LxCameraSDK-Python User Manual

This manual is intended for developers. Users should have some experience with Python development and the corresponding platform development experience.

1. Installation

```
pip install lx_camera_py-1.0-py3-none-any.whl
```

2. Usage

- Import

  `/path/to/your/so_or_dll` indicates the library file for LxCameraApi. On Windows, it's a `.dll` file, and on Linux, it's a `.so` file.

  ```python
  from LxCameraSDK import *
  camera = LxCamera('/path/to/your/so_or_dll')
  ```

  The first parameter returned by all methods is of type `LX_STATE`, where `LX_SUCCESS` indicates success. Other error codes can be retrieved using `camera.DcGetErrorString(state)`, where `state` is the return value of type `LX_STATE`.

- Get API Version

  ```python
  api_version = camera.DcGetApiVersion()
  ```

- Log Configuration

  ```python
  state = camera.DcSetInfoOutput(2, False, 'log/', 0)
  ```

  - The first parameter is the log level: 0 for all information, 1 for warnings, 2 for errors.
  - The second parameter indicates whether to output to the screen.
  - The third parameter is the log storage path.

- Get Camera List

  ```python
  state, dev_list, dev_num = camera.DcGetDeviceList()
  ```

  If successful, the camera list can be obtained through `dev_list`, and `dev_num` is the number of cameras found.

- Open Device

  The open modes are defined in `LxCameraSDK.lx_camera_define.LX_OPEN_MODE`, and there are four modes:

  - `OPEN_BY_INDEX`: Open by the index of the camera list.
  - `OPEN_BY_IP`: Open by IP address.
  - `OPEN_BY_SN`: Open by device serial number.
  - `OPEN_BY_ID`: Open by device ID.

```
open_mode = LX_OPEN_MODE.OPEN_BY_IP
if open_mode == LX_OPEN_MODE.OPEN_BY_IP:
  open_param = "192.168.100.120"
elif open_mode == LX_OPEN_MODE.OPEN_BY_ID:
  open_param = "F131411400000000"
elif open_mode == LX_OPEN_MODE.OPEN_BY_SN:
  open_param = "519889C9A2A6468E"
elif open_mode == LX_OPEN_MODE.OPEN_BY_INDEX:
  open_param = "0"
else:
  raise NotImplementedError(f"Camera open mode {open_mode} not implemented")

# Open Device
state, handle, device_info = camera.DcOpenDevice(open_mode, open_param)
```

- The first parameter is the open mode.
- The second parameter is the corresponding value (string type).
- The return value `handle` is used for all subsequent camera-related operations, and `device_info` is of type `LxCameraSDK.lx_camera_define.LxDeviceInfo`.

- Close Device

```
state = camera.DcCloseDevice(handle)
```

- Start Stream

```
state = camera.DcStartStream(handle)
```

- Stop Stream

```
state = camera.DcStopStream(handle)
```

- Set Camera IP

```
state = camera.DcSetCameraIp(handle, "192.168.1.100", "255.255.0.0",
"192.168.1.1")
```

  - The first parameter is the device handle.
  - The second parameter is the camera IP.
  - The third parameter is the subnet mask, default is `255.255.0.0`.
  - The fourth parameter is the gateway, default is 1 for the last position of the IP.

  After setting, the camera will automatically restart.

- Set Integer Value

```
state = camera.DcSetIntValue(handle, LX_CAMERA_FEATURE.LX_INT_GAIN, 10)
```

  Refer to the integer part of `LxCameraSDK.lx_camera_define.LX_CAMERA_FEATURE` for the enumerable values that can be set.

- Get Integer Value

```
state, value = camera.DcGetIntValue(handle, LX_CAMERA_FEATURE.LX_INT_GAIN)
```

The output `value` is of type `LxCameraSDK.lx_camera_define.LxIntValueInfo`, refer to the definition of this class for specific meanings.

- Set Float Value

  ```
  # For the example below, LX_FILTER_MODE should be set to FILTER_SIMPLE,
  otherwise a function call error will occur.
  # Set first, then get
  state = camera.DcSetIntValue(handle, LX_CAMERA_FEATURE.LX_INT_FILTER_MODE,
  LX_FILTER_MODE.FILTER_SIMPLE)
  state = camera.DcSetFloatValue(handle,
  LX_CAMERA_FEATURE.LX_FLOAT_FILTER_LEVEL, 0.1)
  ```

  Refer to the float part of `LX_CAMERA_FEATURE` for the enumerable values that can be set.

- Get Float Value

  ```
  state, value = camera.DcGetFloatValue(handle,
  LX_CAMERA_FEATURE.LX_FLOAT_FILTER_LEVEL)
  ```

  The return value `value` is of type `LxCameraSDK.lx_camera_define.LxFloatValueInfo`.

- Set String Value

  ```
  state = camera.DcSetStringValue(handle,
  LX_CAMERA_FEATURE.LX_STRING_ALGORITHM_PARAMS, string)
  ```

  Refer to the string part of `LxCameraSDK.lx_camera_define.LX_CAMERA_FEATURE` for the enumerable values that can be set.

- Get String Value

  ```
  state, value = camera.DcGetStringValue(handle,
  LX_CAMERA_FEATURE.LX_STRING_ALGORITHM_PARAMS)
  ```

  The return value `value` is of Python string type.

- Set Boolean Value

  ```
  state = camera.DcSetBoolValue(handle,
  LX_CAMERA_FEATURE.LX_BOOL_ENABLE_2D_STREAM, True)
  ```

  Refer to the boolean part of `LxCameraSDK.lx_camera_define.LX_CAMERA_FEATURE` for the enumerable values that can be set.

- Get Boolean Value

  ```
  state, value = camera.DcGetBoolValue(handle,
  LX_CAMERA_FEATURE.LX_BOOL_ENABLE_3D_UNDISTORT)
  ```

  The return value `value` is of Python bool type.

- Get 3D Camera Transformation Matrix

```
state, trans_matrix = camera.get3DTransMatrix(handle)
```

`trans_matrix` is of type `numpy.ndarray`, with `shape` `(4,3)`. The first three columns are the rotation matrix, and the fourth column is the translation vector.

- Get 2D Camera Intrinsic Parameters

```
state, intrinsic_2d, distort_2d = camera.get2DIntricParam(handle)
```

`intrinsic_2d` is defined as `[fx, fy, cx, cy]`, and `distort_2d` are the distortion parameters.

- Get 3D Camera Intrinsic Parameters

```
state, intrinsic_3d, distort_3d = camera.get3DIntricParam(handle)
```

`intrinsic_3d` is defined as `[fx, fy, cx, cy]`, and `distort_3d` are the distortion parameters.

- Capture Frame

```
state, data_ptr = camera.getFrame(handle)
```

`data_ptr` is a pointer of type `LxCameraSDK.lx_camera_define.FrameInfo`.

- Get RGB Image

```
# First, start the 2D stream. This command only needs to be executed once. If it is not started or not started successfully, then data_ptr.rgb_data.frame_data will be 0.
camera.DcSetBoolValue(handle, LX_CAMERA_FEATURE.LX_BOOL_ENABLE_2D_STREAM, True)
state, rgb_image = camera.getRGBImage(data_ptr)
```

- Get Depth Image

```
# First, start the 3D depth stream. This command only needs to be executed once. If it is not started or not started successfully, then data_ptr.depth_data.frame_data will be 0.
camera.DcSetBoolValue(handle, LX_CAMERA_FEATURE.LX_BOOL_ENABLE_3D_DEPTH_STREAM, True)
state, depth_image = camera.getDepthImage(data_ptr)
```

- Get Intensity Image

```
# First, start the 3D intensity stream. This command only needs to be executed once. If it is not started or not started successfully, then data_ptr.amp_data.frame_data will be 0.
camera.DcSetBoolValue(handle, LX_CAMERA_FEATURE.LX_BOOL_ENABLE_3D_AMP_STREAM, True)
state, amp_image = camera.getAmpImage(data_ptr)
```

- Get Point Cloud

```python
# You need to set the command to get new data before getting the point cloud.
state = camera.DcSetCmd(handle, LX_CAMERA_FEATURE.LX_CMD_GET_NEW_FRAME)
state, points = camera.getPointCloud(handle)
```

`points` is of type `np.ndarray`, with `shape=(depth_width, depth_height, 3)`.

- Save Point Cloud

  ```python
  camera.DcSaveXYZ(handle, "./xxx.pcd")
  ```

  The second parameter is the path

to save the point cloud. Supported point cloud formats are: txt, pcd, ply.

- Set Command

  ```python
  state = camera.DcSetCmd(handle, LX_CAMERA_FEATURE.LX_CMD_WHITE_BALANCE)
  ```

  Refer to the CMD part of `LX_CAMERA_FEATURE` for the commands that can be set.

- Get Algorithm Results

  Before getting algorithm results, the corresponding algorithm mode needs to be set, defined in `LxCameraSDK.lx_camera_define.LX_ALGORITHM_MODE`.

  ```python
  state = camera.DcSetIntValue(handle, LX_CAMERA_FEATURE.LX_INT_ALGORITHM_MODE,
  LX_ALGORITHM_MODE.MODE_PALLET_LOCATE)
  state, value = camera.getAlgorithmStatus(handle)
  ```

  The four algorithm result data types are as follows:

  - MODE_AVOID_OBSTACLE: `LxCameraSDK.lx_camera_application.LxAvoidanceOutput`
  - MODE_PALLET_LOCATE: `LxCameraSDK.lx_camera_application.LxPalletPose`
  - MODE_VISION_LOCATION: `LxCameraSDK.lx_camera_application.LxLocation`
  - MODE_AVOID_OBSTACLE2: `LxCameraSDK.lx_camera_application.LxAvoidanceOutputN`

- For other applications, refer to `LxCameraSDK.Sample`