Version: 0.0.5.4

3DSLiM SDK
User Manual

www.opusmicro.com/

Update History

V 0.0.5.4: @20200601, SDK: V00.905.03

1. Add description of supported FW and FPGA version

V 0.0.5.3: @20200526, SDK: V00.905.03

1. Update SDK to be compatible with both Windows 10 x64 and Ubuntu 18.04 LTS

V 0.0.5.2: @20200521, SDK: V00.905.02

1. Update SDK to be compatible with new calibration method

V 0.0.5.1: @20200507, SDK: V00.905.01

1. Windows version of SDK (Ubuntu 18.04 LTS is confirming compatibility, should be updated soon)

V 0.0.5: @20200415, SDK: V00.905.01

1. Add support for PCD_XYZ_RGB.
2. Add support for Ubuntu 18.04 LTS; however, compatibility of Windows version of SDK is still confirming.

V 0.0.4: @20200409, SDK: V00.904.02

1. Modify depth mapping range: use min_z_cutoff and max_z_cutoff to control the mapping range of depth-map : maps min_z_cutoff --> max_z_cutoff to 65535 -->0.

V 0.0.3: @20200406, SDK: V00.904.01

2. Add depth map data pointer ”p_depth” to output api.
3. Add relative sample code to explain how to use p_depth.
4. Change default value of resampling_factor.

Introduction

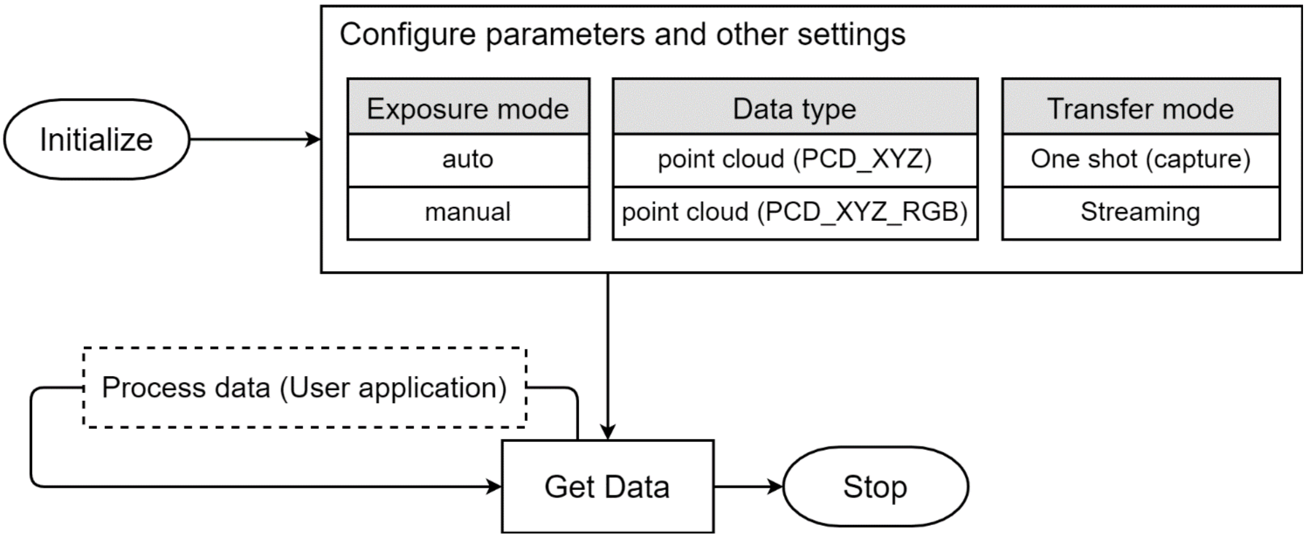
This user manual is intended for users who want to quickly familiarize themselves with 3DSLiM SDK. OPUS provides both a set of API functions (3DSLiM API) and a graphical user interface (Apollo Visualizer) . 3DSLiM SDK supports two ways of outputting point cloud data. One way is to capture a point cloud, it's slow but guarantee the point cloud is generated in the moment you call the function. The other way is to stream point clouds, it's fast but you might get the previous data in the buffer. Please refer to the sample code for more detail.

Notice: If your 3D camera module is mounted with an extra IR camera and connected with another USB cable, note that we do not provide sample code for accessing that IR camera. We suggest that you use 3rd party library (ex. OpenCV) to access it.

Supported platforms and environment

- OS: Windows 10 x64, Ubuntu 18.04 LTS
- Language: C/C++ with .h and .lib (.a)
- Compile scenario: VS2017, release, x64 (Windows), g++ (Ubuntu)
- FW version: V00.909
- FPGA version: 202003311934

Overview



* There's one more data type called IR_IMG for only receiving IR images, but it's only useful for testing if the IR camera is working properly

3DSLiM API types and functions

- Basic types

- `OpUSBDev`

  The general class of Opus USB device.

- `_3DSLiM_API`

  The main class of 3DSLiM object.

- `return_status`

  A return type to indicate if the function execution is successful.

- `_3DSLiM_resolution`

  Indicate the desired output resolution.

- `_3DSLiM_data_type`

  Indicate the output data type. Currently only `PCD_XYZ`, `PCD_XYZ_RGB` (if there's an RGB camera attached) `IR_IMG` are supported.

- `_3DSLiM_xfer_mode`

  Indicate the transfer mode of point cloud data, including `ONE_SHOT`(capture) and `STREAMING`.

- `Exposure_mode`

  Indicate the exposure mode, including `MANUAL` and `AUTO`.

  ** Auto Exposure is experimental feature.

- `img_pcd_data`

  Contains three arrays of output IR image, point cloud data (in xyz format), 16 bit depth map and the mapped RGB data correspond to point cloud. If output data type is PCD_XYZ, mapped_rgb is NULL.

  1. img_data: IR image data pointer. Data structure: single channel unsigned char (mono image).
  2. depth_map: 16bits depth map,
     i. Mapping z value min_z_cutoff ~ max_z_cutoff (mm) to depth value 65535~0.
     ii. z value = (65535 - depth_value)/ 65535 * (max_z_cutoff - min_z_cutoff) + min_z_cutoff;
  3. pcd_data: point cloud xyz data pointer. Data structure: float x 3 channels (x,y,z).
  4. mapped_rgb: Mapped rgb data corresponding to xyz data. If output data type is PCD_XYZ, mapped_rgb is NULL.

- `_3DSLiM_parameter_type`

  Indicate the parameter types, it corresponds to `_3DSLiM_parameter`.

- `_3DSLiM_parameter`

  A structure for all parameters configuration.

1. **quality_factor:**

   . Setting a higher value to reduce the noise caused by white noise, but will make sensor more sensitive to the sensing range.

   . Integer, value range = (3, 9), default = 7;

   . 'NOT' supports execution time value change.

2. z_**cutoff** : Control the sensing range (mini-meter) .

   . Integer, value range = (0, 2000) mm,

   . min_z_cutoff default = 30, max_z_cutoff default = 1200;

   . use min_z_cutoff and max_z_cutoff to control the mapping range of depth-map p_depth: maps min_z_cutoff --> max_z_cutoff to 65535 -->0.

   . z value = (65535 - depth_value)/ 65535 * (max_z_cutoff - min_z_cutoff) + min_z_cutoff;

   . Supports execution time value change.

3. **f_point_cloud_smoothing:**

   . Flag for triggering point cloud smoothing

4. **point_cloud_smoothing_k_size:**

   . Odd integer, value = (3, 5, 7 , 9), default = 3.

   . Supports execution time value change.

5. **sensitivity**: Control 3D sensing sensitivity.

   . Setting a higher value makes the sensor more sensitive to the 3D information of object with low reflection.

   . Float, value range = (0, 1.0), default = 0.90.

   . Supports execution time value change.

6. **noise_filtering_factor:** remove noise points.

   . Float, value range = (0, 1.0), default = 0.15.

   . Value 0 means turn off noise filtering.

   . Supports execution time value change.

7. **f_motion_noise_filtering:** /*Experimental feature */

. Flag for triggering motion noise filtering

8. **threshold_motion_noise_factor:** /*Experimental feature */

   . Smaller value will make sensor more sensitive to the motion noise.

   . Float, value range = (0.05, 0.20), default = 0.08.

   . Supports execution time value change.

9. **f_resampling:** /*Experimental feature */

   . Flag for triggering resampling

10. **resampling_factor:** /*Experimental feature */

    . Setting a higher value will reduce the density of point cloud.

    . Float, value range = (0.001, 1.0), default = 0.0025.

    . Supports execution time value change.

■ `_3DSLiM_output_data`

A structure for output point cloud data.

1. total_num: number of points.
2. effective_num: effective number of point cloud.
3. data: type casting to img_pcd_data* when the data type is PCD_XYZ or PCD_XYZ_RGB, to unsigned char* when using IR_IMG.

- General functions

■ `initialize()`

Declaration:

```
return_status initialize(int dev_num);
```

Description:

Initialize the _3DSLiM_API object.

Parameter:

`dev_num`

The index of device connected to PC. Currently SDK only supports 1 device connected to 1 PC at the same time.

- `config_parameter()`

Declaration:

```
return_status config_parameter(_3DSLiM_parameter* parameter);
```

Description:

Configure all the parameters of point cloud data.

Parameter:

`parameter`

All the parameters of point cloud data.

- `config_parameter() – (other overloaded function)`

Declaration:

```
return_status config_parameter(_3DSLiM_parameter_type type, void*
value);
```

Description:

Configure the specific parameter of point cloud data.

Parameter:

`type`

Specific parameter of point cloud data.

`value`

The value to be set up.

- `start()`

Declaration:

```
return_status start(_3DSLiM_data_type type, _3DSLiM_xfer_mode mode,
int rgb_cam_dev_num = 0);
```

Description:

Start to get data.

Parameter:

`type`

Determine the output data type.

`mode`

Data transfer mode.

`rgb_cam_dev_num`

Indicate the device number of RGB camera. This value is sent to `cv::VideoCapture::open(*)`

- `stop()`

Declaration:

```
return_status stop(void);
```

Description:

Stop data transfer.

- `get_stream_data()`

Declaration:

```
return_status get_stream_data(_3DSLiM_output_data** data);
```

Description:

Get streaming data buffer.

Parameter:

`data`

Output data buffer.

■ release_stream_data()

Declaration:

```
return_status release_stream_data(_3DSLiM_output_data* data);
```

Description:

Release streaming data buffer to SDK.

Parameter:

`data`

Output data buffer.

■ init_one_shot_buffer()

Declaration:

```
return_status init_one_shot_buffer(_3DSLiM_output_data** data);
```

Description:

Initialize data buffer to capture a(n) point cloud / image.

Parameter:

`data`

Output data buffer.

■ get_one_shot_data()

Declaration:

```
return_status get_one_shot_data(_3DSLiM_output_data* data);
```

Description:

Capture a(n) point cloud / image.

Parameter:

`data`

Output data buffer.

- `delete_one_shot_buffer()`

Declaration:

```
return_status delete_one_shot_buffer(_3DSLiM_output_data* data);
```

Description:

Delete the data buffer for capturing the point cloud / image.

Parameter:

`data`

Output data buffer.

- `config_exposure()`

Declaration:

```
return_status config_exposure(Exposure_mode exp_mode, int value = 1);
```

Description:

Configure the exposure mode and exposure value (intensity target value). Notice that the value means exposure value in manual exposure mode while it means intensity target value in auto exposure mode.

Parameter:

`exp_mode`

Determine the exposure mode.

`value`

Desired exposure value in manual exposure mode, ranges from 1 to 3000.

Desired target intensity value in auto exposure mode, ranges from 0 to 255.

- `config_resolution()`

Declaration:

```
return_status config_resolution(_3DSLiM_resolution resolution);
```

Description:

Configure the desired resolution of output data.

Parameter:

`resolution`

Desired resolution.

- `get_resolution()`

Declaration:

```
_3DSLiM_resolution get_resolution(void);
```

Description:

Get the current resolution.

- `save_ply_file()`

Declaration:

```
return_status save_ply_file(_3DSLiM_output_data* data, string
filename);
```

Description:

Save the output point cloud data to .ply file.

Parameter:

`data`

Output point cloud data.

`filename`

The file name of the .ply file.

- `read_lib_version()`

Declaration:

```
string read_lib_version(void);
```

Description:

Return the library version.