DreamVu

PAL Documentation

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

DreamVu Inc.
2150 N 1st Street, Office 454, San Jose, CA, 95131
www.dreamvu.com

Table of Contents

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 1.1: Front view of PAL [USB]

2.1 Hardware Requirements:

- Processor: Intel based architecture
- Ram: 4 GB
- USB 3.0 port

2.2 Software Requirements:

- Operating System: Ubuntu 18.04 64 bit.
- OpenCV 3.4.4 and OpenCV Contrib 3.4.4 libraries.
- Python 3.6 libraries (pytorch, numpy, PIL, etc.)

3. Installation Procedure

3.1 Installing Dependencies

- PAL API heavily depends on OpenCV, Opencv_Contrib libraries
    - Follow **opencv.sh** script present in the *installations* folder of SDK.

3.2 Installing the PAL & Python 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
./install.sh
```

4. Quick Start

4.1 Explorer Application

Once the 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.

- Plug in PAL into a USB 3.0 port.
- Open Explorer directory and launch Explorer Application. See the Explorer Section 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.

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 ../lib/libPAL_CAMERA.so
../lib/libPAL_DEPTH.so `pkg-config --libs --cflags opencv` -O3 -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,
int cameraIndex = -1);
```

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.*

Name	`camera_index`
Type	Read
Optional	Yes, with default value -1
Description	Device index of the camera. (same as the index for opencv camera capture). This index provides a way of identifying different camera inputs connected to a system. When set to the default value -1, the API backend automatically tries to detect the correct index of PAL and uses it internally. In case the automatic detection does not work, or users want to explicitly specify the camera index, they can do so using this argument.

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.

5.2.1.1 Determining Camera Index Manually

Every time a camera is connected to the machine a file is created in `/dev/` folder with a name like `video0, video1,` etc. A new file with an incremented number is created for each new camera added. This number is the camera index.

To determine which `video<No.>` file corresponds to PAL, please navigate to the `/dev/` folder. Then unplug & plug the camera from the USB port to see which `video<No.>` file is removed & added respectively. You should give the system 5-10 seconds each time you unplug / plug the camera for the file changes to take effect. Once the file is identified, the number at the end of the filename is the camera index corresponding to PAL.

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	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	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 USB 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 (if `asynchronous` is set to true).

5.3.3 Synchronize Call

```
void Synchronize();
```

Overview:

This is a "barrier call" that blocks execution until all the threads have completed computation for the latest frame.

This call is useful to force "sync" computation of panoramas when `GrabFrames` was called asynchronously. After this call is returned it is assured that no memory will be updated by PAL API until another `GrabFrames` call is made.

Arguments: None
Return: Void

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 ColorSpace Enum

```
enum ColorSpace
{
      RGB,
      YUV444
};
```

This enum is used for configuring the color space of PAL.

5.4.3 PowerLineFrequency Enum

```
enum PowerLineFrequency
{
      _AUTO,
      _50HZ,
      _60HZ
};
```

This enum is used for configuring the power line frequency in camera properties.

Note: Sometimes AUTO mode cannot detect the powerline frequency correctly. When flickering is observed in AUTO mode, please choose the appropriate power line frequency based on your region. For example, in India it is 50Hz.

5.4.4 Projection Enum

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

This enum can be used to indicate if the PAL 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.

5.4.5 DisparityComputation Enum

```
enum DisparityComputation
{
      FAST = 0,
      HIGH_QUALITY_A = 1,
};
```

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

5.4.6 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;
      ColorSpace color_space;
      PowerLineFrequency power_line_frequency;

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

```
        int   fov_start;
        int   fov_end;

        Projection projection;
        DispartyComputation computation;
};
```

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	15
Saturation	0	60	1	32
Gamma	40	500	1	220
Gain	0	100	1	0
White balance temperature	1000	10000	50	5000
Sharpness	0	127	1	0
Exposure	1	10000	1	312
FOV start	0	360	1	0
FOV end	0	360	1	360

5.4.6.1 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.

5.4.6.2 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.

5.4.6.3 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.

5.4.6.4 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
```

5.4.6.5 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.

5.4.6.6 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.

5.4.6.7 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
```

5.4.6.8 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.

5.4.6.9 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`.

5.4.6.10 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`.

5.4.6.11 Resolution

The resolution of the panoramas returned in `GrabFrames`. This property affects the processing that follows like disparity, depth & point-cloud. It can be one of the values mentioned below:

```
PAL::AvailableResolutions::_3440x713
PAL::AvailableResolutions::_2304x480
PAL::AvailableResolutions::_1720x356
PAL::AvailableResolutions::_432x128
```

The corresponding resolutions are as shown in the following table

The default value of resolution can be accessed as follows:

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

Note: In the current version of the SDK the default resolution is set to `PAL::AvailableResolutions::_3440x713`.

5.4.6.12 Color Space (color_space)

The color space of the panoramas returned in `GrabFrames`. It can be set to any of the ColorSpace enumerations. The default value can be accessed as follows:

```
PAL::CameraProperties::DEFAULT_COLOR_SPACE
```

5.4.6.13 Power Line Frequency (power_line_frequency)

Powerline frequency is used to specify the frequency of electricity being supplied to any artificial light source illuminating the scene. This is required to correct a flickering effect observed sometimes depending on the exposure time, the frequency of the power line and the light source involved.

See PowerLineFrequency enum for more information about what values can be set for this property. The default property values can be accessed as follows:

```
PAL::CameraProperties::DEFAULT_POWER_LINE_FREQUENCY
```

5.4.6.14 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
```

5.4.6.15 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
```

5.4.6.16 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.

5.4.6.17 Setting Field of View (fov_start & fov_end)

PAL allows you to select a horizontal field of view less than 360°. This can be done by setting `fov_start` and `fov_end` values, which define the start and end of the region of interest respectively.

The default property values can be accessed as follows:

```
PAL::CameraProperties::DEFAULT_FOV_START
PAL::CameraProperties::DEFAULT_FOV_END
```

Note 1: It is okay for `fov_start` to be greater than `fov_end`. This is useful, for example, when trying to select a region that wraps around the right border of the default view -- i.e a region which begins close to the right edge of the default view, wraps around, and stops somewhere close to the left edge of the default view.

For example:

This is how a default panorama view looks like:

0° 90° 180° 270° 360°

If fov_start is selected as 90° and fov_end as 270° the final output you'll get will look like this:

Note 2: Disparity cannot be computed when the absolute angular difference between `fov_start` and `fov_end` is less than 2°. i.e the selected field of view is very

small.

5.4.6.18 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
```

5.4.6.19 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
```

5.4.7 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,
	COLOR_SPACE = 0x800,
	POWER_LINE_FREQUENCY = 0x1000,
	VERTICAL_FLIP = 0x2000,
	FILTER_DISPARITY = 0x4000,
	FILTER_SPOTS = 0x8000,
```

```
        FOV = 0x10000,
        PROJECTION = 0x20000,
        DISPARITY_COMPUTATION = 0x40000,
        ALL = 0x7FFFF,
};
```

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::AvailableResolutions::_1720x356;
p.color_space = PAL::RGB;
p.power_line_frequency = PAL::_50HZ;
p.vertical_flip = true;
p.filter_disparity = false;
p.filter_spots = true;
p.fov_start = 180;
p.fov_end = 270;
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 3km 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.

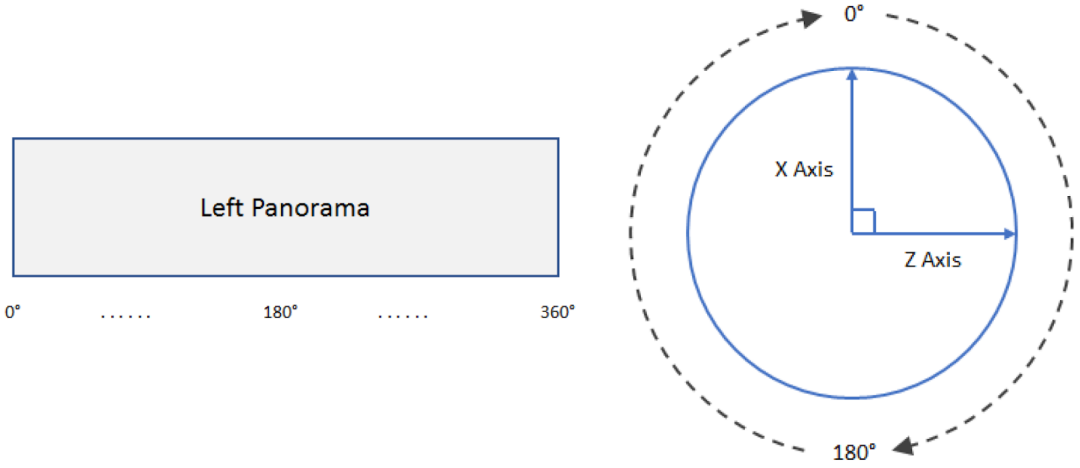

Figure 5.1: Left – Mapping columns of left panorama image into 360° horizontal field of view in the real world. Right – Top view of the left panorama wrapped around a cylinder. It also shows the X-axis mapping to 0° (i.e. start / end of the left panorama) in the real world.

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.

This is a synchronous call. It calls `GrabFrames` internally and computes the point cloud information from the latest images. It returns only 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

5.6 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. Known issues and troubleshooting

6.1 Unable to grab frames

This can happen when the USB cable is not properly inserted, or when the USB cable is pulled out and replaced in a different slot. You are required to keep the USB cable in the same port when the application is running. Once the cable is connected, allow the Linux OS 5-10 seconds to recognize the hardware changes, and then run the code.

6.2 Camera initialization failure

This can happen when the USB cable is inserted into the USB2.0 slot, instead of USB3.0 slot. USB2.0 is not fast enough to support the bandwidth required by PAL. You should make sure that the cable is inserted into the USB3.0 slot.

6.3 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.

6.4 Unable to find PAL .so libraries 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
```

7. Explorer Application

The Explorer is an application provided to the users to quickly explore the features of PAL. It allows you to capture images, videos, and change camera properties. When the camera is connected and the Explorer application is opened you should see an output as shown below displaying the left, right and depth panoramas.

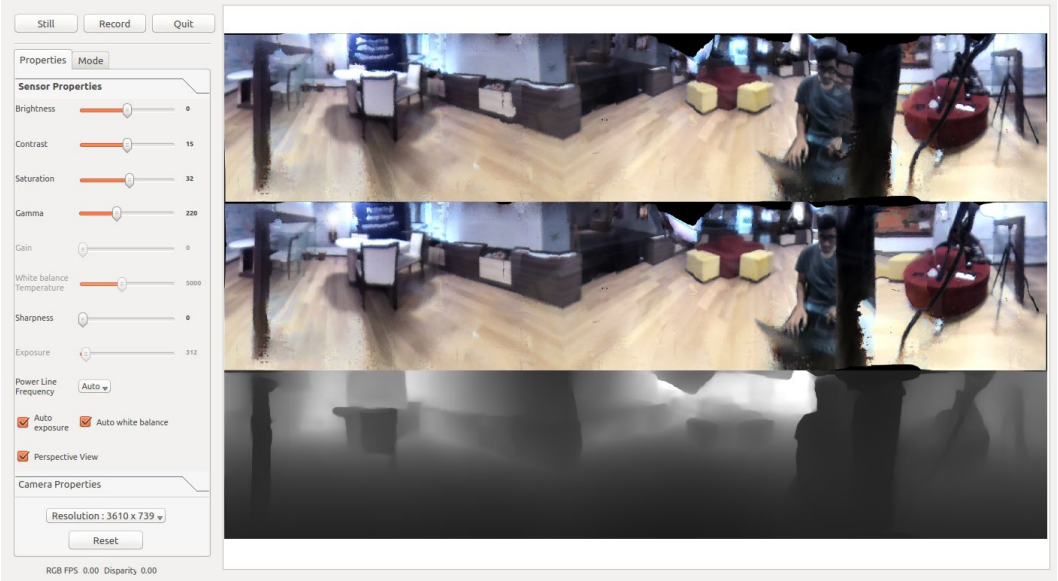

Figure 7.1: Default view of the explorer

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

7.1 Viewing modes

There are three viewing modes in the Explorer to switch between them you first click on the “Mode” tab highlighted in the image below:

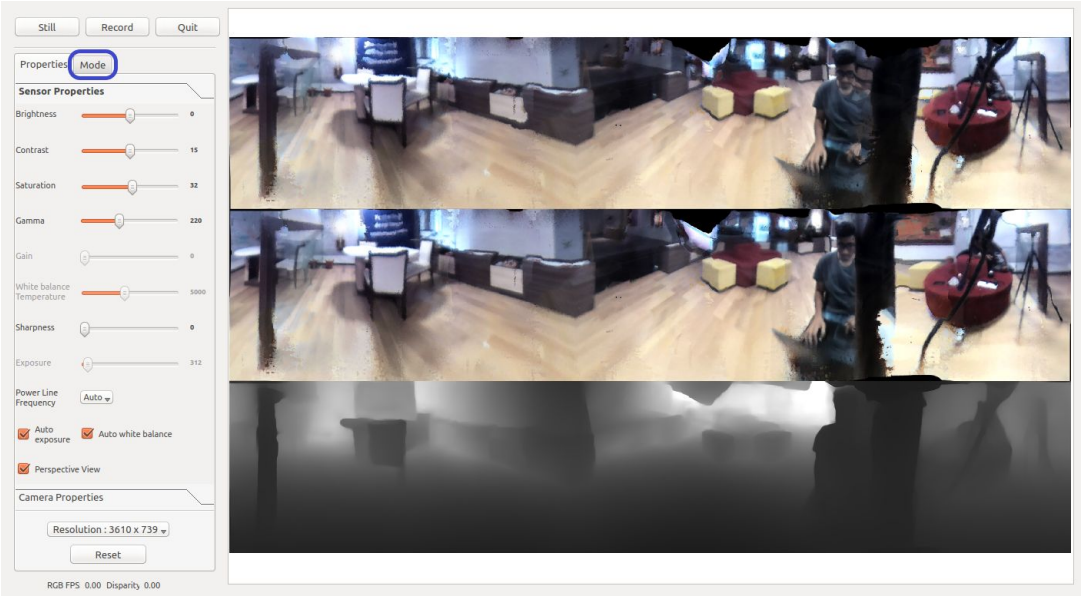

Figure 7.2: Clicking the mode tab

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

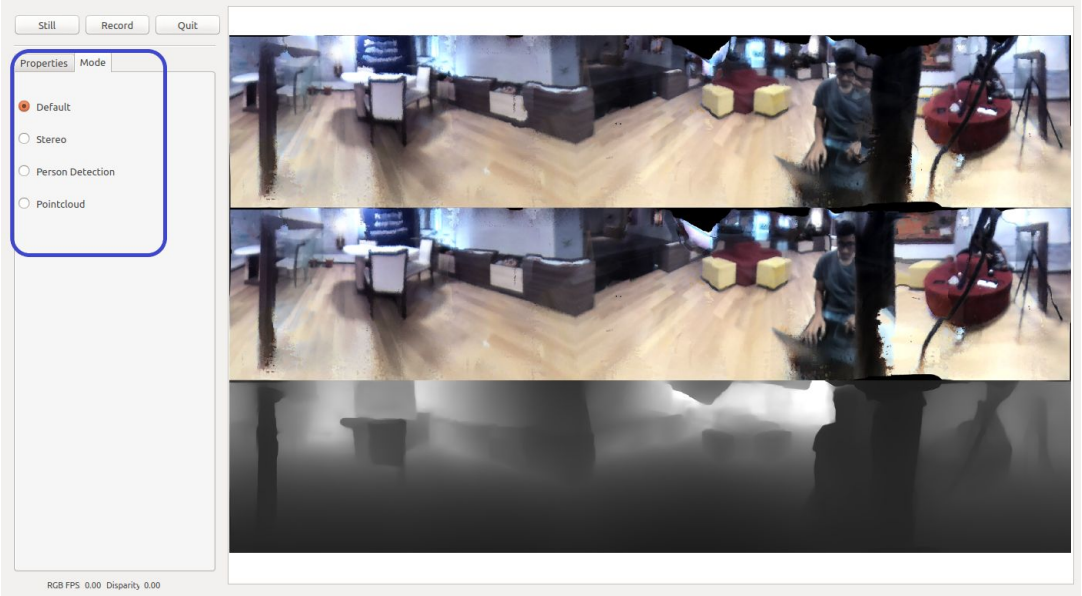

Figure 7.3: Mode pane highlighted

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

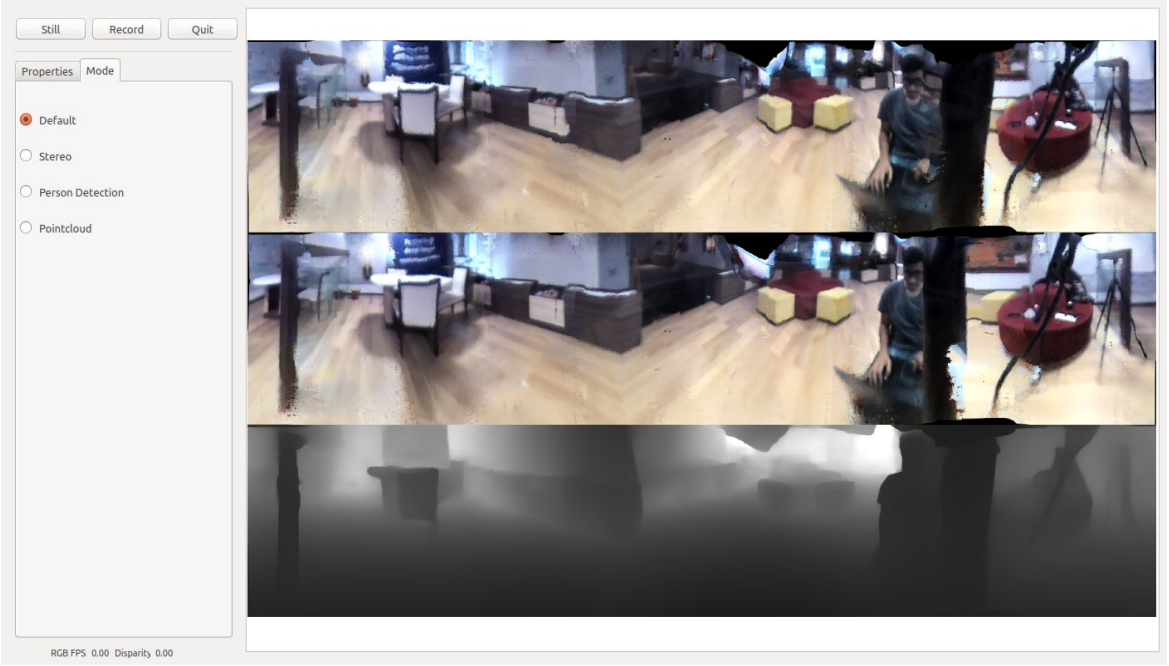

Figure 7.4: default mode

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

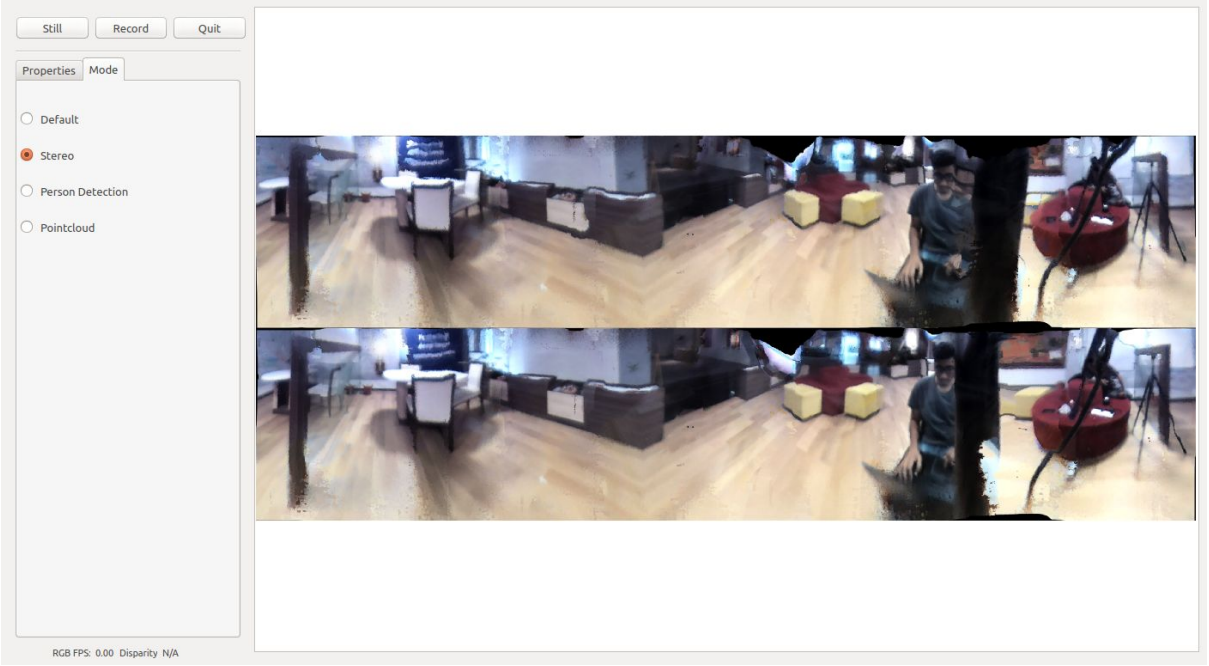

Figure 7.5: Stereo mode

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

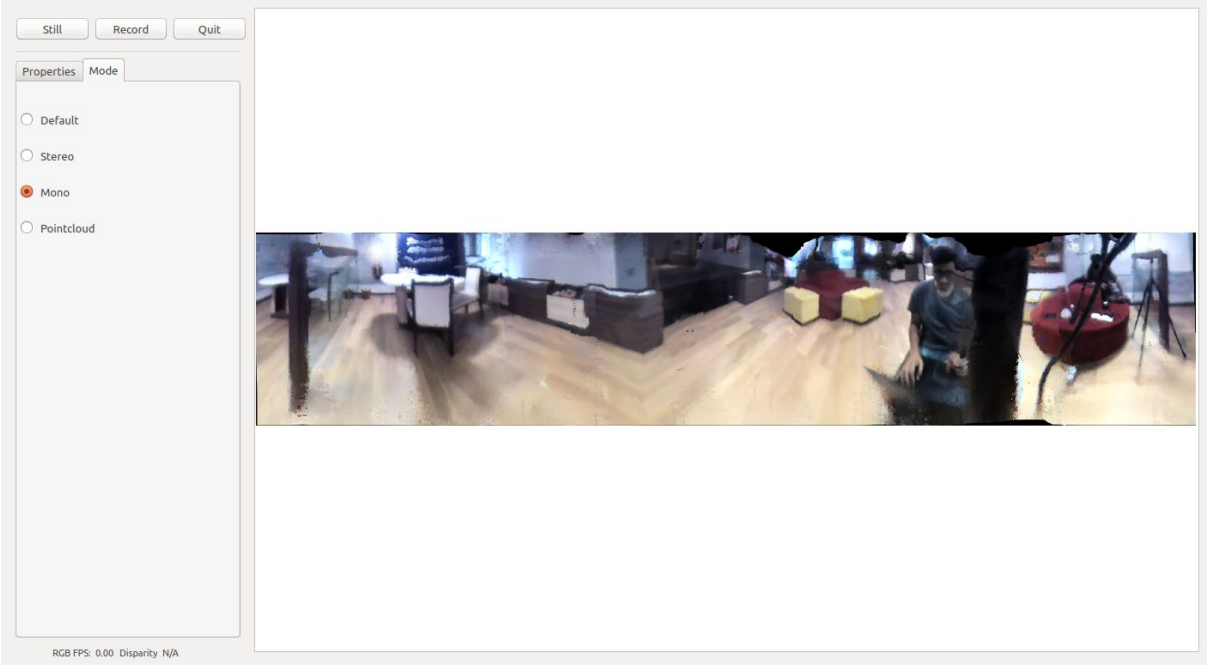

Figure 7.6: Mono mode

Finally, in Point cloud mode, 3D data is visualised in the application.

Figure 7.7 Pointcloud mode

7.2 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:

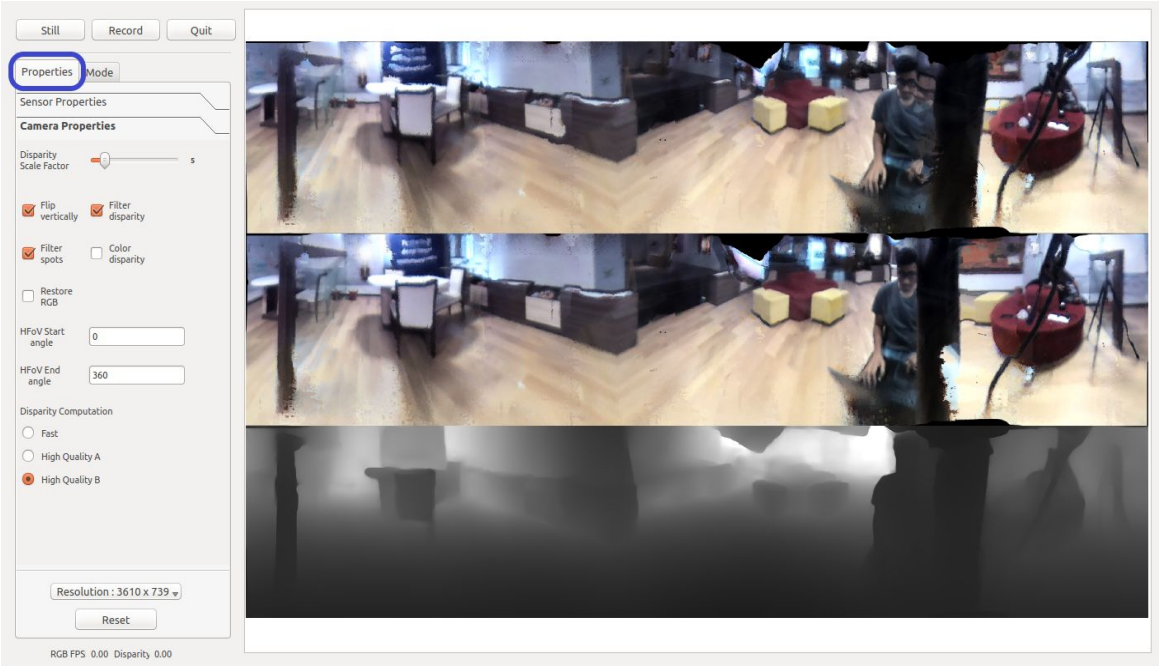

Figure 7.8: Clicking on Properties tab

The camera properties pane is highlighted in the following image:

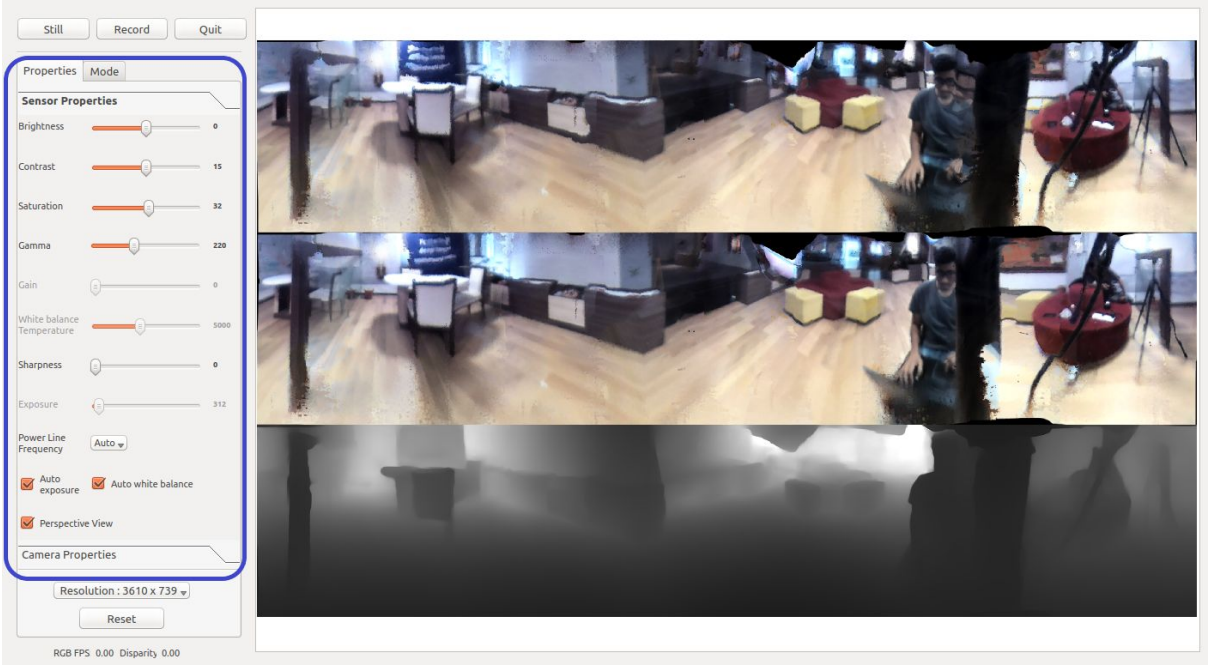

Figure 7.8: Properties pane highlighted

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.

7.2.1 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.

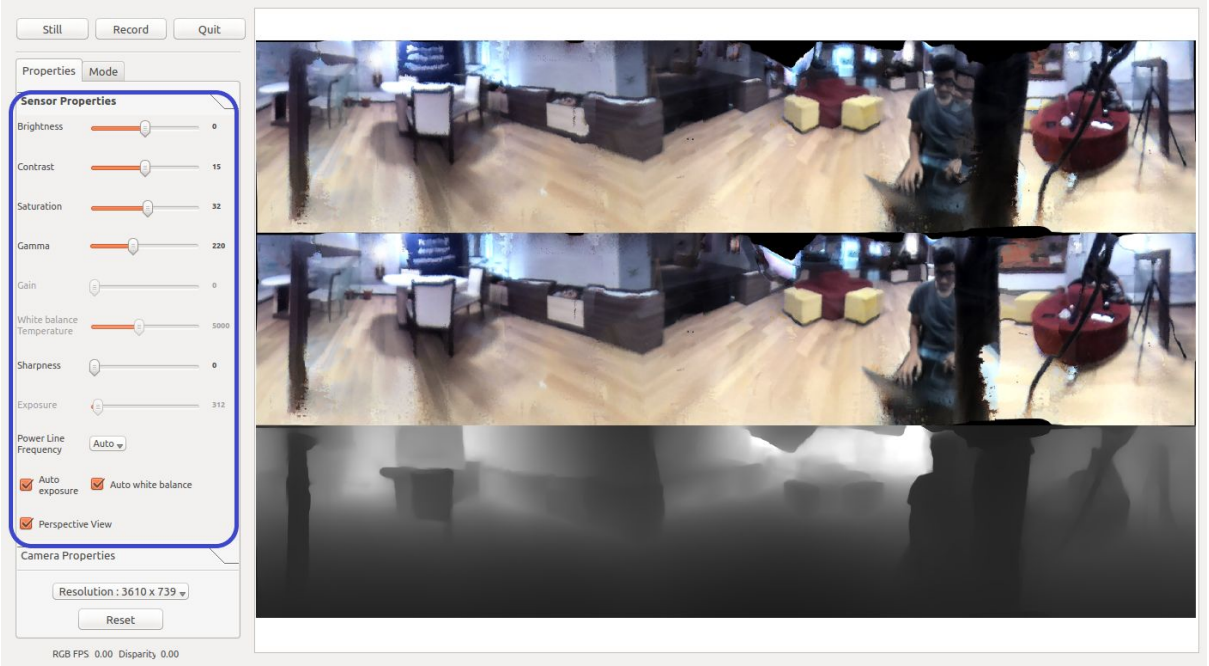

Figure 7.11: Sensor Properties

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.

7.2.2 Camera Properties

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

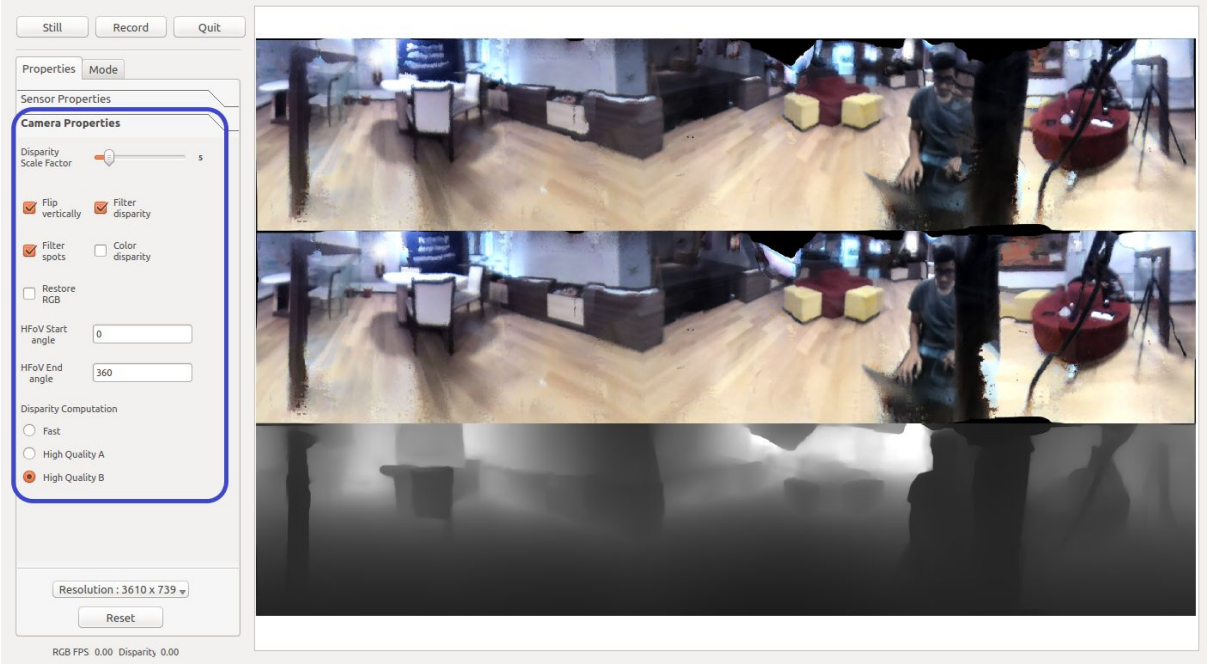

Figure 7.12: Camera Properties

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.

HFoV Start Angle & **HFoV End Angle** can be used to select a subsection of the complete 360° horizontal field of view PAL offers. The inputs should be entered in degrees.

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

7.2.3 Ungrouped elements

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

Figure 7.13: 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.

7.3 Capturing Images

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

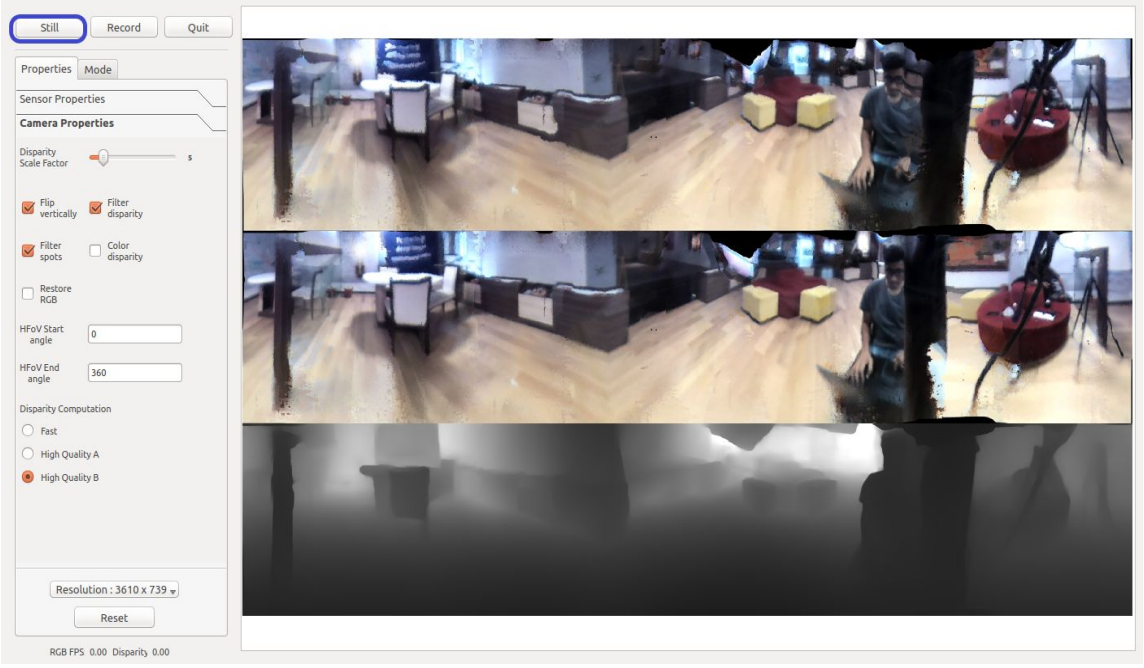

Figure 7.14: Still button highlighted

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

7.4 Recording Video

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

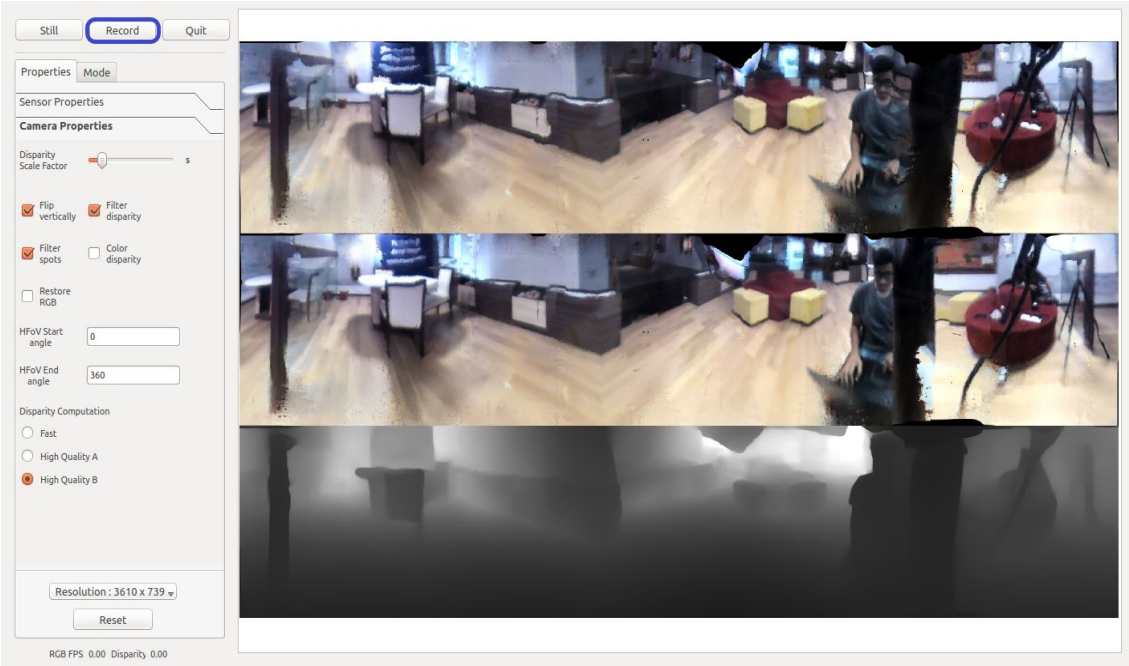

Figure 7.15: Record button highlighted

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

Figure 7.16: 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 uncompressed using YUV420 format in an .avi file container.

Note: When recording a video, changing modes or resolution is not allowed.

7.5 Quit Application

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

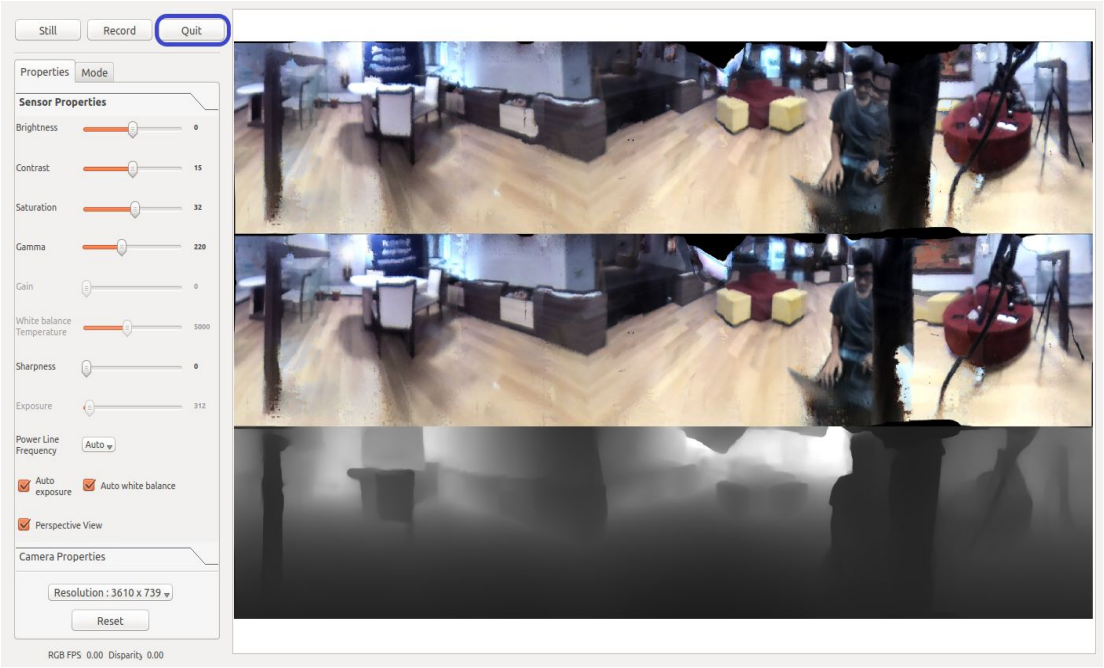

Figure 7.17: Quit button highlighted

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.

8. 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

8.1 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.

8.2 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 (GB)			
	Resolution: 3440x713	Resolution: 2304x480	Resolution: 1720x356	Resolution: 432x128
Init and Destroy	0.8	0.8	0.8	0.8
GrabFrames for Left & Right Panoramas	1.1	1.1	1.1	1.1
GrabFrames for Left, Right & Disparity Panoramas	1.1	1.1	1.1	1.1
GrabFrames for Left, Right, Disparity & Depth Panoramas	1.2	1.2	1.2	1.2
Get PointCloud	1.30	1.25	1.25	1.25

8.3 Frame Rate

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

8.3.1 GrabFrames Call

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

		RGB Color Space	
	Panoramas Queried	Async. mode	Sync. mode
Resolution: 3440x713	Left & Right	10	10
	Left, Right, & Normalized Disparity	10	1.90
	Left, Right, Normalized Disparity & Depth	9.5	1.81
Resolution: 2304x480	Left & Right	20	20
	Left, Right, & Normalized Disparity	20	2.80
	Left, Right, Normalized Disparity & Depth	19.50	2.45
Resolution: 1720x356	Left & Right	39.95	39.52
	Left, Right, & Normalized Disparity	39.84	4.9
	Left, Right, Normalized Disparity & Depth	39.99	4.16
Resolution: 432x128	Left & Right	100.12	100
	Left, Right, & Normalized Disparity	98.57	6.01

	Left, Right, Normalized Disparity & Depth	100.10	5.01

8.3.2 GetPointCloud Call

This section details the average frame rate measured using the `GetPointCloud` call averaged over 50 calls. Please note that this call internally calls `GrabFrames` in a synchronous fashion.

	RGB Color Space
Resolution: 3440x713	1.5
Resolution: 2304x480	2.42
Resolution: 1720x356	4.1
Resolution: 432x128	4.5

8.4 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.

		RGB Color Space	
	Panoramas Queried	Async. mode	Sync. mode
Resolution: 3440x713	Left & Right	149	144
	Left, Right, & Normalized Disparity	199	319
	Left, Right, Normalized Disparity	240	343

	& Depth		
Resolution: 2304x480	Left & Right	121	125
	Left, Right, & Normalized Disparity	137	202
	Left, Right, Normalized Disparity & Depth	125	220
Resolution: 1720x356	Left & Right	55	56
	Left, Right, & Normalized Disparity	65	67
	Left, Right, Normalized Disparity & Depth	65	75
Resolution: 432x128	Left & Right	21	27
	Left, Right, & Normalized Disparity	22	57
	Left, Right, Normalized Disparity & Depth	22	61

8.5 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

	RGB Color Space	
	Asynchronous mode	Synchronous Mode
Resolution: 3440x713	229	688
Resolution: 2304x480	114	344

Resolution: 1720x356	57	172
Resolution: 432x128	< 57*	114

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

8.5.1 Explorer

The rgb latency observed in the Explorer application in default mode is 229 milliseconds.