https://www.ximea.com/support/wiki/apis/Python

xiAPI Python

Python のアプリケーションは、xiAPI Python インターフェースを使用して XIMEA カメラにアクセスできます。

この API は、XIMEA カメラの機能と性能をフルに活用できるインターフェースを作成します。

この Python インターフェースは、OpenCV や Pillow などのさまざまな Python パッケージをサポートしています。

ドキュメント

xiAPI Python マニュアル

サポートされているプラットフォーム

・Windows
・Linux
・MacOS

インストール手順

・Windows のインストール
・Linux のインストール
・MacOS のインストール

サンプル コード:

Ximea Python パッケージは、次の方法でスクリプトで使用できます。

```
from ximea import xiapi

#create instance for first connected camera
cam = xiapi.Camera()

#start communication
#to open specific device, use:
#cam.open_device_by_SN('41305651')
#(open by serial number)
print('Opening first camera...')
cam.open_device()
```

```
#settings
cam.set_exposure(10000)
print('Exposure was set to %i us' %cam.get_exposure())

#create instance of Image to store image data and metadata
img = xiapi.Image()

#start data acquisition
print('Starting data acquisition...')
cam.start_acquisition()

for i in range(10):
    #get data and pass them from camera to img
    cam.get_image(img)

    #get raw data from camera
    #for Python2.x function returns string
    #for Python3.x function returns bytes
    data_raw = img.get_image_data_raw()

    #transform data to list
    data = list(data_raw)

    #print image data and metadata
    print('Image number: ' + str(i))
    print('Image width (pixels):  ' + str(img.width))
    print('Image height (pixels): ' + str(img.height))
    print('First 10 pixels: ' + str(data[:10]))
    print('¥n')

#stop data acquisition
print('Stopping acquisition...')
cam.stop_acquisition()

#stop communication
cam.close_device()

print('Done.')
```

XIMEA Python 構文

xiAPI パラメータの定数文字列定義から生成される XIMEA Python 関数を使用して、さまざまな xiAPI パラメータを設定または読み取ることができます。

列挙子パラメータ

プレフィックス *get_* と *set_* は、次のようにして定数文字列定義に追加されます。

パラメータ XI_PRM_ACQ_TIMING_MODE の場合、acq_timing_mode は定数文字列であるため、Python XIMEA API には関数 get_acq_timing_mode() と set_acq_timing_mode() が含まれます。

```
cam.set_acq_timing_mode('XI_ACQ_TIMING_MODE_FRAME_RATE')
mode_used = cam.get_acq_timing_mode()
```

返される値は文字列なので、返された値をチェックして別の文字列と比較することができます。

```
if mode_used == 'XI_ACQ_TIMING_MODE_FRAME_RATE':
    print('Mode is XI_ACQ_TIMING_MODE_FRAME_RATE')
else:
    print('Mode is not XI_ACQ_TIMING_MODE_FRAME_RATE')
```

列挙子の入力は特定の文字列である必要があります。これらの文字列が表す整数値の形式で入力すると、エラーが発生します。

```
camera.set_gpo_selector('XI_GPO_PORT1') #will work correctly
camera.set_gpo_selector(1) #will result in error
```

非ブールパラメータ

プレフィックス *get_* と *set_* は、列挙子パラメータの場合と同じ方法で追加されます。

たとえば、パラメータ XI_PRM_EXPOSURE の場合、「exposure」は定数文字列であるため、Python XIMEA API には関数 get_exposure() と set_exposure() が含まれます。

```
cam.set_exposure(10000)
print('Current exposure is %s us.' %cam.get_exposure())
```

出力は次のようになります：

```
Current exposure is 10000 us.
```

パラメータ XI_PRM_FRAMERATE を 10 に設定する場合、コマンドは次のようになります:

```
cam.set_acq_timing_mode('XI_ACQ_TIMING_MODE_FRAME_RATE')
cam.set_framerate(10)
```

XI_ACQ_TIMING_MODE_FRAME_RATE の値は MQ カメラと MD カメラでのみ機能することに注意してください。固定フレームレートの設定に関する詳細は、xiAPI マニュアルおよびナレッジベースの記事「フレームレート制御」に記載されています。
API パラメータ修飾子を取得するためのコマンドは次のようになります：

```
print('The maximal width of this camera is %i.' %cam.get_width_maximum())
print('The minimal width of this camera is %i.' %cam.get_width_minimum())
print('The increment of the width of this camera is %i.' %cam.get_width_increment())
```

例えばMQ013MG-ONカメラの出力は次のようになります：

```
The maximal width of this camera is 1280.
The minimal width of this camera is 16.
The increment of the width of this camera is 16.
```

ブール型パラメータ

ブール型パラメータにも同じロジックが使用され、プレフィックス *is_* 、*enable_* 、*disable_* は次のように使用されます：
パラメータ XI_PRM_HORIZONTAL_FLIP の場合、「horizontal_flip」は定数文字列であるため、Python XIMEA API には関数 is_horizontal_flip()、enable_horizontal_flip()、disable_horizontal_flip() が含まれます：

```
print('Is horizontal flip enabled?')
print(cam.is_horizontal_flip())

print('Enabling horizontal flip...')
cam.enable_horizontal_flip()
print(cam.is_horizontal_flip())

print('Disabling horizontal flip...')
cam.disable_horizontal_flip()
print(cam.is_horizontal_flip())
```

そして出力は次のようになります：

```
Is horizontal flip enabled?
False.
Enabling horizontal flip...
True.
Disabling horizontal flip...
False.
```

共通インターフェース

パラメータの設定または取得には、*get_param(arg)* 関数と *set_param(arg)* 関数を使用できます。
arg は、特定のパラメータの定数文字列を次のように使用します:

```
cam.set_param("exposure",10000)
// is the same as
cam.set_exposure(10000)
```

```
cam.get_param("width:max")
// is the same as
cam.get_width_maximum()
```

どちらのアプローチも有効です。

サンプル アプリケーション

xiApiPython examples/ フォルダーには、いくつかのアプリケーションがあります。

ここでは、xiApiPython、OpenCV、または Pillow パッケージのさまざまなサンプルを見つけることができます。

注
Python v3. は、pillow および variety フォルダーのサンプルのみをサポートします。

https://www.ximea.com/support/wiki/apis/Python

xiAPI Python

Applications in Python can access XIMEA cameras using xiAPI Python interface.

This API creates an interface with which you can fully utilize the features and capabilities of XIMEA cameras.
This Python interface supports various Python packages like OpenCV or Pillow.

Documentation

xiAPI Python Manual

Supported platforms

- Windows
- Linux
- MacOS

Installation instructions

- Installation for Windows
- Installation for Linux
- Installation for MacOS

Sample code:

The Ximea Python package may be used in a script in the following way:

```
from ximea import xiapi

#create instance for first connected camera
cam = xiapi.Camera()

#start communication
#to open specific device, use:
#cam.open_device_by_SN('41305651')
#(open by serial number)
print('Opening first camera...')
cam.open_device()
```

```
#settings
cam.set_exposure(10000)
print('Exposure was set to %i us' %cam.get_exposure())

#create instance of Image to store image data and metadata
img = xiapi.Image()

#start data acquisition
print('Starting data acquisition...')
cam.start_acquisition()

for i in range(10):
    #get data and pass them from camera to img
    cam.get_image(img)

    #get raw data from camera
    #for Python2.x function returns string
    #for Python3.x function returns bytes
    data_raw = img.get_image_data_raw()

    #transform data to list
    data = list(data_raw)

    #print image data and metadata
    print('Image number: ' + str(i))
    print('Image width (pixels):  ' + str(img.width))
    print('Image height (pixels): ' + str(img.height))
    print('First 10 pixels: ' + str(data[:10]))
    print('¥n')

#stop data acquisition
print('Stopping acquisition...')
cam.stop_acquisition()

#stop communication
cam.close_device()

print('Done.')
```

XIMEA Python syntax

The various xiAPI parameters can be set or readout using XIMEA Python functions which are generated from the constant string definitions of the xiAPI parameters.
Enumerator parameters
Prefixes *get* _ and *set* _ are added to constant string definition the following way:
For parameter **XI_PRM_ACQ_TIMING_MODE,** acq_timing_mode is the constant string, therefor Python XIMEA API contains functions get_acq_timing_mode() and set_acq_timing_mode():

```
cam.set_acq_timing_mode('XI_ACQ_TIMING_MODE_FRAME_RATE')
mode_used = cam.get_acq_timing_mode()
```

The returned value is string, so checking the retured value could be compared with another string:

```
if mode_used == 'XI_ACQ_TIMING_MODE_FRAME_RATE':
    print('Mode is XI_ACQ_TIMING_MODE_FRAME_RATE')
else:
    print('Mode is not XI_ACQ_TIMING_MODE_FRAME_RATE')
```

Enumerator inputs must be specific strings. Input in form of integer value that these strings represent will result in errors:

```
camera.set_gpo_selector('XI_GPO_PORT1') #will work correctly
camera.set_gpo_selector(1) #will result in error
```

Nonboolean parameters

Prefixes *get* _ and *set* _ are added the same way as for enumerator parameters:
E.g. for parameter **XI_PRM_EXPOSURE**, "exposure" is the constant string, therefore Python XIMEA API contains functions get_exposure() and set_exposure():

```
cam.set_exposure(10000)
print('Current exposure is %s us.' %cam.get_exposure())
```

and the output will be:

```
Current exposure is 10000 us.
```

For setting the parameter **XI_PRM_FRAMERATE** to 10, the commands will be:

```
cam.set_acq_timing_mode('XI_ACQ_TIMING_MODE_FRAME_RATE')
cam.set_framerate(10)
```

Note, that the value *XI_ACQ_TIMING_MODE_FRAME_RATE* works only with MQ and MD cameras. More information regarding setting the fixed framerate can be found in xiAPI Manual and in our knowledge base article Frame Rate Control.
For getting **API parameter modifiers** commands will be:

```
print('The maximal width of this camera is %i.' %cam.get_width_maximum())
print('The minimal width of this camera is %i.' %cam.get_width_minimum())
print('The increment of the width of this camera is %i.' %cam.get_width_increment())
```

and the output for e.g. MQ013MG-ON camera will be:

```
The maximal width of this camera is 1280.
The minimal width of this camera is 16.
The increment of the width of this camera is 16.
```

Boolean parameters

Same logic is used for boolean parameters, where prefixes *is* _, *enable* _ and *disable* _ are used the following way:
For parameter **XI_PRM_HORIZONTAL_FLIP**, "horizontal_flip" is the constant string, therefor Python XIMEA API contains functions is_horizontal_flip(), enable_horizontal_flip() and disable_horizontal_flip():

```
print('Is horizontal flip enabled?')
print(cam.is_horizontal_flip())

print('Enabling horizontal flip...')
cam.enable_horizontal_flip()
print(cam.is_horizontal_flip())

print('Disabling horizontal flip...')
cam.disable_horizontal_flip()
print(cam.is_horizontal_flip())
```

and the output will be:

```
Is horizontal flip enabled?
False.
Enabling horizontal flip...
True.
Disabling horizontal flip...
False.
```

Common interface

For setting or getting parameters can be used *get_param(arg)* and *set_param(arg)* functions, where as **arg** is used the constant string of specific parameter the following way:

```
cam.set_param("exposure",10000)
// is the same as
cam.set_exposure(10000)
```

```
cam.get_param("width:max")
// is the same as
cam.get_width_maximum()
```

Both approaches are valid.

Example applications

There are several applications, that may be found in the xiApiPython examples/ folder.

Here you may find various examples for xiApiPython, OpenCV, or Pillow packages.

Note
The Python v3. supports examples only from *pillow* and *various* folder.