VCodecLibav C++ library

v1.0.1

Table of contents

Overview

VCodecLibav C++ library provides video encoding and decoding functions for H264, HEVC(H265) and JPEG codecs. **VCodecLibav** video codec class inherits interface and data structures form **VCodec** library. **VCodecLibav** uses **libav** library from **ffmpeg** library. Version 1.0.1 supports only **intel integrated GPU** and uses intel hardware encoders and decoders via **ffmpeg**. Supported **ffmpeg** encoders and decoders:

Codec	ffmpeg encoder	ffmpeg decoder
H264	h264_vaapi (Intel VAAPI based codec)	h264_qsv (Intel QuickSync based) - not tested
HEVC (H265)	hevc_vaapi (Intel VAAPI based codec)	hevc_qsv (Intel QuickSync based) - not tested
JPEG	mjpeg_vaapi (Intel VAAPI based codec)	mjpeg_qsv (Intel QuickSync based) - not tested

Versions

Table 1 - Library versions.

Version	Release date	What's new
1.0.0	29.06.2023	First version.
1.0.1	02.07.2023	- Source code cleaned up. - Documentation updated.

Video codec class description

Class declaration

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

```
class VCodecLibav : public VCodec
{
public:
    /**
     * @brief Class constructor.
     */
    VCodecLibav();
    /**
     * @brief Class destructor.
     */
    ~VCodecLibav();
    /**
     * @brief Get string of current library version.
     * @return String of current library version in format "Major.Minor.Patch".
     */
    static std::string getVersion();
    /**
     * @brief Encode or decode frame.
     * @param src Source frame (RAW or compressed).
     * @param dst Result frame (RAW or compressed).
     * @return TRUE if frame was encoded/decoded or FLASE if not.
     */
    bool transcode(Frame& src, Frame& dst);
    /**
     * @brief Set parameter.
     * @param id Parameter ID.
     * @param value Parameter value.
     * @return TRUE if parameter was set of FALSE.
     */
    bool setParam(VCodecParam id, float value);
    /**
     * @brief Get parameter.
     * @param id Parameter ID.
```

```
     * @return Parameter value or -1.
     */
    float getParam(VCodecParam id);
    /**
     * @brief Execute command.
     * @param id Command ID .
     * @return TRUE if the command accepted or FALSE if not.
     */
    bool executeCommand(VCodecCommand id);
};
```

getVersion method

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

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

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

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

Console output:

```
VCodecLibav class version: 1.0.1
```

transcode method

transcode(...) method intended to encode and decode video frame (**Frame** class). Video codec encodes/decodes video frames frame-by-frame. Method declaration:

```
bool transcode(Frame& src, Frame& dst);
```

Parameter	Value
src	Source video frame (see **Frame** class description). To encode video data **src** frame must have **NV12** pixel format. To decode video data **src** frame must have compressed pixel format (field **fourcc** of **Frame** class): **JPEG**, **H264** or **HEVC**.
dst	Result video frame (see **Frame** class description). To decode video data **src** frame must have compressed pixel format (field **fourcc** of **Frame** class): **JPEG**, **H264** or **HEVC**. In case decoding particular video codec will set **NV12** pixel format automatically.

Returns: TRUE if frame was encoded/decoded or FALSE if not.

setParam method

setParam(...) method designed to set new video codec parameters value. Method declaration:

```
setParam(VCodecParam id, float value);
```

Parameter	Description
id	Video codec parameter ID according to **VCodecParam** enum.
value	Video codec parameter value.

Returns: TRUE is the parameter was set or FALSE if not.

getParam method

getParam(...) method designed to obtain video codec parameter value. Method declaration:

```
float getParam(VCodecParam id);
```

Parameter	Description
id	Video codec parameter ID according to **VCodecParam** enum.

Returns: parameter value or -1 of the parameters doesn't exist in particular video codec class.

executeCommand method

executeCommand(...) method designed to execute video codec command. Version 1.0.1 doesn't support commands. Method will return FALSE. Method declaration:

```
bool executeCommand(VCodecCommand id);
```

Parameter	Description
id	Video codec command ID according to **VCodecCommand** enum.

Returns: method returns FALSE in any case.

Data structures

VCodec.h file defines IDs for parameters (**VCodecParam** enum) and IDs for commands (**VCodecCommand** enum).

VCodecCommand enum

Enum declaration:

```
enum class VCodecCommand
{
    /// Reset.
    RESET = 1,
    /// Generate key frame. For H264 and H265 codecs.
    MAKE_KEY_FRAME
};
```

Table 2 - Video codec commands description. Some commands maybe unsupported by particular video codec class.

Command	Description
RESET	Not supported.
MAKE_KEY_FRAME	Not supported.

VCodecParam enum

Enum declaration:

```
enum class VCodecParam
{
    /// [read/write] Log level:
    /// 0-Disable, 1-Console, 2-File, 3-Console and file.
    LOG_LEVEL = 1,
    /// [read/write] Bitrate, kbps. For H264 and H265 codecs.
    BITRATE_KBPS,
    /// [read/write] Quality 0-100%. For JPEG codecs.
    QUALITY,
    /// [read/write] FPS. For H264 and H265 codecs.
    FPS,
    /// [read/write] GOP size. For H264 and H265 codecs.
    GOP,
    /// [read/write] H264 profile: 0 - Baseline, 1 - Main, 2 - High.
    H264_PROFILE,
    /// [read/write] Codec type. Depends on implementation.
    TYPE,
    /// Custom 1. Depends on implementation.
    CUSTOM_1,
    /// Custom 2. Depends on implementation.
    CUSTOM_2,
    /// Custom 3. Depends on implementation.
    CUSTOM_3
};
```

Table 3 - Video codec params description. Some params maybe unsupported by particular video codec class.

Parameter	Access	Description
LOG_LEVEL	read / write	Logging mode. Default values: 0 - Disable, 1 - Only file, 2 - Only terminal, 3 - File and terminal.
BITRATE_KBPS	read / write	Bitrate, kbps. For H264 and H265(HEVC) encoding. According to this value, FPS and GOP size video codec calculate parameter for H264 or H265(HEVC) encoding.
QUALITY	read / write	Quality 0(low quality)-100%(maximum quality). Only for JPEG encoding.
FPS	read / write	FPS. For H264 and H265 codecs. According to this value, FPS and GOP size video codec calculate parameter for H264 or H265(HEVC) encoding.
GOP	read / write	GOP size (Period of key frames) for H264 or H265(HEVC) encoding. Value: 1 - each output frame is key frame, 20 - each 20th frame is key frame etc.
H264_PROFILE	read / write	H264 profile for H264 encoding: 0 - Baseline, 1 - Main, 2 - High.
TYPE	read / write	Not supported.
CUSTOM_1	read / write	Not supported.
CUSTOM_2	read / write	Not supported.
CUSTOM_3	read / write	Not supported.

Build and connect to your project

Typical commands to build **VCodecLibav** library:

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

If you want connect **VCodecLibav** 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 **VCodecLibav** as submodule by commands:

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

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

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

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(${PARENT}_SUBMODULE_VCODEC_LIBAV                    ON  CACHE BOOL "" FORCE)
if (${PARENT}_SUBMODULE_VCODEC_LIBAV)
    SET(${PARENT}_VCODEC_LIBAV                          ON  CACHE BOOL "" FORCE)
    SET(${PARENT}_VCODEC_LIBAV_TEST                     OFF CACHE BOOL "" FORCE)
endif()
```

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

File **3rdparty/CMakeLists.txt** adds folder **VCodecLibav** to your project. Your repository new structure will be:

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

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 VCodecLibav library in your **src/CMakeLists.txt** file:

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

Done!