VCodecOneVpl C++ library

v2.0.2

Table of contents

Overview

VCodecOneVpl C++ library provides hardware video **encoding/decoding** (H264, HEVC and JPEG) for Intel HD Graphics based on oneVPL API. VCodecOneVpl class inherits interface and data structures from open source VCodec library and also includes Logger open source library which provides function for print logs. The library provides simple interface to be implemented in different C++ projects. It is written with C++17 standard. The library is supplied as source code only in form of CMake project.

Encoding time for 11th Gen Intel(R) Core(TM) **i5-1145G7E** on **Ubuntu 22.04 LTS**:

codec / resolution	2560x1440	1920x1080	1280x720	640x512
H264	11.6 msec	8.6 msec	4.4 msec	2.6 msec
HEVC	23.4 msec	15.2 msec	9.3 msec	5.2 msec

codec / resolution	2560x1440	1920x1080	1280x720	640x512
JPEG	8.2 msec	4.8 msec	2.5 msec	1.2 msec

Decoding time for 11th Gen Intel(R) Core(TM) **i5-1145G7E** on **Ubuntu 22.04 LTS**:

codec / resolution	2560x1440	1920x1080	1280x720	640x512
H264	7.3 msec	4.8 msec	2.5 msec	1.3 msec
HEVC	7.4 msec	4.3 msec	2.2 msec	1.3 msec
JPEG	8.7 msec	5.2 msec	2.6 msec	1.3 msec

Versions

Table 1 - Library versions.

Version	Release date	What's new
1.0.0	01.12.2022	First version.
2.0.0	29.09.2023	- Interface changes to VCodec. - Added decoding support. - Added JPEG support.
2.0.1	12.11.2023	- Frame class updated.
2.0.2	10.01.2023	- VCodec interface updated. - Examples updated. - Documentation updated.

Library files

The library is supplied only by source code. The user is given a set of files in the form of a CMake project (repository). The repository structure is shown below:

```
CMakeLists.txt --------------- Main CMake file of the library.
3rdparty --------------------- Folder with third-party libraries.
    CMakeLists.txt ----------- CMake file which includes third-party libraries.
    Logger ------------------- Source code of Logger library.
    VCodec ------------------- Source code of VCodec library.
example ---------------------- Folder with simple example of VCodecOneVpl usage.
    CMakeLists.txt ----------- CMake file for example application.
    main.cpp ----------------- Source code file of example application.
test ------------------------- Folder with codec test application.
    CMakeLists.txt ----------- CMake file for transcode test application.
    main.cpp ----------------- Source code file of transcode test application.
src -------------------------- Folder with source code of the library.
```

```
    CMakeLists.txt ----------- CMake file of the library.
    VCodecOneVpl.cpp --------- Source code file of the library.
    VCodecOneVpl.h ----------- Header file which includes VCodecOneVpl class declaration.
    VCodecOneVplVersion.h ---- Header file which includes version of the library.
    VCodecOneVplVersion.h.in - CMake service file to generate version file.
```

Additionally test application depends on OpenCV library to provide user interface.

VCodecOneVpl class description

VCodecOneVpl class declaration

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

```
class VCodecOneVpl : public VCodec
{
public:

    /// Get library version.
    static std::string getVersion();

    /// Class constructor.
    VCodecOneVpl();

    /// Class destructor.
    ~VCodecOneVpl();

    /// Set parameter value.
    bool setParam(VCodecParam id, float value) override;

    /// Get parameter value.
    float getParam(VCodecParam id) override;

    /// Encode/Decode video frame.
    bool transcode(Frame& src, Frame& dst) override;

    /// Execute command.
    bool executeCommand(VCodecCommand id) override;
};
```

getVersion method

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

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

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

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

Console output:

```
VCodecOneVpl class version: 2.0.2
```

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 **src** frame must have raw pixel format such as **NV12**. To decode video data **src** frame must have compressed pixel format (field **fourcc** of Frame class): **HEVC**, **JPEG**, **H264**.
dst	Result video frame (see Frame class description). To encode video data **dst** frame must have compressed pixel format (field **fourcc** of Frame class): **HEVC**, **JPEG**, **H264**. In contrary, src frame must be **NV12**.

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:

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

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.

VCodec.h file of VCodec library defines IDs for parameters (**VCodecParam** enum) and IDs for commands (**VCodecCommand** enum). VCodecParam 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 2 - Video codec params description. Some params not supported by VCodecOneVpl library.

Parameter	Access	Description
LOG_LEVEL	read / write	Logging mode. Valid values: **0** - Disable. **1** - Only file. **2** - Only terminal (console). **3** - File and terminal.
BITRATE_KBPS	read / write	Bitrate, kbps. For H264 encoding only. According to this value, FPS and GOP size video codec calculate parameter for H264 encoding.
QUALITY	read / write	Quality 0(low quality)-100%(maximum quality). Only for JPEG encoding.
FPS	read / write	FPS. For H264 encoding only. According to this value, FPS and GOP size video codec calculate parameter for H264 encoding.
GOP	read / write	GOP size (Period of key frames) for H264 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** by VCodecOneVpl library.
CUSTOM_1	read / write	**Not supported** by VCodecOneVpl library.
CUSTOM_2	read / write	**Not supported** by VCodecOneVpl library.
CUSTOM_3	read / write	**Not supported** by VCodecOneVpl library.

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 (see Table 2).

Returns: parameter value or -1 if the parameters not supported.

executeCommand method

executeCommand(...) method designed to execute video codec command. Version 2.0.2 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.

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

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

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

Command	Description
RESET	**Not supported** by VCodecOneVpl library.
MAKE_KEY_FRAME	**Not supported** by VCodecOneVpl library.

Build and connect to your project

Before build you have to install OneVpl on your system. Follow Installation on Linux or Installation on Windows instructions. Typical commands to build **VCodecOneVpl** library:

```
cd VCodecOneVpl
git submodule update --init --recursive
mkdir build
cd build
cmake ..
make
```

If you want connect **VCodecOneVpl** 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
```

Create folder **3rdparty** in your repository. Copy repository folder **VCodecOneVpl** to **3rdparty** folder. New structure of your repository:

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

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(${PARENT}_SUBMODULE_VCODEC_ONE_VPL                  ON  CACHE BOOL "" FORCE)
if (${PARENT}_SUBMODULE_VCODEC_ONE_VPL)
    SET(${PARENT}_VCODEC_ONE_VPL                        ON  CACHE BOOL "" FORCE)
    SET(${PARENT}_VCODEC_ONE_VPL_TEST                   OFF CACHE BOOL "" FORCE)
    SET(${PARENT}_VCODEC_ONE_VPL_EXAMPLE                OFF CACHE BOOL "" FORCE)
endif()

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

File **3rdparty/CMakeLists.txt** adds folder **VCodecOneVpl** to your project and will exclude test application from compiling. Your repository new structure will be:

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

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

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

Done!

Installation on Linux

There are several steps to launching VCodecOneVpl on Linux (tested on Ubuntu 22.04 LTS):

1. Install ubuntu with last updates:

```
sudo apt-get update
sudo apt-get upgrade
sudo reboot
```

2. Install LibVA:

```
sudo apt-get install git cmake pkg-config meson libdrm-dev automake libtool
cd Downloads
git clone https://github.com/intel/libva.git
cd libva
./autogen.sh --prefix=/usr --libdir=/usr/lib/x86_64-linux-gnu
make
sudo make install
```

3. Install gmmlib:

```
cd Downloads
git clone https://github.com/intel/gmmlib.git
cd gmmlib
mkdir build
cd build
cmake -DCMAKE_BUILD_TYPE=Release ..
make -j"$(nproc)"
sudo make install
```

4. Install intel media driver for VAAPI:

```
cd Downloads
git clone https://github.com/intel/media-driver.git
mkdir build_media && cd build_media
cmake ../media-driver
make -j"$(nproc)"
sudo make install
```

5. Install oneVPL-intel-gpu:

```
cd Downloads
git clone https://github.com/oneapi-src/oneVPL-intel-gpu onevpl-gpu
cd onevpl-gpu
mkdir build && cd build
cmake ..
make -j"$(nproc)"
sudo make install
```

6. Install additional packets to oneVPL-intel-gpu:

```
sudo apt update
sudo apt install -y gpg-agent wget
wget -qO - https://repositories.intel.com/gpu/intel-graphics.key | sudo gpg --dearmor
--output /usr/share/keyrings/intel-graphics.gpg
echo "deb [arch=amd64 signed-by=/usr/share/keyrings/intel-graphics.gpg]
https://repositories.intel.com/gpu/ubuntu jammy/production/2328 unified" | sudo tee
/etc/apt/sources.list.d/intel-gpu-jammy.list
sudo apt update
sudo apt install -y linux-headers-$(uname -r) flex bison intel-fw-gpu intel-i915-dkms
xpu-smi
sudo reboot
sudo apt install -y intel-opencl-icd intel-level-zero-gpu level-zero intel-media-va-
driver-non-free libmfx1 libmfxgen1 libvpl2 libegl-mesa0 libegl1-mesa libegl1-mesa-dev
libgbm1 libgl1-mesa-dev libgl1-mesa-dri libglapi-mesa libgles2-mesa-dev libglx-mesa0
libigdgmm12 libxatracker2 mesa-va-drivers mesa-vdpau-drivers mesa-vulkan-drivers va-
driver-all vainfo hwinfo clinfo
sudo apt install -y libigc-dev intel-igc-cm libigdfcl-dev libigfxcmrt-dev level-zero-
dev
```

7. Install oneVPL:

```
cd Downloads
git clone https://github.com/oneapi-src/oneVPL.git
cd oneVPL
sudo script/bootstrap
script/build
sudo script/install
```

Installation on Windows

1. Install oneVPL:

```
git clone https://github.com/oneapi-src/oneVPL.git
cd oneVPL
script/bootstrap.bat
script/build.bat
script/install.bat
```

Simple example

Example application generates image color pattern with moving rectangle and writes compressed data to binary file **"out.hevc"**. Example shows how to create codec objects and how to encode video frames:

```
#include <iostream>
#include "VCodecOneVpl.h"

int main(void)
{
    // Set codec parameters.
```

```
cr::video::VCodec* videoCodec = new cr::video::VCodecOneVpl();
videoCodec->setParam(cr::video::VCodecParam::BITRATE_KBPS, 7500);
videoCodec->setParam(cr::video::VCodecParam::GOP, 30);
videoCodec->setParam(cr::video::VCodecParam::FPS, 30);

// Create NV12 frame and fill color planes by random values.
const int width = 1280;
const int height = 720;
cr::video::Frame frameNv12(width, height, cr::video::Fourcc::NV12);
for (uint32_t i = 0; i < frameNv12.size; ++i)
    frameNv12.data[i] = (uint8_t)i;

// Create output HEVC frame.
cr::video::Frame frameHEVC(width, height, cr::video::Fourcc::HEVC);

// Create output file.
FILE *outputFile = fopen("out.hevc", "w+b");

// Params for moving object.
int objectWidth = 128;
int objectHeight = 128;
int directionX = 1;
int directionY = 1;
int objectX = width / 4;
int objectY = height / 2;

// Encode and record 200 frames.
for (uint32_t n = 0; n < 200; ++n)
{
    // Draw moving object.
    memset(frameNv12.data, 128, width * height);
    for (int y = objectY; y < objectY + objectHeight; ++y)
        for (int x = objectX; x < objectX + objectHeight; ++x)
            frameNv12.data[y * width + x] = 255;
    objectX += directionX;
    objectY += directionY;
    if (objectX >= width - objectWidth - 5 || objectX <= objectWidth + 5)
        directionX = -directionX;
    if (objectY >= height - objectHeight - 5 || objectY <= objectHeight + 5)
        directionY = -directionY;

    // Encode.
    if (!videoCodec->transcode(frameNv12, frameHEVC))
    {
        std::cout << "Can't encode frame" << std::endl;
        continue;
    }

    // Write to file.
    fwrite(frameHEVC.data, frameHEVC.size, 1, outputFile);
}

// Close file.
fclose(outputFile);
```

```
    return 1;
}
```

Test application

Test application (VCodecOneVpl/test/main.cpp) for **VCodecOneVpl** C++ library shows how library works on Intel platform. Test application generates artificial video, compresses it according to user's parameters (codec type, bitrate or JPEG quality, GOP size and H264 profile) and writes results to binary file "out.h264", "out.hevc" or "out.mjpeg". To run application perform commands on Linux or just run **VCodecOneVplTest.exe** on Windows:

```
cd <application folder>
sudo chmod +x VCodecOneVplTest
./VCodecOneVplTest
```

After start you will see output:

```
====================================
VCodecOneVpl v2.0.2 test
====================================

Enter Encoder type (0 - JPEG, 1 - H264, 2 - HEVC) :
```

Chose codec type (0 - JPEG, 1 - H264). If H264 codec chosen you will see message:

```
====================================
VCodecOneVpl v2.0.2 test
====================================

Enter Encoder type (0 - JPEG, 1 - H264, 2 - HEVC) : 1
Default params:
Bitrate, kbps 6000
FPS: 30
GOP size: 30
Video width 1920
Video height 1080
H264 Profile: BASELINE
Use default params (0 - no, 1 - yes) :
```

When params chosen test application will create out.h264 file and will start writing encoded frames until stop. User can set custom parameters (video resolution, bitrate, GOP size and H264 profile). If JPEG codec chosen you will see message:

```
====================================
VCodecOneVpl v2.0.2 test
====================================

Enter Encoder type (0 - JPEG, 1 - H264, 2 - HEVC) : 0
Default params:
Bitrate, kbps 6000
FPS: 30
GOP size: 30
Video width 1920
Video height 1080
Quality: 95
Use default params (0 - no, 1 - yes) :
```

When params chosen test application will start writing encoded frames until stop (if user set number of frames). During encoding the application shows encoded data size and encoding time and decoding time:

```
Data size 328076/3110400  encoding time msec  2.19 ||  2.393 msec decoding time
Data size 328807/3110400  encoding time msec  2.199 ||  2.763 msec decoding time
Data size 327878/3110400  encoding time msec  2.168 ||  2.39 msec decoding time
Data size 327884/3110400  encoding time msec  2.125 ||  2.306 msec decoding time
Data size 327623/3110400  encoding time msec  2.442 ||  2.183 msec decoding time
Data size 327498/3110400  encoding time msec  2.051 ||  2.187 msec decoding time
```