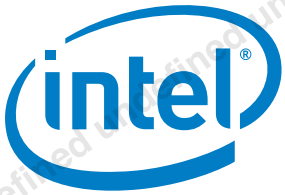


# Smart Video Evaluation Toolkit – Linux* Sample Application User Guide

**Application Note**

***August 2020***

Document Number: 630748-1.0

# Contents



## Figures



## Tables

# Revision History

| Date | Revision | Description |
|---|---|---|
| August 2020 | 1.0 | Initial release. |

§

# 1.0 Introduction

## 1.1 Terminology

**Table 1. Terminology**

| Term | Description |
|---|---|
| SVET | Smart Video Evaluation Toolkit |
| Intel® VA-API | Intel® Video Acceleration API |
| iGPU | Integrated graphic process unit |
| i915 | Intel® graphic Linux kernel DRM driver that supports Intel iGPU |
| iHD driver | Intel® VAAPI backend media driver for Intel iGPU |
| Gmmlib | The Intel® Graphics Memory Management Library provides device specific and buffer management for the Intel® Graphics Compute Runtime for OpenCL™ and the Intel® Media Driver for VAAPI. |
| Intel® Media SDK | Intel® Media Software Development Kit |
| DRI | Direct Rendering Infrastructure |
| DRM | Direct Rendering Management |
| PCI | Peripheral Component Interconnect |

## 1.2 Background

Smart Video Evaluation Toolkit (SVET) can support users to quickly setup and adjust the core concurrent video analysis workload through configuration file to obtain the best performance of video codec, post-processing and inference based on Intel® integrated GPU according to their product requirements. SVET contains two key components, development-guide-for-high-density-video-analytics-workloads-on-ia-component and concurrent video analytics sample application. The SVET Linux sample application can be found at https://github.com/intel-iot-devkit/concurrent-video-analytic-pipeline-optimization-sample-l and the SVET Windows sample application can be found at https://github.com/intel-iot-devkit/concurrent-video-analytic-pipeline-optimization-sample-w. Users can use the sample application for runtime performance evaluation or as a reference for debugging core video workload issues.

This document describes how to download, build, and run the SVET Linux sample application.

## 1.3 Limitations

The instructions in this document are only verified on 6th, 7th, 8th, 9th, and 10th generation Intel Core processors installed with Ubuntu* 18.04. If you're using older Intel processors, the latest Libva and media driver won't work. Please refer to "Known Issues and Limitations" on webpage https://github.com/intel/media-driver.

If you're using other Linux distributions, like CentOS*, Fedora*, and Yocto*, the example building and installation instructions won't work, and you need to change the installation path accordingly.

# 2.0 Before the Start

## 2.1.1 What is SVET Linux Sample Application

Figure 1 shows SVET Linux sample application pipeline structure. It reads pipeline parameters from par file. Use can use par file to configurate the parameters of video decoding, video resize, inference, render results, composition, display and video encoding.

The pipeline contains multiple sessions. Each session is defined by one line in par file. The session can be source or sink. The source session is decoding session and defined by lines starting with "-i". The sink session can be encoding session that is defined "-vpp_com", display session "-vpp_comp_only" or fake sink session "-fake_sink". The source sessions add the decoded surfaces to the shared buffer queue while the sink sessions take the surfaces from shared buffer queue and release them when complete processing.

**Figure 1. SVET Sample Application Pipeline**


## 2.1.2 Where to Download

SVET Linux sample application can be download by the command below:

```
$git clone https://github.com/intel-iot-devkit/concurrent-video-analytic-pipeline-optimization-sample-l.git svet_sample
```

# 3.0 Installation Guide

## 3.1 System Installation

Install Ubuntu 18.04.02 to a device that has Intel 6th generation or newer Core processors

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

### 3.1.1 Tiger Lake U: Upgrade Linux Kernel and iGPU Firmware

You can skip this chapter if you’re using Sky Lake, Coffee Lake, or Whiskey Lake U whose iGPU generation is 9.

To find out the GPU generation number, please run the command below with root permission.

```
$cat /sys/kernel/debug/dri/0/i915_capabilities  | grep gen

gen: 9
```

On Tiger Like U, the iGPU generation is 12.

### 3.1.2 Tiger Lake U: Upgrade Linux Kernel

***Note:*** Please back up your private files before upgrading Linux kernel.

Firstly, check if the Linux kernel has been upgraded to 5.0 or above:

```
$uname -a

Linux NUC 5.3.0-53-generic #47~18.04.1-Ubuntu SMP Thu May 7 13:10:50 UTC 2020 x86_64 x86_64
x86_64 GNU/Linux
```

If not, please use below command to upgrade to kernel 5.0 and reboot the device.

```
$sudo apt-get install --install-recommends linux-generic-hwe-18.04 xserver-xorg-hwe-18.04
```

Then run the commands below to download and install Yocto Linux kernel for TGL

```
$ wget https://github.com/intel/linux-intel-lts/archive/lts-v5.4.39-yocto- 200513T032237Z.tar.gz

$ tar -xzvf lts-v5.4.39-yocto-200513T032237Z.tar.gz

$ cd linux-intel-lts-lts-v5.4.39-yocto-200513T032237Z

$ cp /boot/config-5.0.0-23-generic .config   //Copy Ubuntu default kernel config file
```

```
$ make oldconfig //Select the default value for unset config items

$ make -j8

$ make modules_install

$ make install
```

Then edit the Linux kernel boot option to force GPU module probe

*$* vi /etc/default/grub

```
# If you change this file, run 'update-grub' afterwards to update
# /boot/grub/grub.cfg.
# For full documentation of the options in this file, see:
#   info -f grub -n 'Simple configuration'

GRUB_DEFAULT=0
GRUB_TIMEOUT_STYLE=hidden
GRUB_TIMEOUT=0
GRUB_DISTRIBUTOR=`lsb_release -i -s 2> /dev/null || echo Debian`
GRUB_CMDLINE_LINUX_DEFAULT="quiet splash i915.force_probe=* i915.enable_guc=2"
GRUB_CMDLINE_LINUX=""
```

*$ update-grub*

Please install GPU firmware by the command below:

```
$ wget https://git.kernel.org/pub/scm/linux/kernel/git/firmware/linux-firmware.git/plain/i915/tgl_guc_35.2.0.bin

$ wget https://git.kernel.org/pub/scm/linux/kernel/git/firmware/linux-firmware.git/plain/i915/tgl_huc_7.0.12.bin

$ wget https://git.kernel.org/pub/scm/linux/kernel/git/firmware/linux-firmware.git/plain/i915/tgl_dmc_ver2_04.bin

$ cp *.bin /lib/firmware/i915
```

After rebooting, please use the command below to confirm the kernel upgrade and GPU firmware:

```
$uname -a // Confirm new kernel version after reboot

$ cat /sys/kernel/debug/dri/0/i915_gpu_info //Confirm GPU firmware loaded successfully
```

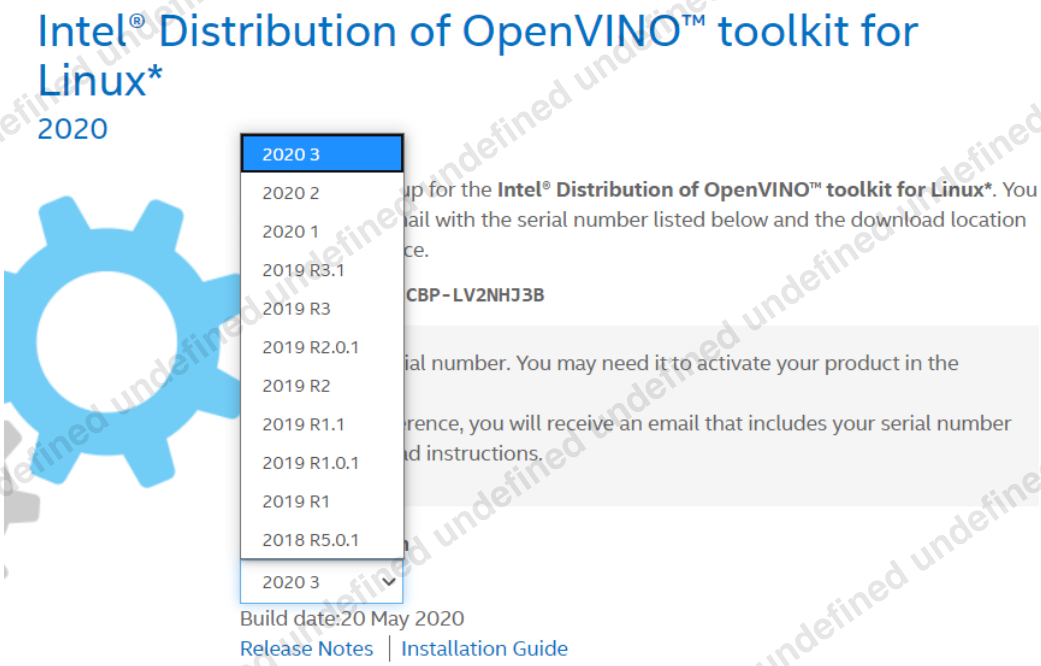
```
GuC firmware: i915/tgl_guc_43.0.1.bin
        status: RUNNING
        version: wanted 43.0, found 43.0
        uCode: 320064 bytes
        RSA: 256 bytes
HuC firmware: i915/tgl_huc_7.0.12.bin
        status: RUNNING
        version: wanted 7.0, found 7.0
        uCode: 529984 bytes
        RSA: 256 bytes
```

## 3.2 Install OpenVINO* 2020.3

The sample application video_e2e_sample depends on OpenVINO libraries. We suggest users to install OpenVINO 2020.3 Linux package from *https://software.intel.com/en-us/openvino-toolkit*

When you choose the OpenVINO Linux package, the latest version is selected by default. Please manually choose the version 2020.3 as below picture:

**Figure 2. Select OpenVINO 2020.3 in Download Page**




Please Install OpenVINO 2020.3 according to https://docs.openvinotoolkit.org/latest/_docs_install_guides_installing_openvino_linux.html#install-openvino

By default, OpenVINO 2020.3 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.

If you're not using Tiger Lake U, make sure the OpenCL* driver is installed correctly by running "sudo /opt/intel/openvino/install_dependencies/install_NEO_OCL_driver.sh". If you are using Tiger Lake U, please following instructions in Chapter 1.3.1 to install OpenCL* NEO driver manually.

If you see the error message below 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 try to uninstall intel-igc-opencl and intel-igc-core manually by the commands below:

```
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.

### 3.2.1 Tiger Lake U: Install OpenCL* NEO Driver 20.20.16837

If you’re using Tiger Lake U, The OpenCL NEO driver needs to be installed manually.

Please follow the commands below:

```
$mkdir neo_20.20.16837
$cd neo_20.20.16837
$ wget https://github.com/intel/compute-runtime/releases/download/20.20.16837/intel-gmmlib_20.1.1_amd64.deb
$wget https://github.com/intel/compute-runtime/releases/download/20.20.16837/intel-igc-core_1.0.3977_amd64.deb
$wget https://github.com/intel/compute-runtime/releases/download/20.20.16837/intel-igc-opencl_1.0.3977_amd64.deb
$wget https://github.com/intel/compute-runtime/releases/download/20.20.16837/intel-opencl_20.20.16837_amd64.deb
```

```
$wget https://github.com/intel/compute-runtime/releases/download/20.20.16837/intel-ocloc_20.20.16837_amd64.deb

$wget https://github.com/intel/compute-runtime/releases/download/20.20.16837/intel-level-zero-gpu_0.8.16837_amd64.deb

$sudo dpkg -I *.deb  //Please uninstall the old NEO drivers if any

$sudo apt install clinfo

$clinfo  //Check if OpenCL NEO driver was installed successfully
```

## 3.3 Build Concurrent Video Analytic Sample Application and Dependent Libraries

Download the source code and run the build_and_install.sh script with the commands below:

```
$git clone https://github.com/intel-iot-devkit/concurrent-video-analytic-pipeline-optimization-sample-l.git cva_sample
$ cd cva_sample
$./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:
```

It will install dependent libraries, download and build Media SDK, media-driver, libva and libva-util. 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.

The table below 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 $INTEL_OPENVINO_DIR exists. | | Environment variable INTEL_OPENVINO_DIR has been set correctly. |
| Run ./msdk_pre_install.py | Run apt install to install dependent libraries | apt command runs successfully |
| | Download libva, libva-util, gmm-lib, media-driver, Media SDK source code for Media SDK 2020.1.1 release. | Source code libva, libva-util, gmm-lib, media-driver, MediaSDK are downloaded into currently directory. |
| | Build and install libva, libva-util, gmm-lib, media-driver | Build and install libva and media-driver libraries to /usr/lib/x86_64-linux-gnu/ successfully. |
| Apply patches under patch/ to Media SDK and copy directory video_e2e_sample/ to Media SDK/samples/. Then build Media SDK and video_e2e_sample | | Binary ./bin/video_e2e_sample (symbol link to ./Media SDK/build/__bin/release/video_e2e_sample) is built successfully. And Media SDK libraries are installed to /opt/intel/mediasdk/ |
| Add libva and Media SDK environment variable setting commands to .bashrc and also run these commands in current bash. | | Add the commands below to ~/.bashrc if they are not added before.<br><br>vainfo can run successfully<br><br>export LD_LIBRARY_PATH=$LD_LIBRARY_PATH:/usr/lib/x86_64-linux-gnu:/usr/lib<br><br>export LIBVA_DRIVERS_PATH=/usr/lib/x86_64-linux-gnu/dri<br><br>export LIBVA_DRIVER_NAME=iHD<br><br>export MFX_HOME=/opt/intel/mediasdk<br><br>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/<br><br>face-detection-retail-0004.bin vehicle-attributes-recognition-barrier-0039.bin<br><br>face-detection-retail-0004.xml vehicle-attributes-recognition-barrier-0039.xml<br><br>human-pose-estimation-0001.bin vehicle-license-plate-detection-barrier-0106.bin<br><br>human-pose-estimation-0001.xml vehicle-license-plate-detection-barrier-0106.xml |

## 3.4 Verify Sample Application's Dependencies

If build_and_install.sh runs successfully, now run Vainfo and you can see the output below:

```
$ vainfo
error: can't connect to X server!
libva info: VA-API version 1.8.0
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_7
libva info: va_openDriver() returns 0
vainfo: VA-API version: 1.8 (libva 2.7.1)
vainfo: Driver version: Intel iHD driver for Intel(R) Gen Graphics - 20.1.1 (26a1c10)
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....
```

And use the command below 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

## 3.5 Prepare 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 this sample application only supports element stream, you can use the command below 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.

# 4.0 Run Sample Application video_e2e_sample

## 4.1 Check Environment Variables

Using the commands below 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
```

## 4.2 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 the error message below when running 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
```

## 4.3 Enable cl_cache

Loading of inference models can take long time. It's recommended to enable OpenCL kernel cache. By default, script build_and_install.sh adds command "mkdir ~/cl_cache" and "export cl_cache_dir=~/cl_cache" to .bashrc.  So the cl_cache is enabled after running script build_and_install.sh.  You can use command "echo $cl_cache_dir" to confirm cl_cache is enabled in current bash terminal.

It's recommended to clear directory $cl_cache_dir when you upgrade OpenVINO in the future.

For cl_cache details, please refer to https://github.com/intel/compute-runtime/blob/master/opencl/doc/FAQ.md

## 4.4 Run video_e2e_sample Application

Before running video_e2_sample with "-rdrm-DisplayPort" in par file, 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 clients running. If there is alive VNC sessions, please close them firstly. The "-p" option is to keep the current user environment variables settings.

If user wants to run video_e2_sample with normal user or with X11 display, user can replace "-rdrm-DisplayPort" with "-rx11". See par_file/inference/n16_face_detection_1080p_x11.par for inference. Note, X11 rendering isn't as efficient as DRM direct rendering. According to our 16-channel face detection test on Coffee Lake, the average time cost of processing one frame increased by 6ms compared to using DRM direct rendering.

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.

### 4.4.1 16-channel Video Decoding, Face detection, Composition, Encode and Display

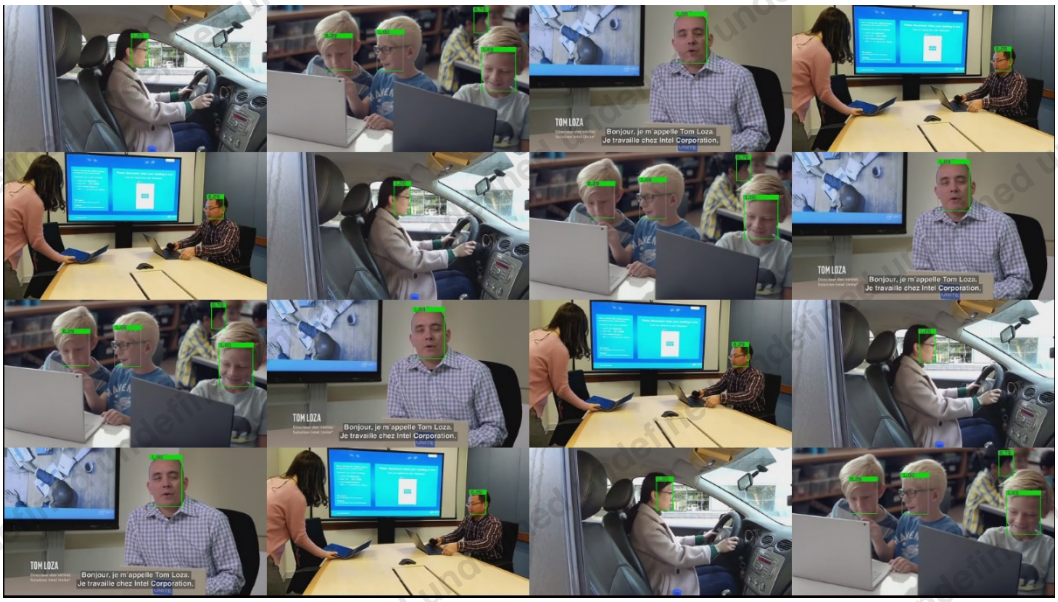
Command line:

```
./bin/video_e2e_sample -par par_file/inference/n16_face_detection_1080p.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 first 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. Then with cl_cache enabled, the next running of face detection models will be much faster, which is about 10 seconds on CFL.

**Figure 3. 16-channel 1080p Video Decoding and Face Detection**



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.

By default, the pipeline is running as fast as it can. If you want to limit the FPS to a certain number, please add “-fps FPS_number” to every decoding session, which start with “-i” in par files. Please refer to par_file/inference/ n16_1080p_face_detect_30fps.par.

### 4.4.2 4-channel Video Decoding, Human Pose Estimation, Composition, and Display

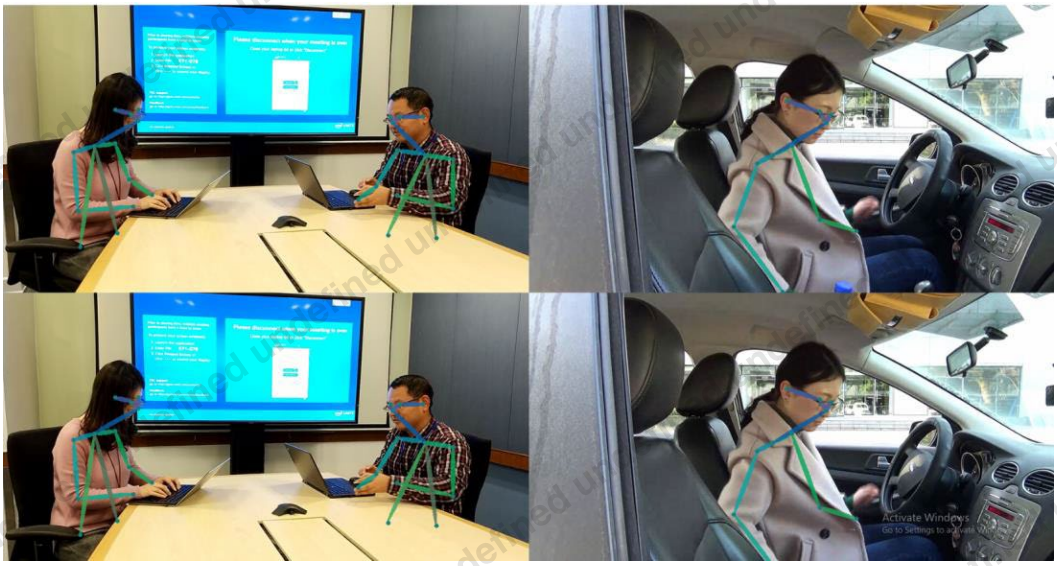
Command line:

```
./bin/video_e2e_sample -par par_file/inference/n4_human_pose_1080p.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.

The picture below is the screenshot of this demo.

Figure 4. 4-channel 1080p Video Decoding and Human Pose Estimation



### 4.4.3 4-channel Video Decoding, Vehicle and Vehicle Attributes Detection, Composition, Encode, and Display

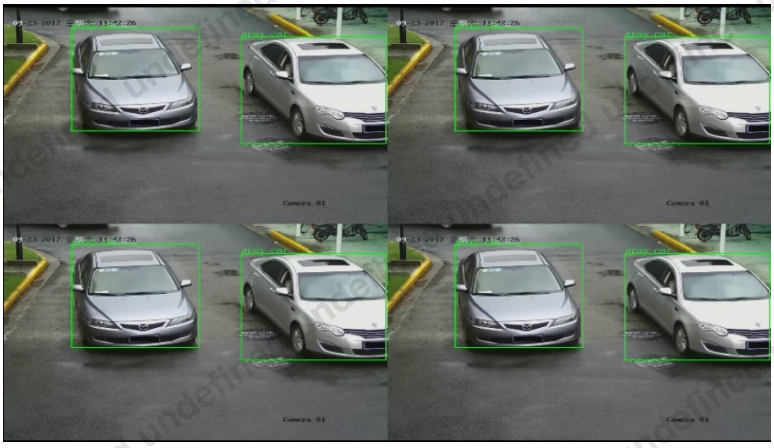
Command line:

```
./bin/video_e2e_sample -par par_file/inference/n4_vehicel_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.

The picture below is the screenshot of this demo.

Figure 5. 4-channel 1080p Video Decoding plus Vehicle and Attributes Detection



### 4.4.4 16-channel RTSP Video Decoding, Face Detection, Composition, Encode, and Display

Command line:

```
./bin/video_e2e_sample -par par_file/rtsp/n16_face_detection_1080p.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
```

### 4.4.5 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.

### 4.4.6 16-channel RTSP Video Decoding, RTSP Stream Storing, Face Detection, Composition, Encode, and Display

Command line:

```
./bin/video_e2e_sample -par par_file/rtsp/n16_face_detection_rtsp_save.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.

### 4.4.7 2-channel RTSP Stream Storing

Command line:

```
./bin/video_e2e_sample -par par_file/rtsp/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/rtsp_dump_only.par
par_file/rtsp/n16_face_detection_rtsp_save.par
```

### 4.4.8 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, for example 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/basic/n16_1080p_30fps_videowall.par par_file/basic/
n16_1080p_30fps_videowall.par
```

### 4.4.9 Use Fake Sink

By using option "-fake_sink", user can run the concurrent video decoding with fake sink instead of display or encoder. In this mode, the composition of decoding or inference result is disabled. Please refer to example par files n16_1080p_decode_fakesink.par under folder par_file/misc and n16_1080p_face_detection_fakesink.par under folder par_file/inference.

### 4.4.10 Use VPP Instead of SFC in Decoding Session

H265 decoder doesn't support SFC, so VPP is used for scaling and color format convert in video decoding sessions. Please refer to example par file n16_1080p_h265_fd.par under folder par_file/inference and n16_h265_1080p_rtsp_simu.par under folder par_file/rtsp.

### 4.4.11 Configurate the Inference Target Device, Inference Interval, and Maximum Object Number

By default, GPU is used as inference target device. User can also use option "-infer::device HDDL" to specify HDDL as target device. Please make sure the HDDL device has been set up successfully. See n4_vehicel_detect_hddl_1080p.par for inference.

The option "-infer::interval" indicates the distance between two inference frames. For example, "-infer::interval 3" means frame 1, 4, 7, 10... will be sent to inference device and other frames will be skipped. For face detection and human pose estimation, the default interval is 6. For vehicle detection, the default interval is 1, which means running inference on every frame.

The option "-infer::max_detect" indicates the maximum number of detected objects for further classification or labeling. By default, there is no limitation of the number of detected objects.
Please refer to example par file n1_infer_options.par.

### 4.4.12 Configurate the Interval of JPEG Encoding

By using option "-frameskip", user can specify interval for H264 to JPEG transcoding. See par_file/basic/n1_jpeg_enc_test.par and par_file/basic/n4_jpeg_enc_test.par.

## 4.5 Usage of Media Codec, Inference, and Display Parameters in Par File

### 4.5.1 New Parameters in Par File

Comparing to original video transcoding application 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.<br><br>Examples:<br><br>-infer::fd ./model →face detection<br><br>-infer::hp ./model →human pose estimation<br><br>-infer:vd ./model →vehicle and vehicle attributes detection<br><br>-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. |
| -dc::rgb4 | Use VPP instead of SFC for scaling and color format conversion in decoding sessions. This option can't be used together with -dec_postproc. Please refer to n16_1080p_h265_fd.par and n16_h265_1080p_rtsp_simu.par. |
| -fake_sink <number of sources> | Use a fake sink instead of display(-vpp_comp) or encoding(-vpp_comp_only). This fake sink won't do composition of sources. The number of sources must be equal the number of decoding sessions. See n16_1080p_decode_fakesink.par and n16_1080p_infer_fd_fakesink.par for example. Please note, "-o" option must be used together with this option, but it won't generate any output file. |
| -infer::device <GPU, HDDL> | Indicate the inference target device. Please refer to example par file n1_infer_options.par. |
| -infer::interval <number> | Indicate the distance between two inference frames. Please refer to example par file n1_infer_options.par. |
| -infer::max_detect <number> | indicates the maximum number of detected objects for further classification or labeling. By default, there is no limitation of the number of detected objects.<br><br>Please refer to example par file n1_infer_options.par. |
| -frameskip interval | This option is only used in H264/H265 to JPEG transcoding. It's used to specify the interval of JPEG encoding. For example, with "-frameskip 5", on video frame will be encoded to JPEG every 5 frames. See par_file/basic/n1_jpeg_enc_test.par and par_file/basic/n4_jpeg_enc_test.par |

### 4.5.2 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<br><br>(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. Please switch Ubuntu to text mode (Ctrl + Alt + F3) and root user by command "su -p" before using this parameter. |
| -rx11 | Using X11 as display. Please make sure environment variable DISPLAY set correctly if run the sample application remotely in a console terminal. |

# 5.0 Pack video_e2e_sample Binaries and Install on Another Device

## 5.1 Pack video_e2e_sample Binaries

After install_and_build.sh script running successfully, user can use script/ pack_binary.sh to copy video_e2e_sample and other dependent binaries into a folder.

Run below command under the source code directory and all video_e2e_sample and other dependent binaries will be copied to a folder named “cva_e2e_sample_l”.

```
$./script/pack_binary.sh

$ls cva_e2e_sample_l/

download_models.sh  libva       media-driver  par_file              video_e2e_sample

install_binary.sh   libva-utils MediaSDK      run_face_detection_test.sh
```

## 5.2 Install video_e2e_sample Binaries

Please make sure the new device has been installed Ubuntu* 18.04 and OpenVINO* 2020.3. And the OpenVINO must be installed to the same path as the device that the video_e2e_sample binaries were built on.

User can copy the folder “cva_e2e_sample_l” to the new device and run “sudo -E ./ install_binary.sh” under folder “cva_e2e_sample_l”. Then video_e2e_sample, libva, media-driver and Media SDK binaries will be installed. The script install_binary.sh also set environment variables LIBVA_DRIVERS_PATH, LIBVA_DRIVER_NAME and LD_LIBRARY_PATH variables with proper values.

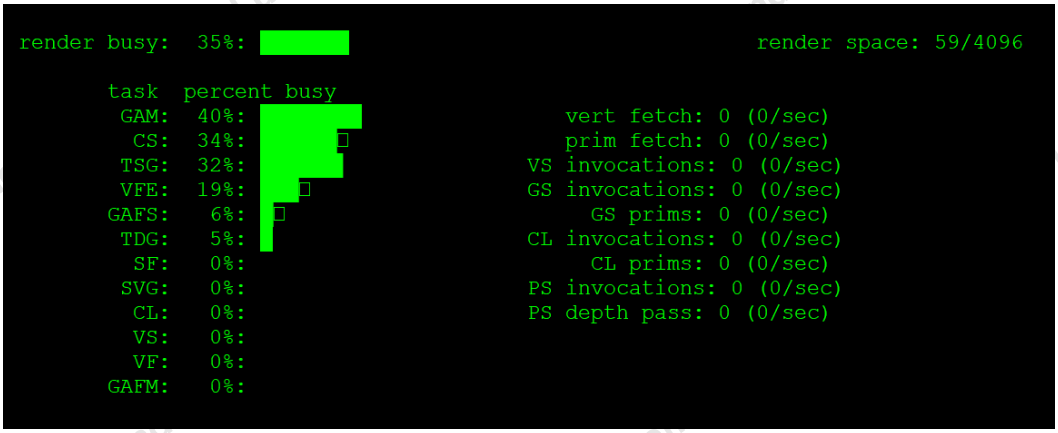
After running install_binary.sh successfully, the user can follow instructions in Chapter 2 to run video_e2e_sample application with par files.

# 6.0 Monitor Overall GPU Resource Usage Statistics

There are some tools can be used to view GPU resource usage statistics. Please also refer to chapter 3.1.4 white paper CDI#621636

## 6.1 Intel_gpu_top

To install intel_gpu_top, run command “sudo apt install intel-gpu-tools”. Then run it with command “sudo intel_gpu_top”




## 6.2 Intel-telemetry-tool

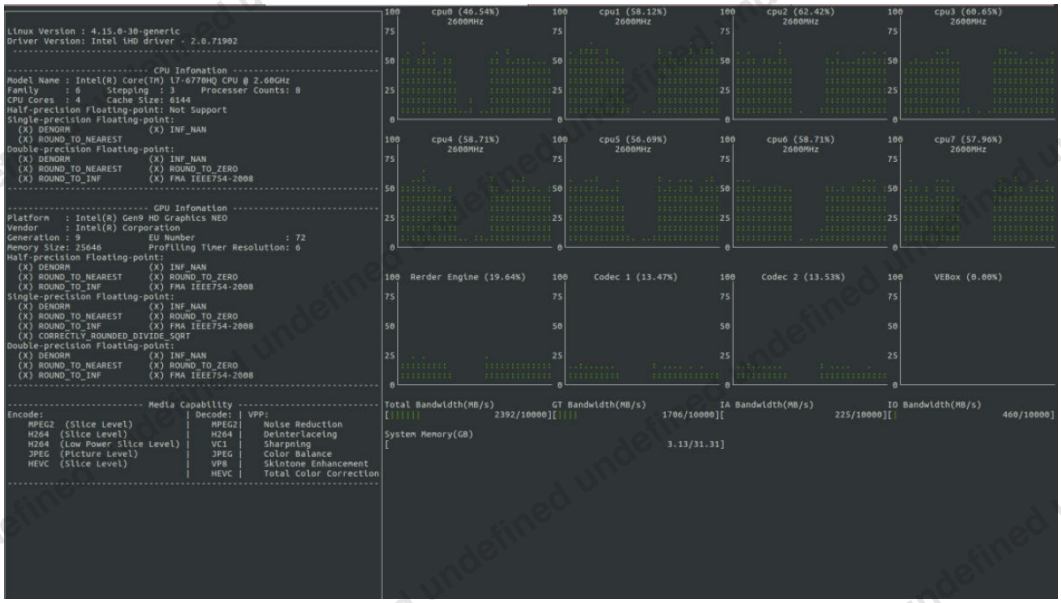
Intel-telemetry-tool is another open-source tool to monitor system resource utilization. It leverages some information from /proc & /sys file system to get static and run-time information. Compared to intel_gpu_top, it can monitor more sub-components such as VDBox and VEBox. User can toggle “s” to show static system information on the left. See intel-telemetry-tool for More details.

To download and run this tool, please refer to the commands below:

```
$git clone https://github.com/Xiaogang-Li/intel-telemetry-tool.git

$cd intel-telemetry-tool

$./build.sh

$sudo -E ./build/tool/telemetry
```

***Note:*** To view GPU resource utilization, user must use “sudo” to run this tool. Please wait one minute if there is no GPU resource usage statistics showing on screen.

Here is a screenshot of intel-telemetry-tool:


Linux Version : 4.15.0-30-generic
Driver Version: Intel iHD driver - 2.0.71902
CPU Infomation
Model Name : Intel(R) Core(TM) i7-6770HQ CPU @ 2.60GHz
Family : 6 Stepping : 3 Processer Counts: 8
CPU Cores : 4 Cache Size: 6144
Half-precision Floating-point: Not Support
Single-precision Floating-point:
(X) DENORM (X) INF_NAN
(X) ROUND_TO_NEAREST
Double-precision Floating-point:
(X) DENORM (X) INF_NAN
(X) ROUND_TO_NEAREST (X) ROUND_TO_ZERO
(X) ROUND_TO_INF (X) FMA IEEE754-2008
GPU Infomation
Platform : Intel(R) Gen9 HD Graphics NEO
Vendor : Intel(R) Corporation
Generation : 9 EU Number : 72
Memory Size: 25646 Profiling Timer Resolution: 6
Half-precision Floating-point:
(X) DENORM (X) INF_NAN
(X) ROUND_TO_NEAREST (X) ROUND_TO_ZERO
(X) ROUND_TO_INF (X) FMA IEEE754-2008
Single-precision Floating-point:
(X) DENORM (X) INF_NAN
(X) ROUND_TO_NEAREST (X) ROUND_TO_ZERO
(X) ROUND_TO_INF (X) FMA IEEE754-2008
(X) CORRECTLY_ROUNDED_DIVIDE_SQRT
Double-precision Floating-point:
(X) DENORM (X) INF_NAN
(X) ROUND_TO_NEAREST (X) ROUND_TO_ZERO
(X) ROUND_TO_INF (X) FMA IEEE754-2008
Media Capability
Encode:
MPEG2 (Slice Level)
H264 (Slice Level)
H264 (Low Power Slice Level)
JPEG (Picture Level)
HEVC (Slice Level)
Decode:
MPEG2
H264
VC1
JPEG
VP8
HEVC
VPP:
Noise Reduction
Deinterlacing
Sharpning
Color Balance
Skintone Enhancement
Total Color Correction
cpu0 (46.54%)
2600MHz
cpu1 (58.12%)
2600MHz
cpu2 (62.42%)
2600MHz
cpu3 (60.65%)
2600MHz
cpu4 (58.71%)
2600MHz
cpu5 (56.69%)
2600MHz
cpu6 (58.71%)
2600MHz
cpu7 (57.96%)
2600MHz
Rerder Engine (19.64%)
Codec 1 (13.47%)
Codec 2 (13.53%)
VEBox (0.00%)
Total Bandwidth(MB/s)
2392/10000
GT Bandwidth(MB/s)
1706/10000
IA Bandwidth(MB/s)
225/10000
IO Bandwidth(MB/s)
460/10000
System Memory(GB)
3.13/31.31