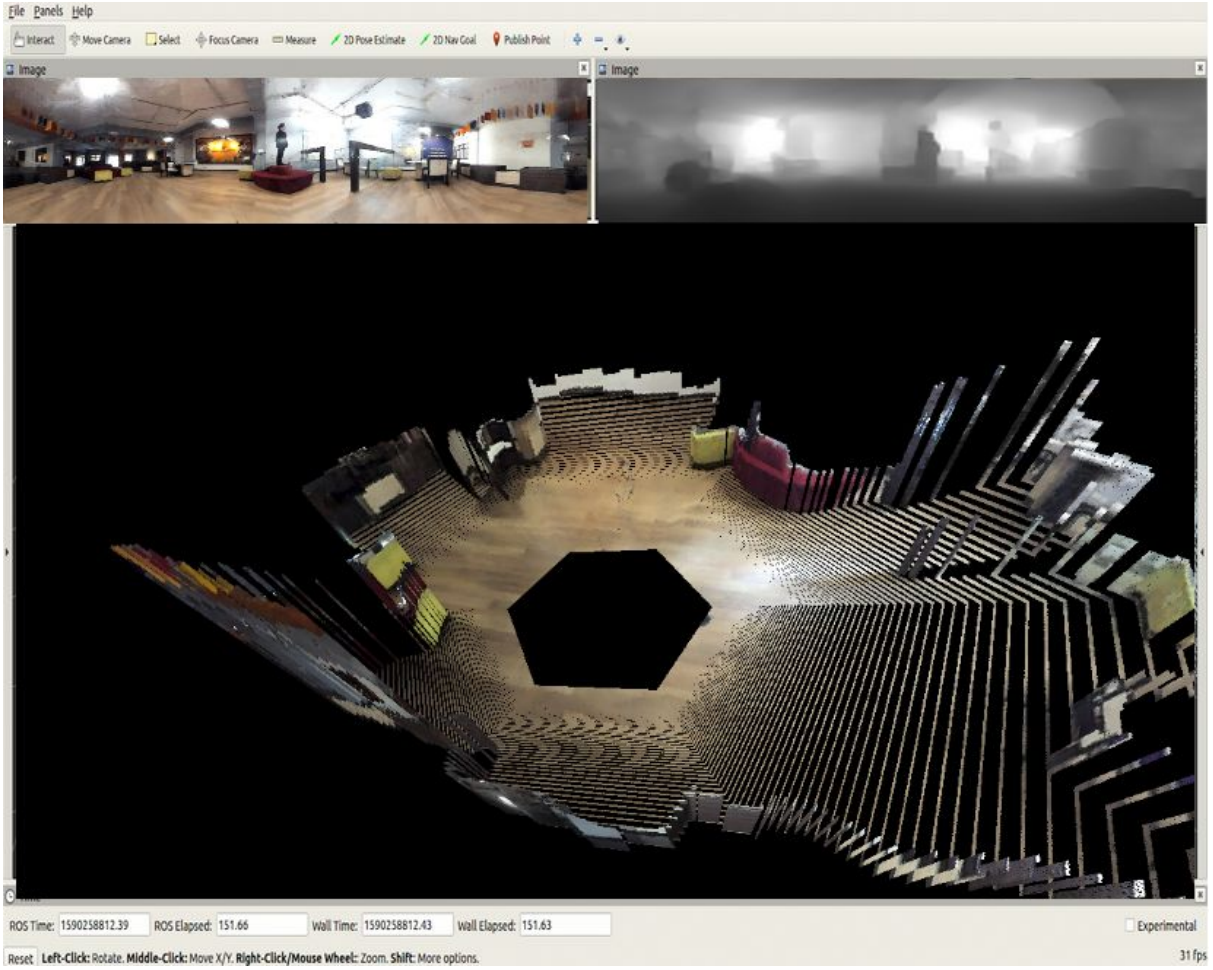


Figure 6.1: rviz preview of PAL data

6.5 Launching ROS Wrapper with Properties File

The `pal_camera_node` node launches with default properties defined in the SDK. If users want to initialize it with different values they can do so using a properties file. This file can be obtained as follows:

- File saved while Explorer application quits
- File saved using the SaveProperties call.

The `pal_camera_node` node expects a file with name and path as specified below (relative to ROS home environment variable):

```
../catkin_ws/src/dreamvu_pal_camera/src/SavedPalProperties.txt
```

Note: If the properties file is corrupted or cannot be opened then the default

parameters are loaded from the SDK.

6.5.1 Copying the properties file to the default location

If you've followed the same ROS setup process as noted in this document you can just copy the `SavedPalProperties.txt` file saved by Explorer application to the following path:

```
~/catkin_ws/src/dreamvu_pal_camera/src/SavedPalProperties.txt
```

6.5.2 Explicitly specifying path to the properties file

Users can also explicitly set the file path by redefining the `PROPERTIES_FILE_PATH` macro in the pal_camera_node.cpp file. This file is located in the following path in the SDK root folder:

```
dreamvu_pal_camera/src/pal_camera_node.cpp
```

Note: The ROS package will have to be compiled again using catkin_make in this case.

6.5.3 Note about using properties saved by the Explorer

The explorer does not allow users to modify the Color Space setting.

These will have to be manually modified by the user in the `SavedPalProperties.txt` file:

- Color Space can be set to 0 or 1 for RGB or YUV444 respectively.

6.6 Capturing ROS Bag

ROS allows users to record and replay sensor data. Please see this tutorial to understand how to capture and playback ROS bag files.

7. Benchmarking

This section details the benchmarking performed on the API and Explorer application. Particularly the following parameters are benchmarked:

1. Memory
2. Frame Rate
3. CPU Cycles
4. Latency

7.1 System Configuration

The configuration of the system used for benchmarking is as follows:

Processor	Intel® Core™ i7-8750H CPU @ 2.20GHz × 12
OS Type	Ubuntu 18.04 64 bit
RAM	16GB (2x 8Gib DIMM Synchronous 2133 MHz)
Cache Details	384KiB L1 cache 1536KiB L2 cache 12MiB L3 cache
OpenCV Version	3.4.4

Note: All the measurements in the benchmarking section were made with properties set to default values; However, the following exceptions apply:

- Auto exposure was disabled.
- Absolute exposure was set to 100 (i.e 10 msec).

This was done to ensure repeatability of the benchmarks under differing scene conditions.

7.2 Memory Usage

This section details the peak memory usage of the API. It was measured by using a minimal application to perform just the operations / API calls under consideration.

Operations Performed in Application	Peak Memory Usage (MB)
	Resolution: 3544x1216
Init and Destroy	60
Set Properties	60
GrabFrames for Left & Right Panoramas	60
GrabFrames for Left, Right & Disparity Panoramas	60
GrabFrames for Left, Right, Disparity & Depth Panoramas	60
Get PointCloud	115

7.3 Frame Rate

This section details the average frame rate measured under different conditions using the PAL API.

7.3.1 GrabFrames Call

This section details the average frame rate measured for different combinations using the `GrabFrames` call averaged over 50 frames.

	Panorama Queried	FPS
Resolution: 3544x1216	Left & Right	14
	Left, Right, & Normalized Disparity	1.6
	Left, Right, Normalized Disparity & Depth	1.6

7.3.2 GetPointCloud Call

This section details the average frame rate measured using the `GetPointCloud` call averaged over 50 calls.

	FPS
Resolution: 3544x1216	1.5

7.4 CPU Cycles

This section details the clock cycles taken by the `GrabFrames` call for different combinations averaged over 50 calls.

The table below lists values in Millions of CPU cycles.

	Panorama Queried	CPU Cycles
Resolution: 3544x1216	Left & Right	7
	Left, Right, & Normalized Disparity	13
	Left, Right, Normalized Disparity & Depth	17

7.5 Latency

This section lists the latency measured using a minimal OpenCV application similar to the disparity code sample provided in the SDK that makes a `GrabFrames` call to query Left, Right, Disparity & Depth panoramas.

The table below lists values measured in milliseconds

	Latency
Resolution: 3544x1216	344

Note: For values marked with * the latency observed was smaller than what could be measured by our setup.

7.5.1 Explorer

The rgb latency observed in the Explorer application in default mode is around 414 milliseconds (with filterspots disabled).

8. Known issues and troubleshooting

8.1 Unable to grab frames

- This can happen when the Ethernet connection is not properly established. Make sure PAL is powered on and is connected to the host machine via ***DirectConnect*** through LAN cable.
- If the camera is not able to send data. Please ssh into the device and following commands in such case:
    - ssh -XC dreamvu@192.168.0.175
    - Password to login is dreamvu
    - ./pal_stream

8.2 Running multiple PAL applications

Running multiple applications simultaneously is not supported by the SDK currently. When an application is communicating with PAL, and if another application tries to communicate with PAL, the behaviour is undefined. It is recommended to use a single application at a time.

8.3 Unable to find PAL .so library error

This error might happen when the .so file is not found in the expected path to the application. In order to resolve the issue please add the path of the lib folder of the SDK to the environment variable LD_LIBRARY_PATH. For eg:

```
export LD_LIBRARY_PATH=$LD_LIBRARY_PATH:/path/to/lib/folder
```

8.4 Screen freezes with cv::imshow

This is a known issue. Sometimes cv::waitKey will not return when the PAL images are streamed over ethernet. In this version, cv::imshow is not supported. Instead of

opencv based imshow, opengl texture based code sample is provided in the SDK

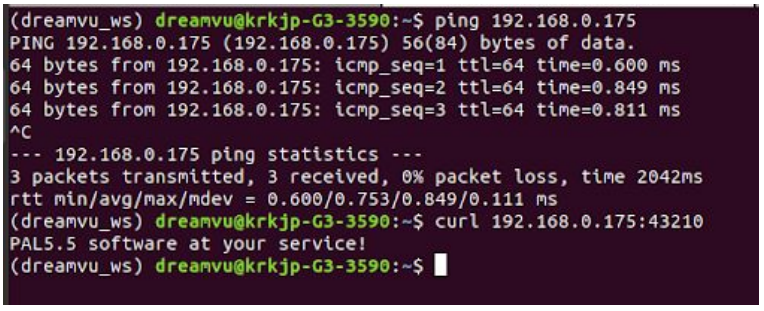
8.5 PAL::Init takes indefinite time

When the streaming is not started, PAL::Init might take indefinite time. In such a case exit the app (press ctrl+C), restart the camera (remove and re-insert the power cable if required) and make sure that the camera is connected to your computer.

The following shell commands will verify if the camera is up and connected...

- ping 192.168.0.175 (press ctrl+C to stop the process)
- curl 192.168.0.175:43210

  The above commands should produce an output like this...



8.6 PAL::GrabFrames return blank images

The sensor might still be adjusting to the light when the network stream started. Discard the initial few (5-10) calls and check the images from later calls.