OptrisCamera transport handler

The OptrisCamera transport handler allows receiving videos and images from Optris Infrared cameras using the Optris PI Connect SDK. The operator can only operate in push mode.

Options

- **`serialnumber`**: The serial number of the camera to use. If no serial number is specified, the first free camera will be used. (optional)

Schema

The output of the handler provides the following attributes as a schema. All attributes are optional and can be in an arbitrary order and will be identified by the type (IMAGEJCV, STARTTIMESTAMP) or name (flagstate).

Name	Type	Description
image	IMAGEJCV	The current frame of the video stream
starttimestamp	STARTTIMESTAMP	The start time stamp of the frame
flagstate	STRING	Current state of the camera flag. The flag is periodically (~12s) closed to re-calibrate the camera sensor which results in a short delay (~500ms). Possible values are: • fsFlagOpen: The camera flag is opened and new images are sent • fsFlagClose: The camera flag is closed and the last image before closing is sent • fsFlagClosing: The flag is currently closing • fsFlagOpening: The flag is currently opening • fsError: An error occured

Example: Show temperature video

This example shows how to grab a thermal video from an optris camera (16-bit 1 channel), transform it to a grayscale RGB image, and display it in a window. Temperatures are mapped as a linear function from 10.00°C -> 0 (black) to 40.00°C -> 255 (white).

PQL

Show temperature video

```
video = ACCESS({source='Video',
                                wrapper='GenericPush',
                                transport='OptrisCamera',
                                protocol='none',
                                datahandler='Tuple',
                                schema= [
                                            ['image', 'IMAGEJCV'],
                                            ['timestamp', 'STARTTIMESTAMP'],
                            ['flagState', 'STRING']
                                        ]})

/// or shorter, as a source operator:
video = OPTRISCAMERA({source='Video'})

grayscale = MAP({expressions = [['stretchContrastCV(image, 1000, 4000, 0, 255)', 'rgb_grayscale']]}, video)
output = UDO({class='ShowImageJCV', init='0,IR Video'}, grayscale)
```

Example: Encode temperature video to video file

This example shows how to grab a thermal video from an optris camera and encode it into a video file. Since most video codecs operate on 24- or 32-bit RGB(A) or YUV data, the FFV1 codec (34) is used which supports 32-bit RGBA data (pixel format 30). The 16-bit 1 channel temperature video is then reinterpreted as a 32-bit RGBA video with half of the width, where 2 16-bit pixels form one 32-bit pixel. Since fusing two pixels into one on bit level introduces jagged edges which confuse normal video encoders, it is necessary to use FFV1 since it is a lossless video compression algorithm.

Attention: Although the FFV1 codec writes RGBA video, the pixel format has to be set to BGRA (30)! Upon reading the video RGBA (28) can be used to retreive the original data.

PQL

Encode temperature video to video file

```
/// Grab Optris IR video stream
original = OPTRISCAMERA({source='Video'})

/// Reinterpret image: 2 16-bit pixels form one 32-bit pixel. Width
halves.
encoded = MAP({expressions = [['reinterpretCV(image, getWidthCV(image)/2, getHeightCV(image), 8, 4, 28)',
'image']]}, original)

/// Write to file
file = SENDER({sink='videofile',
                  wrapper='GenericPush',
               transport='none',
               protocol='FFmpegVideoStream',
               dataHandler='Tuple',
               options=[
                                                    ['streamUrl', 'ir.avi'],
                        ['format', 'avi'],                                  /// Necessary if streamUrl doesn't
end with .avi, since FFmpeg can't determine the format then
                                              ['videoCodec', '34'],                       /// FFV1
                                                  ['frameRate', '27'],                       /// Optris camera
operates at 27Hz
                                                  ['videoQuality', '0'],                        /// Optimize
Quality -> Lossless
                                              ['pixelformat', '30'],                       /// Need to supply
images in BGRA, but images get written as RGBA
                                                  ['codec:preset', 'veryslow'],        /// Best encoding ->
Lossless
                                                  ['frameSizeMultiple', '2']                /// Codec requires
image width and height with multiple of 2, extend image without stretching to preserve original data
                      ]}, encoded)
```

Example: Display encoded temperature video

This example shows how to read an RGBA-encoded temperature video, convert it back to its original 16-bit 1-channel format and display it as an RGB grayscale image.

PQL

Display encoded temperature video

```
encoded = FFMPEGVIDEO({source='Video', options=[
                                                    ['streamUrl', 'ir.avi'],
                                                    ['timeStampMode', 'filetime'],
                                                    ['useDelay', 'true'],
                                                    ['pixelFormat', '28'], /// RGBA
                                                    ['bitsPerPixel', '32']
                                             ]})

/// Reinterpret image: 1 32-bit pixel is extended to 2 16-bit pixels. Width
doubles.
original = MAP({expressions = [['reinterpretCV(image, getWidthCV(image)*2, getHeightCV(image), 16, 1, 28)',
'image']]}, encoded)
grayscale = MAP({expressions = [['stretchContrastCV(image, 2000, 4000, 0, 255)', 'image']]},
original)

output = UDO({class='ShowImageJCV', init='0,IR Video'},
grayscale)
```