OWLIFT Type-H
Python SDK
開発者ガイド

2021/02/19

インフィニテグラ株式会社

目次

1 概要 3
1.1 はじめに 3
1.2 動作環境 3
1.3 SDK の内容 3
1.4 表記 3
2 セットアップ 4
2.1 インストール 4
2.2 アンインストール 4
3 ツール 5
3.1 発熱検知アプリケーション BodyTemp 5
3.2 汎用計測ツール live-temp.py 5
3.3 録画・再生ツール write.py / read.py 6
3.3.1 OWLIFT における録画 6
3.3.2 録画方法 6
3.3.3 再生方法 6
4 サンプルコード 7
4.1 最大値表示 max.py 7
4.2 画像表示 preview.py 8
4.3 イベント処理 detect.py 9
4.4 距離取得 dist.py 10
4.5 LED 操作 led-test.py 11

1 概要

1.1 はじめに

本書では体表温度計測システムの開発キット「OWLIFT Type-H Python SDK」についての使用方法を説明します。

1.2 動作環境

- Windows
  - Windows 10 32bit/64bit
  - Python 3.6/3.7 32bit/64bit
- Raspberry Pi
  - Raspberry Pi OS Buster 32bit
  - Python 3.7

1.3 SDK の内容

SDK のアーカイブファイルの内容を示します。

項目		ファイル・ディレクトリ
OWLIFT Type-H Python モジュール		owlifttypeh-*.whl
API リファレンス		doc/python-ja/index.html
ソースコード	発熱検知アプリケーション	samples/BodyTemp
	汎用計測ツール	samples/live-temp.py
	録画ツール	samples/write.py
	再生ツール	samples/read.py
	サンプルコード：最大値表示	samples/max.py
	サンプルコード：画像表示	samples/preview.py
	サンプルコード：イベント処理	samples/detect.py
	サンプルコード：距離取得	samples/dist.py
	サンプルコード：LED 操作	samples/led-test.py

1.4 表記

文章中で以下の表記を用います。

- %OWH_PY_SDK_DIR% もしくは $OWH_PY_SDK_DIR
  OWLIFT Type-H Python SDK をインストールしたフォルダの絶対パスを表します。

また、Python は仮想環境を前提としています。特に Raspberry Pi で仮想環境を使用しない場合はコマンド pip、python はそれぞれ pip3、python3 に置き換えてください。

2 セットアップ

2.1 インストール

・ 開発環境のプラットフォームに合った OWLIFT Type-H Python モジュールの whl ファイルを pip コマンドでインストールします。Raspberry Pi の場合は依存ライブラリもインストールします。

(#.#.# はバージョン番号)

```
Windows 32bit の場合
> pip install owlifttypeh-#.#.#-py3-none-win32.whl
Windows 64bit の場合
> pip install owlifttypeh-#.#.#-py3-none-win_amd64.whl
Raspberry Pi の場合
> sudo apt install libatlas3-base
> pip install owlifttypeh-#.#.#-py3-none-linux_armv7l.whl
```

・ ツールとサンプルコードの依存モジュールをインストールします。Raspberry Pi の場合は依存ライブラリもインストールします。

```
Windows の場合
> %OWH_PY_SDK_DIR%¥samples¥setup.bat
Raspberry Pi の場合
> sudo apt install libsdl2-ttf-2.0-0 libsdl2-image-2.0-0 libsdl2-mixer-2.0-0
> $OWH_PY_SDK_DIR/samples/setup.sh
```

2.2 アンインストール

・ OWLIFT Type-H Python モジュールを削除します。

```
> pip uninstall owlifttypeh
```

・ setup.bat もしくは setup.sh に記述されたモジュールを pip uninstall で削除します。

・ Raspberry Pi の場合は apt install でインストールしたライブラリを apt remove もしくは apt purge で削除します。

3 ツール

3.1 発熱検知アプリケーション BodyTemp

OWLIFT Type-H の標準アプリケーションです。

%OWH_PY_SDK_DIR%¥samples¥BodyTemp に移動して、以下のコマンドで実行します。

```
> python main.py
```

使用方法については「OWLIFT Type-H ユーザーズガイド」を参照してください。

3.2 汎用計測ツール live-temp.py

任意の範囲の平均・最大温度を表示するツールです。

以下のコマンドで実行します。

```
> python live-temp.py
```

使用方法については「OWLIFT Type-H ユーザーズガイド」を参照してください。

3.3 録画・再生ツール write.py / read.py

3.3.1 OWLIFT における録画

OWLIFT Type-H Python モジュールにはカメラの出力を全てファイルに記録して再現する機能があります。その機能を利用したツールが write.py と read.py です。

録画ファイルは拡張子が.owi の「OWI ファイル」と呼びます。全てのフレームの温度データ、距離センサデータが可逆的な状態で保存されます。一般的な非可逆圧縮での動画と区別して OWLIFT では「Raw 録画」と呼ぶことがあります。

3.3.2 録画方法

以下のコマンドで録画を開始します。

```
> python write.py -d [保存先ディレクトリのパス]
```

- 終了するときは画面上で ESC キーを押下します。
- 保存先ディレクトリには cam0.owi・cam1.owi ファイルが生成されます。
- 良く使うオプション
  - -b：背景を表示する
  - -F：人体検知を無効にする

3.3.3 再生方法

以下のコマンドで再生します。

```
> python read.py -d [保存先ディレクトリのパス]
```

- 終了するときは画面上で ESC キーを押下します。
- 良く使うオプション
  - -i：フレーム間のインターバル(秒)。0.01 などの小さい値を指定すると再生速度が上がります。
  - -s：スキップする秒数。目的の箇所まで画像表示を省略します。
  - -m：manu_corr（マニュアル補正値: 体表温度と体温の差）を指定します。

4 サンプルコード

4.1 最大値表示 max.py

温度テーブルを取得して最大値を表示します。

```
import numpy as np
import time
from owlifttypeh import OwhDevice

ow = OwhDevice.connect()                                    # 1
ow.set_options({ "image_tab": False,                        # 2
        "temp_tab": True })                                 # 3
ow.capture_start()                                          # 4
fc0 = 0

while ow.alive:
    fc = ow.frame_counter
    if fc0 and fc == fc0:                                   # 5
        time.sleep(0.01)
        continue
    fc0 = fc
    img, meta = ow.get_frame()                              # 6
    if meta.temp_tab is None:
        continue
    max_temp = np.max(meta.temp_tab)                        # 7
    print("{:.2f}".format(max_temp - 273.15))
```

1. カメラに接続します。
2. オプションの image_tab:False で画像出力を OFF。デフォルトでは ON。画像を使用しないとき、処理の負荷を低くする効果があります。
3. オプションの temp_tab:True で温度テーブル出力を ON。デフォルトでは OFF。
4. キャプチャ開始。
5. 新しいフレームが来るまでループ。
6. フレームを取得。
7. 温度テーブルを取得できたら最大値を取得、表示。meta.temp_tab は 120x120 の float 要素が格納された numpy.array です。値の単位がケルビン（K）なので摂氏（℃）に変換するため 273.15 を引きます。

4.2 画像表示 preview.py

画像を取得して表示します。

```
import cv2
from owlifttypeh import OwhDevice

ow = OwhDevice.connect()
wx, wy = ow.frame_size
ow.set_options({"hide_bg": False})                    # 1

winName = 'Thermography'
cv2.namedWindow(winName, cv2.WINDOW_NORMAL)           # 2
cv2.resizeWindow(winName, wx * 4, wy * 4)
ow.capture_start()
fc0 = 0

while ow.alive:
    c = cv2.waitKey(1)                                # 3
    if c == 27:
        break
    fc = ow.frame_counter
    if fc == fc0:
        continue

    fc0 = fc
    img, meta = ow.get_frame()

    cv2.imshow(winName, ow.image_to_array(img))       # 4
```

1. オプションの hide_bg:False で背景を表示します。背景とは動体検知により動体ではないと判断された部分です。
2. ウィンドウの設定をします。詳しくは OpenCV のリファレンスを参照してください。
3. ウィンドウ上で ESC キーが押下されたら終了します。
4. 画像を表示します。img は 32 ビット RGB 画像データの bytes オブジェクトです。cv2.imshow で表示できるよう 120x120x4（uint8）の numpy.array に変換して渡します。

4.3 イベント処理 detect.py

イベント処理の仕方です。

```
～省略～
while ow.alive:
～省略～
    fc0 = fc
    img, meta = ow.get_frame()
～省略～
    eid = meta.event_id                                                    # 1
    aimg = ow.image_to_array(img)

    if eid != eid0:
        if (meta.event_type & OwhMeta.EV_CORRECT) != 0:                    # 2
            print("¥ncorrect_cnt={:d} correct_error={:d}" ¥
                    .format(meta.correct_count, meta.correct_error))
        if (meta.event_type & OwhMeta.EV_BODY_TEMP) != 0:                  # 3
            msg = "{:.2f}".format(meta.body_temp)
            msg2 = "({:d},{:d})".format(meta.body_temp_x, meta.body_temp_y)
        if (meta.event_type & OwhMeta.EV_LOST) != 0:                       # 4
            msg = None
        if (meta.event_type & OwhMeta.EV_DIST_VALID) != 0:                 # 5
            msg2 = "VALID"
        if (meta.event_type & OwhMeta.EV_DIST_INVALID) != 0:               # 6
            msg2 = "INVALID"
        eid0 = meta.event_id
～省略～
```

1. イベント ID を取得します。イベント ID はイベントが発生するたびにシーケンシャルに増加する値です。
2. EV_CORRECT フラグが立っていれば、残りの計測回数・エラーを出力します。コマンドラインオプションで –correct-mode を指定するとマニュアル補正計測が開始されます。その状態で体表温度が計測されると EV_CORRECT フラグが立ちます。
3. EV_BODY_TEMP フラグが立っていれば計測位置を出力します。マニュアル補正計測ではないとき体表温度が計測されると EV_BODY_TEMP フラグが立ちます。
4. EV_LOST フラグが立っていれば表示メッセージを削除します。動体検知で一度追跡された動体を見失ったときに EV_LOST が立ちます。
5. EV_DIST_VALID フラグが立っていればメッセージを表示します。距離センサによる距離の出力が計測可能範囲内になったとき EV_DIST_VALID フラグが立ちます。
6. EV_DIST_INVALID フラグが立っていればメッセージを表示します。距離センサによる距離の出力が計測可能範囲内外なったとき EV_DIST_INVALID フラグが立ちます。

4.4 距離取得 dist.py

距離センサの出力を表示します。

```
import time
from owlifttypeh import OwhDevice

ow = OwhDevice.connect()
wx, wy = ow.frame_size
ow.set_options({"image_tab": False,
        "face_detect": False})                                  # 1

ow.capture_start()
fc0 = 0

while ow.alive:
    fc = ow.frame_counter
    if fc == fc0:
        time.sleep(0.01)
        continue

    fc0 = fc
    _, meta = ow.get_frame()
    print("distance = {:4d}".format(meta.distance))             # 2
```

1. オプションの face_detect:False で人体検知を OFF にします。人体検知を使用しないとき処理の負荷を低くする効果があります。
2. 距離センサの出力である meta.distance を取得して表示します。測定可能範囲外のときは 0 です。

4.5 LED 操作 led-test.py

LED を操作します。

```
import time
from owlifttypeh import OwhDevice, OwhLedStat

ow = OwhDevice.connect()

ow.set_led(OwhLedStat.ON, OwhLedStat.ON, OwhLedStat.ON, OwhLedStat.ON)                    # 1
time.sleep(2)
ow.set_led(OwhLedStat.OFF, OwhLedStat.ON, OwhLedStat.OFF, OwhLedStat.ON)                  # 2
time.sleep(2)
ow.set_led(OwhLedStat.ON, OwhLedStat.OFF, OwhLedStat.ON, OwhLedStat.OFF)                  # 3
time.sleep(1)
ow.set_led(OwhLedStat.OFF, OwhLedStat.OFF, OwhLedStat.OFF, OwhLedStat.OFF)                # 4
time.sleep(1)
ow.set_led(OwhLedStat.FLASH, OwhLedStat.FLASH, OwhLedStat.FLASH, OwhLedStat.FLASH)        # 5
time.sleep(2)
ow.set_led(OwhLedStat.BLINK_ON, OwhLedStat.BLINK_ON,
        OwhLedStat.BLINK_ON, OwhLedStat.BLINK_ON)                                         # 6
time.sleep(3)
ow.set_led(OwhLedStat.BLINK_ON, OwhLedStat.BLINK_OFF,
        OwhLedStat.BLINK_ON, OwhLedStat.BLINK_OFF)                                        # 7
time.sleep(3)
ow.set_led(OwhLedStat.OFF, OwhLedStat.OFF, OwhLedStat.OFF, OwhLedStat.OFF)                # 8
```

各番号において、LED が下表の状態に変更されます。

	デバイス 0		デバイス 1	
	赤	緑	赤	緑
1	ON	ON	ON	ON
2	OFF	ON	OFF	ON
3	ON	OFF	ON	OFF
4	OFF	OFF	OFF	OFF
5	一瞬 ON	一瞬 ON	一瞬 ON	一瞬 ON
6	ON⇒点滅	ON⇒点滅	ON⇒点滅	ON⇒点滅
7	ON⇒点滅	OFF⇒点滅	ON⇒点滅	OFF⇒点滅
8	OFF	OFF	OFF	OFF