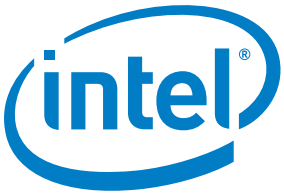

Smart Video Evaluation Toolkit – Sample Application User Guide 2020.1.0

Mar 2020

You may not use or facilitate the use of this document in connection with any infringement or other legal analysis concerning Intel products described herein. You agree to grant Intel a non-exclusive, royalty-free license to any patent claim thereafter drafted which includes subject matter disclosed herein

No license (express or implied, by estoppel or otherwise) to any intellectual property rights is granted by this document.

All information provided here is subject to change without notice. Contact your Intel representative to obtain the latest Intel product specifications and roadmaps.

The products described may contain design defects or errors known as errata which may cause the product to deviate from published specifications. Current characterized errata are available on request.

Copies of documents which have an order number and are referenced in this document may be obtained by calling 1-800-548-4725 or by visiting: http://www.intel.com/design/literature.htm

[Delete this paragraph if your document does not message any Intel® technologies that need anything else in order to work.] Intel technologies' features and benefits depend on system configuration and may require enabled hardware, software or service activation. Learn more at http://www.intel.com/ or from the OEM or retailer.

Intel [include any Intel trademarks which are used in this document] and the Intel logo are trademarks of Intel Corporation in the U.S. and/or other countries.

*Other names and brands may be claimed as the property of others.

Contents

Revision History

Date	Revision	Description
2020/03/03	2.0	Add new example par file. Add usage of parameters in par file
2019/12/26	1.0	Initial release

Note: Releases in the table are listed in reverse order so that the latest/newest is in the top row.

§

1.0 Installation Guide

1.1 System installation

Install Ubuntu 18.04 to a CFL device (e.g. NUC8i7BEH)

Set up the network correctly and run “sudo apt update”

1.2 Install OpenVINO 2019 R3

The sample application video_e2e_sample depends on OpenVINO libraries. We suggest users to install OpenVINO 2019 R3 package from https://software.intel.com/en-us/openvino-toolkit.

Please Install OpenVINO 2019 R3 according to https://docs.openvinotoolkit.org/latest/_docs_install_guides_installing_openvino_linux.html#install-openvino

By default, OpenVINO 2019 R3 is installed to “/opt/intel/openvino”. It also can be installed to ~/intel/openvino. In this case, please replace “/opt/intel/openvino” with “~/intel/openvino” in the following instructions.

Make sure the OpenCL driver is installed correctly by running “sudo /opt/intel/openvino/install_dependencies/install_NEO_OCL_driver.sh”. If you see below error message during the installation of NEO OCL driver:

```
dpkg: dependency problems prevent removal of intel-igc-core:
 intel-igc-opencl depends on intel-igc-core (= 1.0.10-2407).
dpkg: error processing package intel-igc-core (--remove):
 dependency problems - not removing
Errors were encountered while processing:
 intel-igc-core
```

Please uninstall intel-igc-opencl and intel-igc-core manually by bellow commands:

```
sudo dpkg -r intel-igc-opencl
sudo dpkg -r intel-igc-core
```

Then re-run command “sudo /opt/intel/openvino/install_dependencies/install_NEO_OCL_driver.sh”

Run “source /opt/intel/openvino/bin/setupvars.sh” and add “source /opt/intel/openvino/bin/setupvars.sh” to .bashrc under home directory. This step is important because both the building and running of video_e2e_sample can fail if setupvars.sh doesn’t run firstly in the same bash.

1.3 Build SVET sample application and dependent libraries

Run the build_and_install.sh script under the svet_e2e_sample_l. It will install dependent libraries, download and build Media SDK, media-driver, libva and live555. It can take 10 to 20 minutes that depends your network bandwidth. It will ask password for “sudo” command. Please input the “sudo” password to continue the installation.

Please note if libva, media-driver and Media SDK libraries have been installed to /usr/lib/x86_64-linux-gnu/ and /opt/intel/mediasdk/, original version of these libraries will be overwritten. If libva has been installed to /usr/lib or any other path in $LD_LIBRARY_PATH, please uninstall the libraries and header files firstly. Otherwise, Media SDK and media-driver can refer to wrong libva header files or link to wrong libva libraries.

```
$ tar zxvf svet_e2e_sample_l.tar.gz
$ cd svet_e2e_sample_l
$./build_and_install.sh
        [ INFO ] Working directory: /home/work/vaas_e2e_sample_l

        ************************************************************************
        [ INFO ] Install required tools and create build environment.
        ************************************************************************
        Please input the sudo password to proceed

[sudo] password for userxxx:
```

Below table list the detailed steps in build_and_install.sh. If any step fails, user can try to find the corresponding commands and run them manually.

Step Description		**Expected Results**
Check if directory $INTEL_OPENVINO_DIR exists.		Environment variable INTEL_OPENVINO_DIR has been set correctly.
Run ./msdk_pre_install.py	Run apt install to install dependent libraries	apt command runs successfully
	Download libva, libva-util, gmm-lib, media-driver, Media SDK source code for Media SDK 2019 R4 release.	Source code libva, libva-util, gmm-lib, media-driver, MediaSDK are downloaded under directory svet_e2e_sample_l.
	Build and install libva, libva-util, gmm-lib, media-driver	Build and install libva and media-driver libraries to /usr/lib/x86_64-linux-gnu/ successfully.

Apply patches under patch/ to Media SDK and copy directory video_e2e_sample/ to Media SDK/samples/. Then build Media SDK and video_e2e_sample	Binary ./bin/video_e2e_sample (symbol link to ./Media SDK/build/__bin/release/video_e2e_sample) is built successfully. And Media SDK libraries are installed to /opt/intel/mediasdk/
Add libva and Media SDK environment variable setting commands to .bashrc and also run these commands in current bash.	Add bellow commands to ~/.bashrc if they are not added before. vainfo can run successfully export LD_LIBRARY_PATH=$LD_LIBRARY_PATH:/usr/lib/x86_64-linux-gnu:/usr/lib export LIBVA_DRIVERS_PATH=/usr/lib/x86_64-linux-gnu/dri export LIBVA_DRIVER_NAME=iHD export MFX_HOME=/opt/intel/mediasdk LD_LIBRARY_PATH="$LD_LIBRARY_PATH:/opt/intel/mediasdk/lib
Run script/download_and_copy_models.sh to download OpenVINO face detection, human pose estimation and vehicle detection models IR files to directory model/	$ ls model/ face-detection-retail-0004.bin vehicle-attributes-recognition-barrier-0039.bin face-detection-retail-0004.xml vehicle-attributes-recognition-barrier-0039.xml human-pose-estimation-0001.bin vehicle-license-plate-detection-barrier-0106.bin human-pose-estimation-0001.xml vehicle-license-plate-detection-barrier-0106.xml

1.4 Verify SVET sample application's dependency

If build_and_install.sh runs successfully, now run vainfo and you can see below output

```
$ vainfo

error: can't connect to X server!

libva info: VA-API version 1.6.0

libva info: User environment variable requested driver 'iHD'

libva info: Trying to open /usr/lib/x86_64-linux-gnu/dri/iHD_drv_video.so

libva info: Found init function __vaDriverInit_1_6
```

```
libva info: va_openDriver() returns 0
vainfo: VA-API version: 1.6 (libva 2.6.0)
vainfo: Driver version: Intel iHD driver - 19.4.0
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
      VAProfileVP8Version0_3      : VAEntrypointEncSlice
```

```
      VAProfileHEVCMain          : VAEntrypointVLD

      VAProfileHEVCMain          : VAEntrypointEncSlice

      VAProfileHEVCMain          : VAEntrypointFEI

      VAProfileHEVCMain10        : VAEntrypointVLD

      VAProfileHEVCMain10        : VAEntrypointEncSlice

      VAProfileVP9Profile0       : VAEntrypointVLD

      VAProfileVP9Profile2       : VAEntrypointVLD     VAProfileH264High          :
VAEntrypointEncSlice
```

And use below command to check if there are any missing libraries:

```
ldd ./bin/video_e2e_sample  | grep "not found"
```

If there is any library not found, it means the installation wasn't completed. Please contact your account manager from Intel and send the output of above command in email

1.5 Prepare the video clips for testing

If you don't have video clip for testing, you can download sample videos for face detection from https://raw.githubusercontent.com/intel-iot-devkit/sample-videos/master/head-pose-face-detection-male.mp4, human pose estimation from https://github.com/intel-iot-devkit/sample-videos/blob/master/classroom.mp4 and vehicle detection sample video from https://github.com/intel-iot-devkit/sample-videos/blob/master/car-detection.mp4. Since SVET sample application only supports element stream, you can use bellow command to extract the element stream from MP4 file:

```
ffmpeg -i classroom.mp4 -vcodec copy -an -bsf:v h264_mp4toannexb classroom.h264
```

After that, classroom.h264 can be used as input video stream.

2.0 Run SVET sample application

2.1 Check environment variables

Using below commands to check if environment variables LIBVA_DRIVERS_PATH and INTEL_OPENVINO_DIR set correctly.

```
$echo $LIBVA_DRIVERS_PATH

/usr/lib/x86_64-linux-gnu/dri

$echo $INTEL_OPENVINO_DIR

/opt/intel/openvino_2019.3.376
```

2.2 Modify the video path in parameter file

Modify the video path (following "-i::h264") of **every line** in example par file s under face_detection_1080p_16_channel.par. Please use absolute path of testing video clip.

```
-i::h264 /home/work/video/classroom.h264 -join -hw  -async 10 -dec_postproc -threads 2   -o::sink -vpp_comp_dst_x 0 -vpp_comp_dst_y 0 -vpp_comp_dst_w 480 -vpp_comp_dst_h 270 -ext_allocator -infer::fd ./model
```

Otherwise you will see below error message when run the sample application

```
[ERROR], sts=MFX_ERR_NULL_PTR(-2), Init, m_fSource pointer is NULL at /home/work/video_e2e_sample_l/MediaSDK/samples/video_e2e_sample/src/file_and_rtsp_bitstream_reader.cpp:165
```

2.3 Run video_e2e_sample application

Before running the SVET sample application, you must switch ubuntu to text mode by "Ctrl + Alt + F3". And then switch to root user by "su -p" because the DRM direct rendering requires root permission and no X Window. The "-p" option is to keep the current user environment variables settings.

There are many par files under folder par_file. This chapter lists example par files for several typical use cases. Please refer to Chapter 2.4 for the detailed information of parameters in par files.

2.3.1 16-channel video decoding, face detection, composition, encode and display

Command line:

```
./bin/video_e2e_sample -par par_file/n16_1080p_1080p_dp.par
```

The face detection inference is specified by "-infer::fd ./model" in the par file. " ./model " is the directory that stores face detection model IR files.

The loading of face detection models to GPU is slow and you might need to wait for a minute until the video showing on display as below screenshot. The loading speed issue has some solution in OpenVINO 2020 R1 release. SVET sample application will upgrade to that version in R2 release.

If you want to stop the application, press "Ctrl + c" in the bash shell.

If you want to play 200 frames in each decoding session, you can append "-n 200" to parameters lines starting with "-i" in par files.

2.3.2 4-channel video decoding, human pose estimation, composition, and display

Command line:

```
./bin/video_e2e_sample -par par_file/n4_1080p_1080p_dp.par
```

The face detection inference is specified by "-infer::hp ./model" in the par file. " ./model " is the directory that stores human pose estimation model IR files.

Below picture is the screenshot of this demo.

2.3.3 4-channel video decoding, vehicle and vehicle attributes detection, composition, encode and display

Command line:

```
./bin/video_e2e_sample -par par_file/n4_ vehical_detect_1080p.par
```

The vehicle and vehicle attributes detection inference is specified by "-infer::vd ./model" in the par file. " ./model " is the directory that stores vehicle and vehicle attributes detection model IR files.

Below picture is the screenshot of this demo.

2.3.4 16-channel RTSP video decoding, face detection, composition, encode and display

Command line:

```
./bin/video_e2e_sample -par par_file/n16_1080p_rtsp_simu.par
```

To use RTSP video stream instead of local video file, you can modify the par file and use RTSP URL to replace local video file path.

```
-i::h264 rtsp://192.168.0.8:1554/simu0000  -join -hw -async 4 -dec_postproc  -o::sink -vpp_comp_dst_x 0 -vpp_comp_dst_y 0 -vpp_comp_dst_w 480 -vpp_comp_dst_h 270 -ext_allocator -infer::fd ./model
```

2.3.5 Offline inference mode

The results of inference are rendered to the composition by default. It can be disabled by add parameter "-infer::offline" after "-infer::fd ./model", then the result of inference won't be rendered.

2.3.6 16-channel RTSP video decoding, RTSP stream storing, face detection, composition, encode, and display

Command line:

```
./bin/video_e2e_sample -par par_file/n16_1080p_rtsp_simu_dump.par
```

The name of RTSP streaming local file is specified by option "-rtsp_save filename" in decoding session in par file. User can choose one or more sessions to invoke the RTSP stream storing

2.3.7 2-channel RTSP stream storing

Command line:

```
./bin/video_e2e_sample -par par_file/rtsp_dump_only.par
```

When there are only "-i" and "-rtsp_save" options in par file, the session won't run decode or inference or display but only save the specified RTSP stream to local file.

Please note, such sessions must be put into one separated par file. If you'd like to run RTSP stream storing sessions together with other decoding and inference sessions, you can run with two par files. For example

Command line:

```
./bin/video_e2e_sample -par par_file/rtsp_dump_only.par par_file/n16_1080p_rtsp_simu.par
```

2.3.8 Multiple displays

Below is an example to run 16 1080p decode sessions on one display and run 4 1080p decode and inference sessions on another display.

Please note: if the two par files specify different resolutions for display, e.g. 1080p and 4k, and there is one 1080p and one 4k monitors connects to the device, this command line could run into error due to 4k par file selecting 1080p monitor, in this case, you can try to switch the order of par files passed to video_e2e_sample. In current implementation, "-rdrm-XXXX" options are ignored. Sample application will choose the first unused display emulated from the DRM for each par file. The order is according to the CRTC id showed in "/sys/kernel/debug/dri/0/i915_display_info". Display with smaller CRTC id is emulated earlier. Generally, the first par file in the command can get the display with smallest CRTC id. But since we create different thread for each par file, the actual order of display assigned to each par file may not be strictly the same as the order of par file in the command.

Command line:

```
./bin/video_e2e_sample -par par_file/n16_1080p_1080p_dp_noinfer.par
par_file/n4_1080p_1080p_dp.par
```

2.4 Usage of media codec, inference and display parameters in par file

2.4.1 New parameters in Par file

Comparing to original sample_multi_transcode, we add some new parameters.

Parameter	Usage
-infer::infer_type ir_file_dir	Specify the inference type and directory that stores the IR files. Can be used together with -infer::offline. Examples: -infer::fd ./model →face detection -infer::hp ./model →human pose estimation -infer:vd ./model →vehicle and vehicle attributes detection -infer::fd ./model -infer::offline →face detection but not render the results to display
-i::h264 rtsp://url	Specify the source H264 file with RTSP URL
-rtsp_save filename.h264	Save RTSP stream to local file. This parameter must be used together with "-i::h264 rtsp://url".

	If the whole line of session parameters only contains "-i::h264 rtsp://url -rtsp_save filename.h264" and don't have other decoding parameters, we call such sessions as RTSP stream storing session and they must be put into a separated par file.

2.4.2 Decode, encode and display parameters

Below table explains the parameters used in example par files. The full parameter list can also be found at https://github.com/Intel-Media-SDK/MediaSDK/blob/master/doc/samples/readme-multi-transcode_linux.md

Parameter	Usage
-i::h264 | h264 input_video_filename	Set input file and decoder type
-o::h264 | h265 output_video_filename	Set output file and decoder type
-o::sink	The output will be passed to the sink sessions,, e.g. encoding session or composition session
-i::source	The input is coming from source sessions like decoding session
-dec_postproc	Resize after decoder using direct pipe (should be used in decoder session)
-vpp_comp_dst_x 0 -vpp_comp_dst_y 270 -vpp_comp_dst_w 480 -vpp_comp_dst_h 270	(x, y) position and size of this stream in composed stream
-join	Join session with other session(s). If there are several transcoding sessions, any number of sessions can be joined. Each session includes decoding, preprocessing (optional), and encoding
-hw	GPU will be used for HW accelerated video decoding, encoding and post-processing.
-async <async_depth>	Depth of asynchronous pipeline.
-threads <thread_number>	Number of session internal threads to create
-ext_allocator	Force usage of external allocators
-n	Number of frames to transcode (session ends after this number of frames is reached). In decoding sessions (-o::sink) this parameter limits number of frames acquired from decoder. In encoding sessions (-o::source) and transcoding sessions this parameter limits number of frames sent to encoder.
-fps <fps>	Transcoding frame rate limit

-vpp_comp <sourcesNum>	Enables composition from several decoding sessions. Result is written to the file
-vpp_comp_only <sourcesNum>	Enables composition from several decoding sessions. Result is shown on screen.
-ec::nv12 | rgb4	Forces encoder input to use provided chroma mode.
-rdrm-DisplayPort	Using drm direct rendering. 'DisplayPort' will be ignored. The sample application will try to use the first DP or HDMI display it can connect to.