Introduction

In this experiment, performance (video stream encoding and decoding) and ease of use of the OpenCV library for CPU and iGPU were studied. For this purpose, the libraries libva (VA-API) and FFmpeg were used. FFMpeg was used for processing on CPU, because OpenCV wrapper for this library does not support hardware acceleration yet.

Several popular codecs (mpeg2, mpeg4, h264, h265, mjpeg, vp8) and containers (mkv, mp4, mov, avi) have been chosen for the evaluation.

FFmpeg is the leading multimedia framework, able to decode, encode, transcode, mux, demux, stream, filter and play pretty much anything that humans and machines have created. It supports the most obscure ancient formats up to the cutting edge. No matter if they were designed by some standards committee, the community or a corporation. It is also highly portable: FFmpeg compiles, runs, and passes our testing infrastructure FATE across Linux, Mac OS X, Microsoft Windows, the BSDs, Solaris, etc. under a wide variety of build environments, machine architectures, and configurations.

GStreamer is a library for constructing graphs of media-handling components. The applications it supports range from simple Ogg/Vorbis playback, audio/video streaming to complex audio (mixing) and video (non-linear editing) processing.

VA-API (Video Acceleration API) is an open-source library and API specification, which provides access to graphics hardware acceleration capabilities for video processing. It consists of a main library and driver-specific acceleration backends for each supported hardware vendor.

Intel® Media SDK provides an API to access hardware-accelerated video decode, encode and filtering on Intel® platforms with integrated graphics.

Benchmark tool

This is a program implemented as OpenCV sample application which allows user to select one of predefined video processing pipelines and measure its performance. The source code for this program can be found here

Prerequisites

In order to use all supported modes of the benchmark tool you should install several libraries, mostly GStreamer plugins. In Ubuntu 17 at can be done with the following commands:

```
# GStreamer library and plugins
sudo apt install \
    libgstreamer1.0-dev \
    gstreamer1.0-plugins-{base,good,bad} \
    libgstreamer-plugins-{base,good,bad}1.0-dev \
    gstreamer1.0-libav \
    gstreamer1.0-vaapi
# FFmpeg dev package
sudo apt install libavcodec-dev
```

Note: to use gstreamer-MediaSDK plugin you should install MediaSDK and build the plugin from sources. Official repository https://github.com/intel/gstreamer-media-SDK have several forks, for example https://github.com/ishmael1985/gstreamer-media-SDK which can have some issues fixed, but may be less tested. Building instruction can be found in instructions of the respective repository. Another plugin variants: https://github.com/Intel-Media-SDK/gstreamer-plugins and https://github.com/GStreamer/gst-plugins-bad/tree/master/sys/msdk - are not supported by the benchmark tool yet.

Build

To build the benchmark tool configure and build OpenCV as follows (Linux):

```
# configure
cmake -DBUILD_EXAMPLES=ON -DWITH_GSTREAMER=ON -DWITH_FFMPEG=ON
<path_to_opencv_source>
# build
make -j8
# check it
./bin/example_cpp_gstreamer_pipeline --help
```

Note: Windows platform can, in theory, work too, but we have not checked it yet.

Run

The benchmark tool supports two modes: *decode* and *encode*; and several backend configurations described below.

Default configuration (GStreamer, FFmpeg)

Backend option: *gst-default* or *ffmpeg*

VideoCapture/VideoWriter object is created using file name, OpenCV builds processing pipelines automatically:

```
VideoCapture cap(file_name, CAP_GSTREAMER);
// or
VideoCapture cap(file_name, CAP_FFMPEG);
```

Basic configuration (GStreamer)

Backend option: *gst-basic*

GStreamer pipeline is generated according to codec and file format, *bin* elements automatically choose appropriate plugins to process video stream:

```
VideoCapture cap(
    "filesrc location=<filename> ! avidemux ! decodebin ! appsink",
    CAP_GSTREAMER);
```

Note: the pipeline built in this mode is mostly equivalent to the one produced internally by OpenCV in the *gst-default* mode.

HW-accelerated configuration (GStreamer)

Backend option: *gst-vaapi* or *gst-mfx*

Decode/encode elements are selected from VAAPI or MediaSDK GStreamer plugins according to selected codec:

```
VideoCapture cap(
    "filesrc location=<filename> ! avidemux ! vaapidecodebin ! appsink",
    CAP_GSTREAMER);
// or
VideoCapture cap(
    "filesrc location=<filename> ! avidemux ! mfxh264dec ! appsink",
    CAP_GSTREAMER);
```

libav GStreamer plugin (GStreamer)

Backend option: *gst-libav*

In this mode decoding/encoding elements are selected from libav GStreamer plugin:

```
VideoCapture cap(
    "filesrc location=<filename> ! avidemux ! avdec_h264 ! appsink",
    CAP_GSTREAMER);
```

Results

Test video: full HD resolution (1920x1080) encoded with some default settings (using ffmpeg tool).

Test platform: Ubuntu 17.10 / Intel® Core™ i5-6600 CPU @ 3.30GHz / Intel® HD Graphics 530 (Skylake GT2)

Decoding (FPS):

format	gst-libav	ffmpeg	gst-vaapi
mkv / mpeg2	266	244	198 (x0.74)
mkv / h264	126	158	59 (x0.37)
mkv / h265	185	255	65 (x0.26)
mkv / vp8	226	309	68 (x0.22)
mp4 / mpeg2	343	258	364 (x1.06)
mp4 / h264	167	169	376 (x2.2)
mp4 / h265	233	256	391 (x1.5)
avi / mpeg2	306	243	328 (x1.07)
avi / h264	140	159	373 (x2.4)
avi / vp8	299	309	386 (x1.3)

Encoding (FPS):

format	gst-libav	ffmpeg	gst-vaapi
mkv / mpeg2	81	106	124 (x1.2)
mkv / h264	-	51	193 (x3.8)
mkv / vp8	-	12	124 (x10.3)
mp4 / h264	-	51	194 (x3.8)
avi / mpeg2	81	106	123 (x1.2)

avi / h264	-	50	185 (x3.7)
avi / vp8	-	12	115 (x9.6)