VOutputV4L2 C++ library

v1.0.0

Table of contents

Overview

VOutputV4L2 C++ library provides video output based on **V4L2 API** for Linux OS. **VOutputV4L2** class depends on open source libraries: **Frame** (describes video frame structure and pixel formats) and **Logger** (provides method to write logs).

Versions

Table 1 - Library versions.

Version	Release date	What's new
1.0.0	10.08.2023	First version

VOutputV4L2 class description

VOutputV4L2 class declaration

VOutputV4L2 class declared in **VOutputV4L2.h** file. Class declaration:

```
class VOutputV4L2
{
public:
    /**
     * @brief Get library version.
     * @return String orf current library version: Major.Minor.Patch.
     */
    static std::string getVersion();
    /**
     * @brief Class constructor.
     */
    VOutputV4L2();
    /**
     * @brief Class destructor.
     */
    ~VOutputV4L2();
    /**
     * @brief Open device.
     * @param device device name. E.g "/dev/video4".
     * @return TRUE if devicde is open or FALSE if not.
     */
    bool open(std::string device);
    /**
     * @brief Write data to device.
     * @param frame Frame object.
     * @return TRUE is frame was written or FALSE if not.
     */
    bool write(Frame& frame);
    /**
     * @brief Set log level.
     * @param flag Log flag.
     */
    void setLogLevel(cr::utils::PrintFlag flag);
    /**
     * @brief Close device.
     */
    void close(void);
};
```

getVersion method

getVersion() method returns string of current version of **VOutputV4L2** class. Method declaration:

```
static std::string getVersion();
```

Method can be used without **VOutputV4L2** class instance:

```
cout << "VOutputV4L2 class version: " << VOutputV4L2::getVersion() << endl;
```

Console output:

```
VOutputV4L2 class version: 1.0.0
```

open method

open(...) method initializes video output device, checks capabilities of device and turns stream on.

```
bool open(string device);
```

Parameter	Value
device	v4l2 video device name. (/dev/video4 etc.)

Returns: TRUE if the video device open or FALSE if not.

write method

write(...) method writes input arguments frame into output device.

```
bool write(Frame& frame);
```

Parameter	Value
frame	Output video frame (see **Frame** class description). Please check **table-2** for supported pixel formats.

Table 2 - Supported Pixel format.

Fourcc	RGB24	BGR24	YUYV	UYVY	GRAY	YUV24	NV12	NV21	YU12	YV12
Support	Yes	Yes	Yes	Yes	Yes	No	Yes	No	Yes	Yes

Returns: TRUE if the format is written to device or FALSE if not.

setLogLevel method

setLogLevel(...) method designed to set new video source parameters value. Method declaration:

```
void setLogLevel(cr::utils::PrintFlag flag);
```

Parameter	Description
flag	Logger flag which defined in **Logger.h** from Logger library.

close method

close() method intended to close output device and turn stream off. Method declaration:

```
void close();
```

Example

Below is the code for a simple application streaming of video content from a file onto the /dev/video4 device.

```
#include <iostream>
#include <opencv2/opencv.hpp>
#include "VOutputV4L2.h"

// Link namespaces.
using namespace cr::video;
using namespace std;
using namespace std::chrono;

/// Entry point.
int main(void)
{
    // video device name
    std::string deviceName = "/dev/video4";

    // video output device
    VOutputV4L2 videoOutput;

    //open video device
    if(!videoOutput.open(deviceName))
    {
        return -1;
    }

    //Set logger flag.
    videoOutput.setLogLevel(cr::utils::PrintFlag::CONSOLE);

    // Open video file.
```

```
    cv::VideoCapture videoSource;
    if (!videoSource.open(0))
    {
        std::cerr << "Video file test.mp4 not open" << std::endl;
        std::this_thread::sleep_for(std::chrono::seconds(1));
        return -1;
    }

    // Get frame size.
    int width = (int)videoSource.get(cv::CAP_PROP_FRAME_WIDTH);
    int height = (int)videoSource.get(cv::CAP_PROP_FRAME_HEIGHT);

    // Init frames.
    cv::Mat inputFrameBgr(cv::Size(width, height), CV_8UC3);
    Frame srcFrame(width, height, Fourcc::BGR24);

    // Main loop.
    int frameId = 0;

    while (true)
    {
        // Capture next video frame.
        videoSource >> inputFrameBgr;

        if (inputFrameBgr.empty())
        {
            // Set first video frame position.
            videoSource.set(cv::CAP_PROP_POS_FRAMES, 1);
            continue;
        }

        // Copy bgr frame into srcFrame data.
        memcpy(srcFrame.data, inputFrameBgr.data, srcFrame.size);

        // Put frame into output device.
        if(!videoOutput.write(srcFrame))
        {
            std::cerr << "Error writting frame to output device. " << std::endl;
        }

        // 33ms delay for 30 fps stream
        std::this_thread::sleep_for(std::chrono::milliseconds(33));
    }

    return 1;
}
```

Build and connect to your project

Typical commands to build **VOutputV4L2** library:

```
git clone https://github.com/ConstantRobotics-Ltd/VOutputV4L2.git
cd VOutputV4L2
git submodule update --init --recursive
mkdir build
cd build
cmake ..
make
```

If you want connect **VOutputV4L2** library to your CMake project as source code you can make follow. For example, if your repository has structure:

```
CMakeLists.txt
src
    CMakeList.txt
    yourLib.h
    yourLib.cpp
```

You can add repository **VOutputV4L2** as submodule by commands:

```
cd <your respository folder>
git submodule add https://github.com/ConstantRobotics-Ltd/VOutputV4L2.git
3rdparty/VOutputV4L2
git submodule update --init --recursive
```

In you repository folder will be created folder **3rdparty/VOutputV4L2** which contains files of **VOutputV4L2** repository with subrepositories **Frame** and **Logger**. New structure of your repository:

```
CMakeLists.txt
src
    CMakeList.txt
    yourLib.h
    yourLib.cpp
3rdparty
    VSourceV4L2
```

Create CMakeLists.txt file in **3rdparty** folder. CMakeLists.txt should contain:

```
cmake_minimum_required(VERSION 3.13)

################################################################################
## 3RD-PARTY
## dependencies for the project
################################################################################
project(3rdparty LANGUAGES CXX)

################################################################################
```

```
## SETTINGS
## basic 3rd-party settings before use
################################################################################
# To inherit the top-level architecture when the project is used as a submodule.
SET(PARENT ${PARENT}_YOUR_PROJECT_3RDPARTY)
# Disable self-overwriting of parameters inside included subdirectories.
SET(${PARENT}_SUBMODULE_CACHE_OVERWRITE OFF CACHE BOOL "" FORCE)

################################################################################
## CONFIGURATION
## 3rd-party submodules configuration
################################################################################
SET(${PARENT}_SUBMODULE_VOUTPUT_V4L2                    ON  CACHE BOOL "" FORCE)
if (${PARENT}_SUBMODULE_VOUTPUT_V4L2)
    SET(${PARENT}_VOUTPUT_V4L2                          ON  CACHE BOOL "" FORCE)
    SET(${PARENT}_VOUTPUT_V4L2_TEST                     OFF CACHE BOOL "" FORCE)
    SET(${PARENT}_VOUTPUT_V4L2_EXAMPLE                  OFF CACHE BOOL "" FORCE)
endif()

################################################################################
## INCLUDING SUBDIRECTORIES
## Adding subdirectories according to the 3rd-party configuration
################################################################################
if (${PARENT}_SUBMODULE_VOUTPUT_V4L2)
    add_subdirectory(VOutputV4L2)
endif()
```

File **3rdparty/CMakeLists.txt** adds folder **VOutputV4L2** to your project and excludes test application (VOutputV4L2 class test applications) from compiling. Your repository new structure will be:

```
CMakeLists.txt
src
    CMakeList.txt
    yourLib.h
    yourLib.cpp
3rdparty
    CMakeLists.txt
    VOutputV4L2
```

Next you need include folder 3rdparty in main **CMakeLists.txt** file of your repository. Add string at the end of your main **CMakeLists.txt**:

```
add_subdirectory(3rdparty)
```

Next you have to include VOutputV4L2 library in your **src/CMakeLists.txt** file:

```
target_link_libraries(${PROJECT_NAME} VOutputV4L2)
```

NOTE: You should run your application which includes VOutputV4L2 library with root (sudo) privileges.