

LxCameraSDK-Python使用手册

此手册面向开发者，使用者须有一定的Python开发经验，以及对应平台开发经验

1. 安装

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

2. 使用方法

- 导入

`/path/to/your/so_or_dll` 表示LxCameraApi的库文件，windows下为.dll文件，linux下为.so文件。

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

所有方法返回的第一个参数类型均为 `LX_STATE`，`LX_SUCCESS` 表示成功，其余错误码可通过 `camera.DcGetErrorString(state)` 获取具体错误信息，`state` 是 `LX_STATE` 类型的返回值。

- 获取API版本

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

- 日志配置

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

- 第一个参数为输出日志等级，0 所有信息，1 警告信息，2 错误信息
- 第二个参数表示是否输出至屏幕
- 第三个参数表示日志存放路径

- 获取相机列表

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

若成功，可通过 `dev_list` 获取搜索到的相机列表，`dev_num` 是搜索到的相机数量。

- 打开设备

打开模式定义于 `LxCameraSDK.lx_camera_define.LX_OPEN_MODE`，有以下四种打开模式：

- `OPEN_BY_INDEX`：根据搜索到的相机列表索引打开
- `OPEN_BY_IP`：通过IP打开
- `OPEN_BY_SN`：通过设备序列号打开
- `OPEN_BY_ID`：通过设备ID打开

```
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")

# 打开设备
state, handle, device_info = camera.DcOpenDevice(open_mode, open_param)

```

- 第一个参数为打开模式
- 第二个参数为对应的值（字符串类型）
- 返回值 `handle` 用于后续所有与相机有关的操作，`device_info` 是 `LxCameraSDK.lx_camera_define.LxDeviceInfo` 类型。

- 关闭设备

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

- 启流

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

- 停流

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

- 设置相机IP

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

- 第一个参数为设备 `handle`
- 第二个参数为相机IP
- 第三个参数为子网掩码，默认为 `255.255.0.0`
- 第四个参数为网管，默认为IP最后一位置1

设置成功后相机会自动重启。

- 设置整型值

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

参考 `LxCameraSDK.lx_camera_define.LX_CAMERA_FEATURE` 整型部分获取可设置的枚举值。

- 获取整型值

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

输出结果 `value` 的类型为 `LxCameraSDK.lx_camera_define.LxIntValueInfo`，参考该类的定义获取具体含义。

- 设置浮点型值

```
# 对于下方示例，需要将LX_FILTER_MODE设为FILTER_SIMPLE，否则会报函数调用错误
# 先设置，再获取
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)
```

参考 `LX_CAMERA_FEATURE` 浮点型部分获取可设置的枚举值。

- 获取浮点型值

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

返回值 `value` 的类型为 `LxCameraSDK.lx_camera_define.LxFloatValueInfo`

- 设置字符型值

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

参考 `LxCameraSDK.lx_camera_define.LX_CAMERA_FEATURE` 字符型部分获取可设置的枚举值。

- 获取字符型值

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

返回值 `value` 的类型为python字符串类型。

- 设置布尔型值

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

参考 `LxCameraSDK.lx_camera_define.LX_CAMERA_FEATURE` 布尔型部分获取可设置的枚举值。

- 获取布尔型值

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

返回值 `value` 的类型为python bool类型。

- 获取3D相机变换矩阵

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

`trans_matrix` 是 `numpy.ndarray` 类型，`shape` 为 `(4,3)`，前三列为旋转矩阵，第四列为平移向量。

- 获取2D相机内参

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

`intrinsic_2d` 定义为 `[fx,fy,cx,cy]` , `distort_2d` 为畸变参数。

- 获取3D相机内参

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

`intrinsic_3d` 定义为 `[fx,fy,cx,cy]` , `distort_3d` 为畸变参数。

- 取帧

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

`data_ptr` 是 `LxCameraSDK.lx_camera_define.FrameInfo` 类型的指针。

- 获取RGB图像

```
# 先后2D流, 该指令只需操作一次, 如果未启动或启动不成功, 则data_ptr.rgb_data.frame_data  
# 为0  
camera.DcSetBoolValue(handle, LX_CAMERA_FEATURE.LX_BOOL_ENABLE_2D_STREAM,  
True)  
state, rgb_image = camera.getRGBImage(data_ptr)
```

- 获取深度图

```
# 先后3D深度流, 该指令只需操作一次, 如果未启动或启动不成功, 则  
# data_ptr.depth_data.frame_data为0  
camera.DcSetBoolValue(handle,  
LX_CAMERA_FEATURE.LX_BOOL_ENABLE_3D_DEPTH_STREAM, True)  
state, depth_image = camera.getDepthImage(data_ptr)
```

- 获取强度图

```
# 先后3D强度流, 该指令只需操作一次, 如果未启动或启动不成功, 则  
# data_ptr.amp_data.frame_data为0  
camera.DcSetBoolValue(handle, LX_CAMERA_FEATURE.LX_BOOL_ENABLE_3D_AMP_STREAM,  
True)  
state, amp_image = camera.getAmpImage(data_ptr)
```

- 获取点云

```
# 需要先设置指令获取新的数据, 才能拿到点云  
state = camera.DcSetCmd(handle, LX_CAMERA_FEATURE.LX_CMD_GET_NEW_FRAME)  
state, points = camera.getPointCloud(handle)
```

`points` 是 `np.ndarray` 类型, `shape=(depth_width, depth_height, 3)`

- 保存点云

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

第二个参数是存储点云的路径, 支持保存的点云格式有: txt、pcd、ply

- 设置指令

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

可设置的指令参考 `LX_CAMERA_FEATURE` CMD 部分。

- 获取算法结果

获取算法结果前，需要设置对应的算法模式，定义于

```
LxCameraSDK.lx_camera_define.LX_ALGORITHM_MODE
```

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

四种算法结果数据类型如下：

- 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`

- 其他应用参考 `LxCameraSDK.Sample`