Video I/O hardware acceleration

Introduction

Since OpenCV 4.5.2 new properties are added to control H/W acceleration modes for video decoding and encoding tasks. New builtin properties brings easy to use API for OpenCV Users.

```
VideoCapture capture(filename, CAP_FFMPEG,
    {
        CAP_PROP_HW_ACCELERATION, VIDEO_ACCELERATION_ANY,
    }
);
```

Hardware-accelerated API is described in OpenCV API documentation.

:speech_balloon: This feature currently is in preview mode, so feel free to try it with your configuration and provide feedback through OpenCV issues.

Software prerequisites

OpenCV uses external Media I/O libraries and/or OS-provided APIs under unified `VideoCapture` and `VideoWriter` APIs. Wrapper code in OpenCV over some external framework is called backend.

H/W accelerated processing is supported through these libraries:

- FFmpeg 4.0+ with enabled `hwaccels` support: https://trac.ffmpeg.org/wiki/HWAccelIntro
- GStreamer 1.x+ with installed VAAPI plugin and others.
- (Windows) Microsoft Media Foundation (MSMF)

To properly utilize hardware capabilities there are several low-level API/SDKs used:

- (Linux) VA-API (Video Acceleration API) is an open-source library and API specification, which provides access to graphics hardware acceleration capabilities for video processing.

- (Windows) DirectX Video Acceleration (DXVA/DXVA2) is a Microsoft API that allows video decoding/encoding to be hardware-accelerated.
- Intel® Media SDK provides an API to access hardware-accelerated video decode, encode and filtering on Intel® platforms with integrated graphics. oneVPL (oneAPI Video Processing Library) is an evolution of Intel Media SDK.
- NVIDIA Video Codec SDK is a NVIDIA proprietary library for hardware-accelerated video decode/encode on CUDA-compatible GPUs.
- Video Decode and Presentation API for Unix (VDPAU) is an open source library and API to offload portions of the video decoding process and video post-processing to the GPU video-hardware, developed by NVIDIA.
- AMD AMF
- others APIs/SDK from hardware vendors

Installation guidelines for some Best Known Configurations (BKC) are described below.

Current HW acceleration types support matrix, in priority order:

OS	Backend	VideoCapture	VideoWriter
Linux	FFMPEG	VAAPI	MFX, VAAPI
GStreamer	VAAPI (and others HW plugins)	VAAPI (and others HW plugins)	
Windows	FFMPEG	D3D11	MFX
MSMF	D3D11	-	

Hardware prerequisites

Hardware-accelerated decoding/encoding requires capable hardware.

Intel hardware

You can check H/W support matrix on these resources:

- Intel Media Driver page which provides VAAPI support on Linux
- Wiki page about Intel Quick Sync Video.

AMD hardware

You can check H/W support matrix on these resources:

- Wiki page about AMD Video Core Next

NVIDIA hardware

You can check H/W support matrix on these resources:

- Wiki page about NVIDIA NVDEC
- Wiki page about NVIDIA NVENC
- NVIDIA Video Codec SDK

Installation BKC

Check these resources about installation of media libraries:

- FFmpeg
- GStreamer
- Microsoft Media Foundation (MSMF) runtime is usually already preinstalled on Windows (except some "Base" editions, which are widely used in Docker Windows images)

Installation BKC on Ubuntu 20.04 (Intel CPU with HD Graphics)

Install these packages:

- VAAPI: `apt-get install libva-dev vainfo`
- FFmpeg: `apt-get install ffmpeg libavcodec-dev libavformat-dev libswscale-dev`
- GStreamer and its plugins:

  ```
  apt-get install --no-install-recommends \
    libgstreamer1.0-0 libgstreamer1.0-dev \
    libgstreamer-plugins-base1.0-dev libgstreamer-plugins-bad1.0-dev \
    gstreamer1.0-plugins-base gstreamer1.0-plugins-bad gstreamer1.0-libav
  gstreamer1.0-plugins-good \
    gstreamer1.0-plugins-ugly gstreamer1.0-vaapi gstreamer1.0-tools
  ```

- Media SDK packages: `apt-get install libmfx-dev libmfx-tools`

After installation of the packages above you need to rebuild OpenCV from scratch (clean build directory). You should see these entries in CMake summary log:

```
--   Video I/O:
...
--     FFMPEG:                      YES
--       avcodec:                   YES (58.54.100)
--       avformat:                  YES (58.29.100)
--       avutil:                    YES (56.31.100)
--       swscale:                   YES (5.5.100)
--       avresample:                YES (4.0.0)
--     GStreamer:                   YES (1.16.2)
```

Install full-feature VAAPI driver for Intel hardware:

```
apt-get install intel-media-va-driver-non-free
```

This package installs VAAPI driver with support for both HW decode and encode, and automatically uninstalls package 'intel-media-va-driver' (which supports HW decode only) if was installed previously as dependency of other packages.

Correct installation should output something like this for `vainfo` call (CPU: Intel i5-6600 (Skylake)):

```
libva info: VA-API version 1.7.0
libva info: Trying to open /usr/lib/x86_64-linux-gnu/dri/iHD_drv_video.so
libva info: Found init function __vaDriverInit_1_7
libva info: va_openDriver() returns 0
```

```
vainfo: VA-API version: 1.7 (libva 2.6.0)
vainfo: Driver version: Intel iHD driver for Intel(R) Gen Graphics - 20.1.1 ()
vainfo: Supported profile and entrypoints
      VAProfileNone                   : VAEntrypointVideoProc
      VAProfileNone                   : VAEntrypointStats
      VAProfileMPEG2Simple            : VAEntrypointVLD
      VAProfileMPEG2Simple            : VAEntrypointEncSlice
      VAProfileMPEG2Main              : VAEntrypointVLD
      VAProfileMPEG2Main              : VAEntrypointEncSlice
      VAProfileH264Main               : VAEntrypointVLD
      VAProfileH264Main               : VAEntrypointEncSlice
      VAProfileH264Main               : VAEntrypointFEI
      VAProfileH264Main               : VAEntrypointEncSliceLP
      VAProfileH264High               : VAEntrypointVLD
      VAProfileH264High               : VAEntrypointEncSlice
      VAProfileH264High               : VAEntrypointFEI
      VAProfileH264High               : VAEntrypointEncSliceLP
      VAProfileVC1Simple              : VAEntrypointVLD
      VAProfileVC1Main                : VAEntrypointVLD
      VAProfileVC1Advanced            : VAEntrypointVLD
      VAProfileJPEGBaseline           : VAEntrypointVLD
      VAProfileJPEGBaseline           : VAEntrypointEncPicture
      VAProfileH264ConstrainedBaseline: VAEntrypointVLD
      VAProfileH264ConstrainedBaseline: VAEntrypointEncSlice
      VAProfileH264ConstrainedBaseline: VAEntrypointFEI
      VAProfileH264ConstrainedBaseline: VAEntrypointEncSliceLP
      VAProfileVP8Version0_3          : VAEntrypointVLD
      VAProfileHEVCMain               : VAEntrypointVLD
      VAProfileHEVCMain               : VAEntrypointEncSlice
      VAProfileHEVCMain               : VAEntrypointFEI
```

Note: There are several VAAPI drivers for Intel hardware: `i965` and `iHD`. There is strong recommendation to use `iHD` version (mandatory for modern hardware).

Installation BKC on Windows

Media decoders/encoders runtimes are usually a part of Graphics Drivers Software on Windows.

Dedicated SDKs may be required if you want to rebuild customized versions of FFmpeg/GStreamer.

This section is not complete

Environment variables

Environment variable `OPENCV_FFMPEG_CAPTURE_OPTIONS` allows to experiment with acceleration types other than D3D11VA/VAAPI/MFX in VideoCapture/VideoWriter APIs with FFMPEG backend implementation. For example, to use VAAPI and VDPAU acceleration (in priority order) in VideoCapture, open VideoCapture with parameters '{ CAP_PROP_HW_ACCELERATION, VIDEO_ACCELERATION_ANY }' and set environment variable

```
OPENCV_FFMPEG_CAPTURE_OPTIONS="hw_decoders_any;vaapi,vdpau"
```

To use NVENC/CUDA acceleration in VideoWriter, open VideoWriter with parameters '{ VIDEOWRITER_PROP_HW_ACCELERATION, VIDEO_ACCELERATION_ANY }' and set environment variable

```
OPENCV_FFMPEG_WRITER_OPTIONS="hw_encoders_any;cuda"
```

Acceleration naming in these environment variables follows FFMpeg convension.

Samples and benchmarks

1. samples/tapi/video_acceleration.cpp
2. samples/cpp/videocapture_gstreamer_pipeline.cpp Wiki page
3. :information_source: Media SDK backend Wiki page

Troubleshooting

Before reporting the problem please collect information about:

- used video stream, including information about used codec (see below, use FFmpeg or GStreamer native tools)
- installed packages: `apt list --installed | grep -e va-driver -e mfx -e ffmpeg -e libva -e opencl -e intel-media -e gstreamer -e i965` (use `yum list installed` on Fedora/CentOS/RedHat)
- dump of `vainfo` command

FFmpeg

Run `ffplay` / `ffmpeg` on the media stream with issues. Try to run with or without the `-hwaccel` option.

Use `ffprobe -show_streams <filename>` to extract information about the video stream

GStreamer

Use `gst-launch` utility to check media stream with issues.

Use `GST_DEBUG` environment variable to see extra messages from GStreamer.