Smart Video Evaluation Toolkit – Linux* Concurrent Video Analytic Sample

Application User Guide

June 2021

Document Number: 630748-4.0

Contents

Tables

Revision History

Date	Revision	Description
2021/05/13	4.0	1. Separated the installation guideline chapters for Tiger Lake and other platforms. 2. Added additional explanation on how to use this document at the beginning.
2021/03/24	4.0	1. Added descriptions for R4 new features (MOT) 2. Updated build script that installed MediaSDK can coexist with the previous media stack version installed in the same computer
2020/09/23	3.0	1. Updated the OpenVINO and Media SDK version 2. Added descriptions for R3 new features
2020/05/19	2.0	1. Updated the OpenVINO and Media SDK version 2. Added descriptions for R2 new features
2020/03/03	1.0	1. Added new example par files 2. Added tables to explain parameters usage in par file
2019/12/26	0.5	Initial release

§

1.0 Installation Guide

In this chapter, you will learn how to install the **video_e2e_sample** application on Ubuntu* 18.04.04.

First, check the integrated GPU generation number with below command:

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

gen: 9
```

If the CPU is Sky Lake, Coffee Lake or Whiskey Lake U, the output is `gen:9`. You can refer to Chapter 1.2 for OpenVINO™ installation.

If the CPU is Tiger Lake, the output is `gen:12`. You can refer to Chapter 1.3 for Linux kernel upgrade and NEO driver upgrade.

If you are not going to use inference, but only video decode and encode, you can skip OpenVINO™ installation, and refer to chapter 2.4.17 on steps to build the sample application without OpenVINO™.

1.1 System installation

Install Ubuntu* 18.04.04 to and make sure the CPU has integrated graphic. You can search the CPU model name on ark.intel.com and check if *Processor Graphic* is included.

Set up the network correctly and run *sudo apt update*.

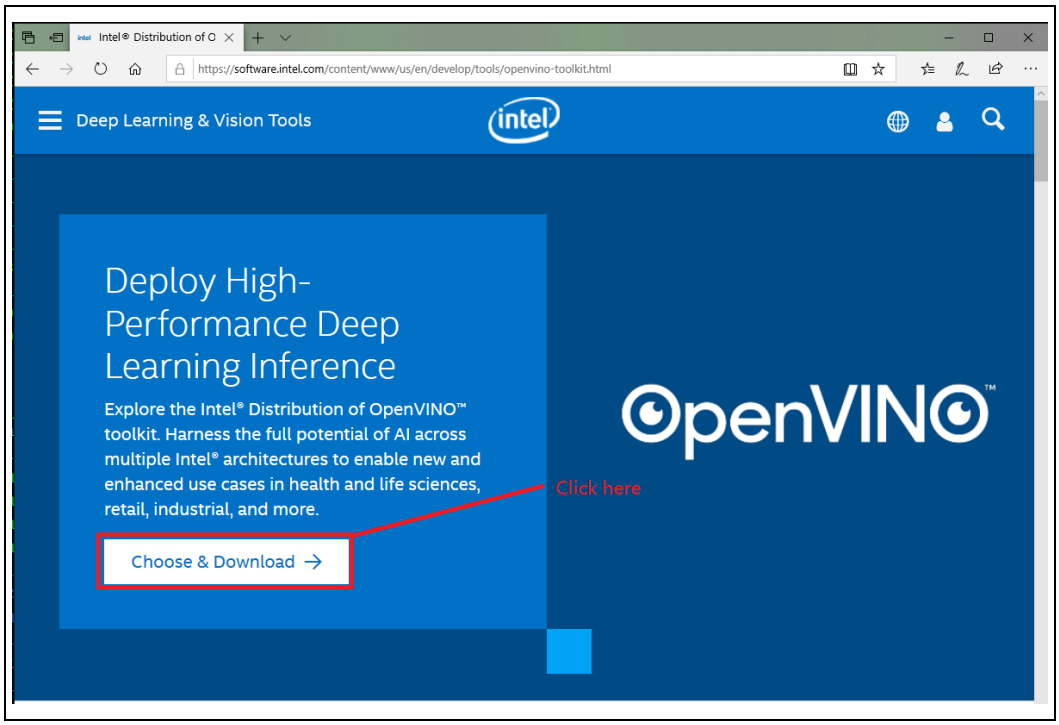
1.2 Install Software Dependencies for CPUs other than Tiger Lake

Skip this step if you are using Tiger Lake and refer to Chapter 1.3.

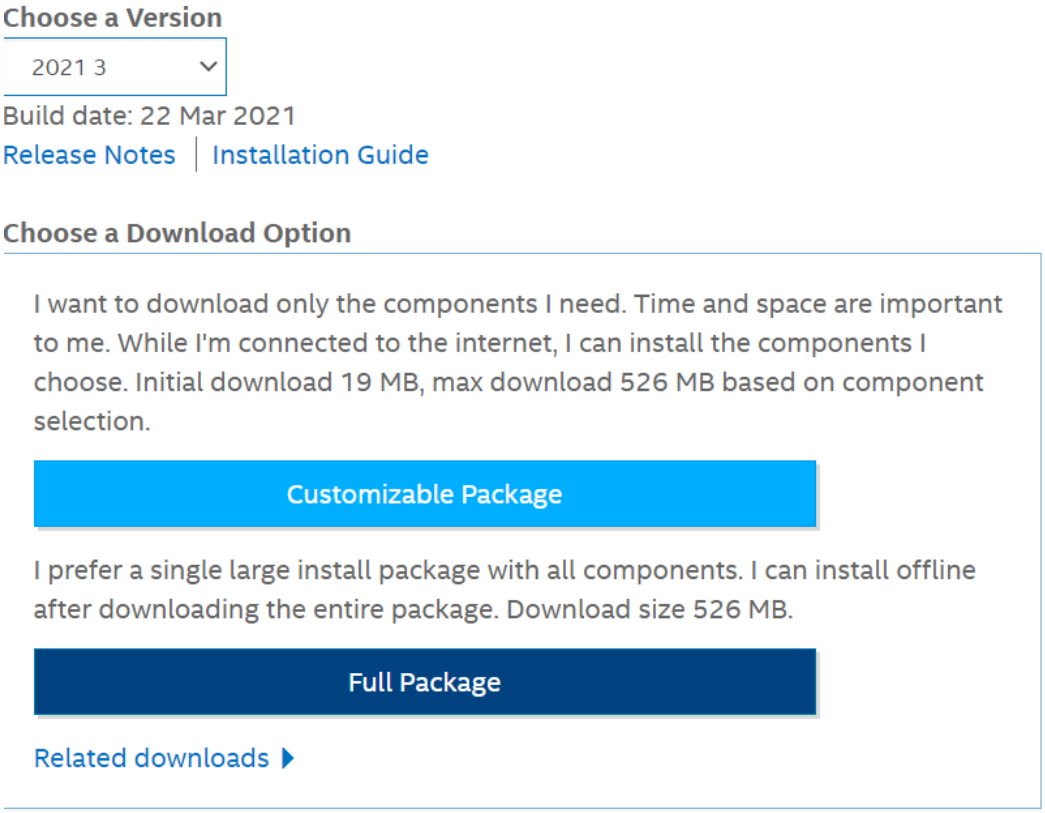
1.2.1 Download OpenVINO™ 2021.3 Linux Release

The sample application **video_e2e_sample** depends on OpenVINO™ libraries. It is suggested that the user installs OpenVINO™ 2021.3 Linux package from https://software.intel.com/en-us/openvino-toolkit.

Use either Edge, Chrome, Safari or Firefox browser to open the above URL. Select the **Choose & Download**.



Select **Linux** → **2021.3** → **Full Package** in options. Then click **Download**.



Next, you may need to provide registry information if you are not signing in with your Intel account.

Note: Make sure that version **2021.3** is selected. Otherwise, downloading other version may cause SVET compiling or runtime error.

1.2.2 Install OpenVINO™

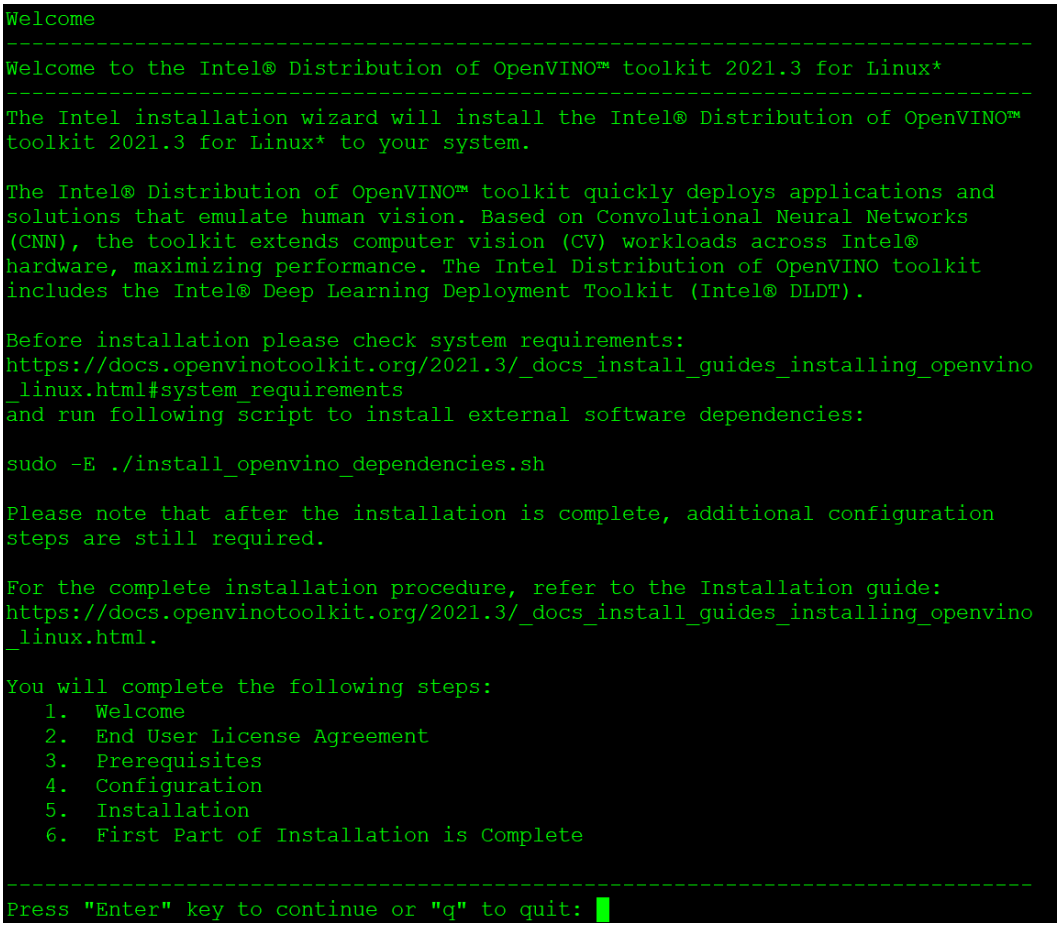
Use below command to uncompress the package:

```
$tar zxf l_openvino_toolkit_p_2021.3.394.tgz
```

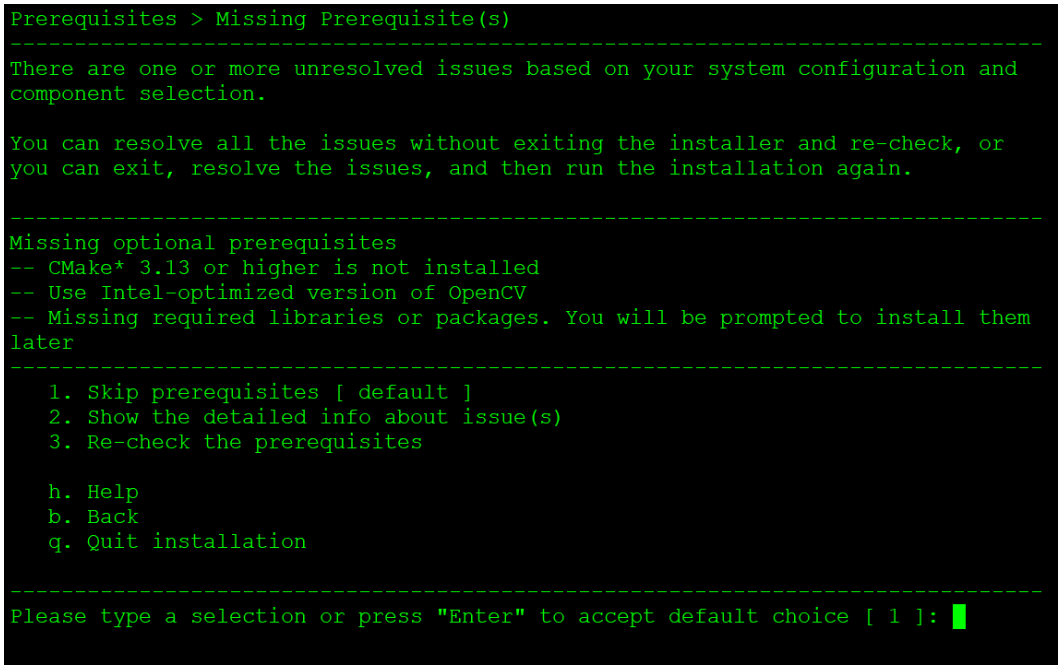
Uninstall the older version of OpenVINO™ toolkit if it has been installed before.

Then run the installation script with sudo:

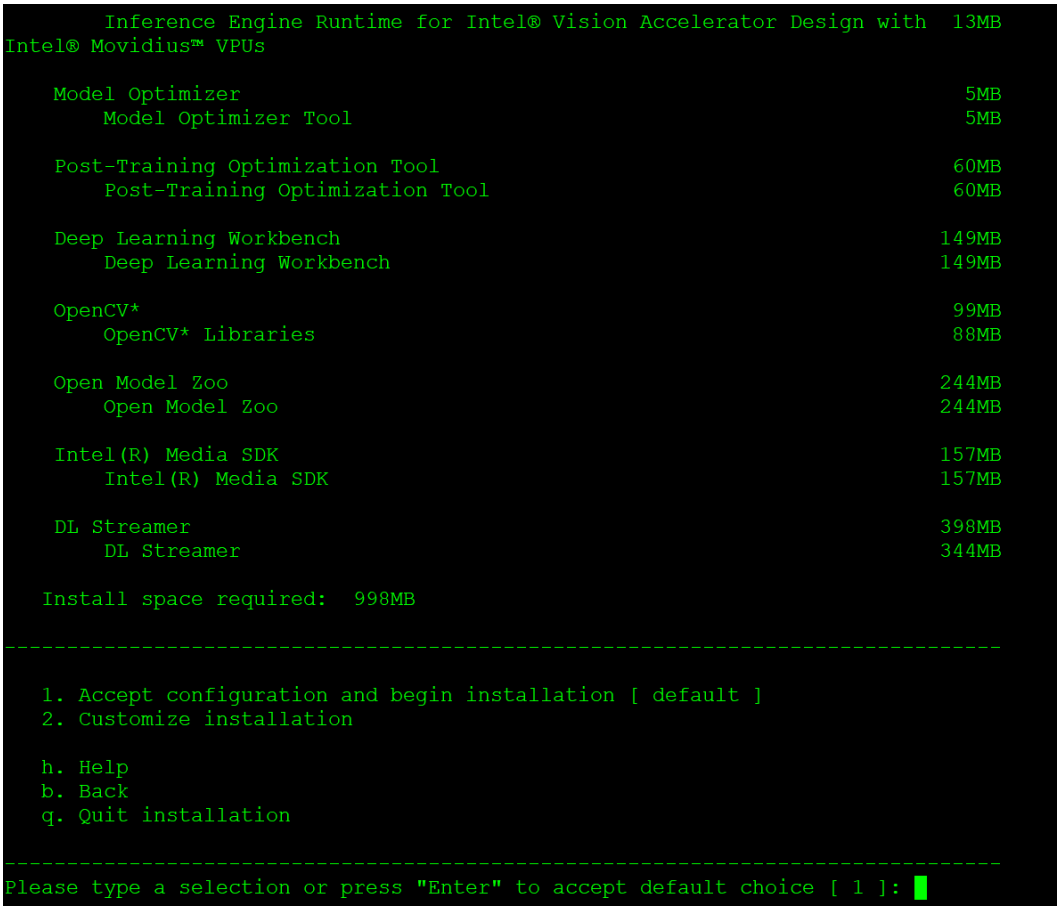
```
$cd l_openvino_toolkit_p_2021.3.394

$sudo ./install.sh
```



Follow the instructions to complete the installation. Type **Enter** to continue. Then type **accept**. Type **1** to continue and you will see commands as shown in the image below.

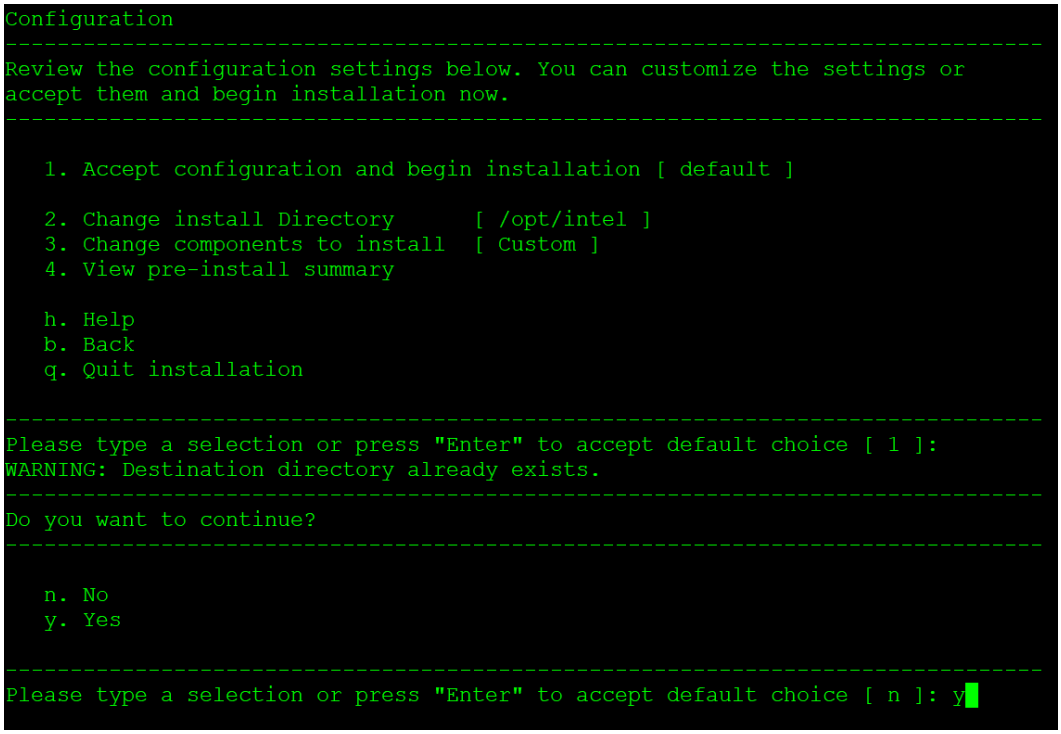


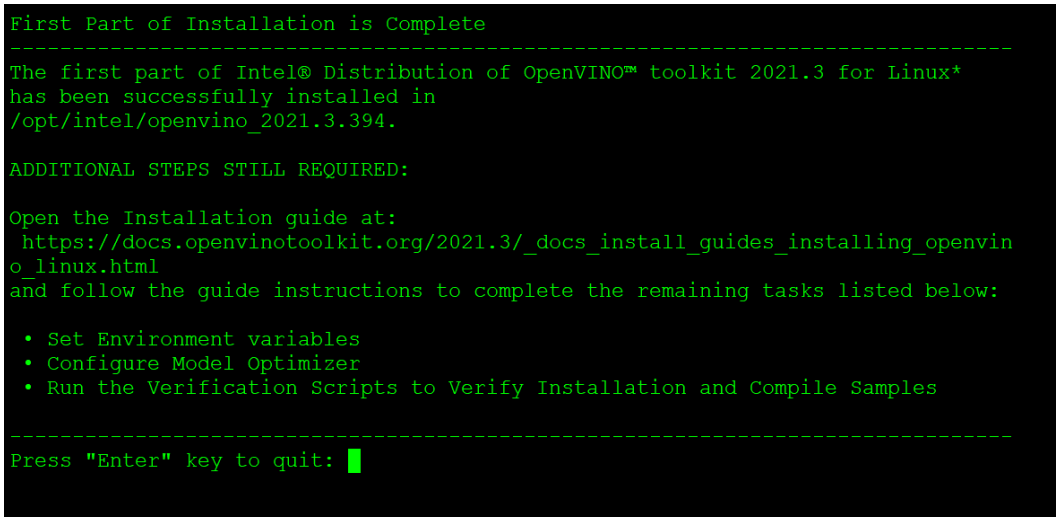
Select **1** to skip prerequisites in this step and you will see the installation configuration page.



Select **1** to start installation.

If you had installed OpenVINO™ to /opt/intel/ before, a confirmation is required to overwrite the folder. Type **y** as shown in the image.





The installation will take few minutes to complete. Run below command and also add to *~/.bashrc*. This command runs the OpenVINO™ environment variables setting up script. SVET build the scripts depending on these environment variables.

```
$ source /opt/intel/openvino_2021/bin/setupvars.sh
```

By default, OpenVINO™ is installed to */opt/intel/openvino*. It also can be installed to *~/intel/openvino*. In this case, replace */opt/intel/openvino* with *~/intel/openvino* in the following instructions in this document.

To avoid running the command *source /opt/intel/openvino_2021/bin/setupvars.sh* multiple times, you can 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* does not run prior in the same bash.

1.2.3 Install NEO driver

Next, run below command to install NEO driver.

```
$ 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
```

Try to uninstall *intel-igc-opencl* and *intel-igc-core* manually with below commands:

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

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

Then add current user to the video group. Replace **USERNAME** with your username using this command and run it:

```
$sudo usermod -a -G video USERNAME
```

Then logout and login again to make the sure the user has been added to group “video”.

Next, you can refer to Chapter 1.4 for building sample application.

1.3 Install Software Dependencies for Tiger Lake CPU

Tiger Lake requires higher Linux kernel version in Ubuntu* 18.04. It also requires higher NEO driver version in OpenVINO™ release.

This chapter provides instructions to upgrade the Linux kernel version and NEO driver version on Tiger Lake.

1.3.1 Upgrade Linux Kernel Manually

Note: Back up your private files before upgrading Linux kernel in case the system is broken after restarting.

Make sure the network connection is stable and there is at least 15G free space in the system before running below commands.

Run below commands to download and install new Linux kernel for Tiger Lake:

```
$sudo apt install coreutils build-essential bc kmod cpio flex
libncurses5-dev libelf-dev libssl-dev bison

$wget https://github.com/intel/linux-intel-
lts/archive/refs/tags/lts-v5.4.102-yocto-210310T010318Z.tar.gz

$tar -xzf linux-intel-lts-lts-v5.4.102-yocto-
210310T010318Z.tar.gz

$cd linux-intel-lts-lts-v5.4.102-yocto-210310T010318Z

$cp /boot/config-5.4.0-77-generic .config

$make oldconfig #Select the default value for unset config
items
$make -j8

$sudo -E make INSTALL_MOD_STRIP=1 modules_install

$sudo -E make install
```

If there is no /boot/config-5.4.0-77-generic no your system, you can check the available 5.4 kernel config by command “ls /boot/config-5.4*”.

Then edit the Linux kernel boot option and add `i915.force_probe=* i915.enable_guc=2` to force GPU module probe

```
$ sudo 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="1> 2"
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=""

$ sudo -E update-grub
```

Then install GPU firmware with the commands:

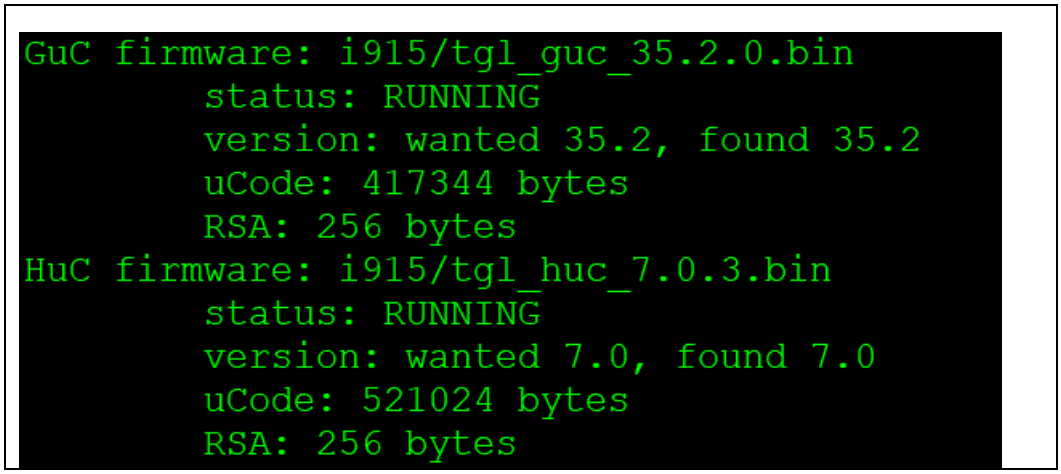
```
$wget
https://git.kernel.org/pub/scm/linux/kernel/git/firmware/linux-
firmware.git/plain/i915/tgl_guc_35.2.0.bin

$wget
https://git.kernel.org/pub/scm/linux/kernel/git/firmware/linux-
firmware.git/plain/i915/tgl_huc_7.0.3.bin

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

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

$sudo update-initramfs -u -k all
```

After restarting, use below commands to confirm the kernel upgrade and GPU firmware.

If there is Linux kernel version newer than v5.4.102 in the system, you need to manually select v5.4.102 kernel in Grub menu during boot.

If v5.4.102 is the latest Linux kernel version in the system, after restarting, it will be booted automatically. You can check the Linux kernel version using command `uname -a`.

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

$sudo cat /sys/kernel/debug/dri/0/i915_gpu_info | grep
firmware: -A 5
```



1.3.2 Install OpenVINO™ Toolkit

Refer to Chapter 1.2.1 and 1.2.2, install OpenVINO™ 21.3 firstly.

1.3.3 Install OpenCL NEO driver 20.52.18783

For Tiger Lake, the OpenCL NEO driver need to be installed manually. Install NEO r20.52.18783 according to instructions on NEO r20.52.18783 release.

Then add current user to the video group. Replace **USERNAME** with your username using this command and run it:

```
$sudo usermod -a -G video USERNAME
```

Then logout and login again to make the sure the user has been added to group “video”.

If the NEO driver is installed correctly, when run *clinfo*, you will see below information

```
$sudo apt install clinfo

$clinfo
```

```
test@test-SYSTEM-PRODUCT-NAME:~$ clinfo
Number of platforms                               1
  Platform Name                                   Intel(R) OpenCL HD Graphics
  Platform Vendor                                 Intel(R) Corporation
  Platform Version                                OpenCL 3.0
  Platform Profile                                FULL_PROFILE
  Platform Extensions                             cl_khr_byte_addressable_store cl_khr_fp16 cl_khr_
bal_int32_extended_atomics cl_khr_icd cl_khr_local_int32_base_atomics cl_khr_local_int32_extended_a
equired_subgroup_size cl_intel_subgroups_short cl_khr_spir cl_intel_accelerator cl_intel_driver_dia
r_throttle_hints cl_khr_create_command_queue cl_intel_subgroups_char cl_intel_subgroups_long cl_khr
memory cl_khr_subgroup_extended_types cl_khr_subgroup_non_uniform_vote cl_khr_subgroup_ballot cl_kh
_khr_subgroup_shuffle cl_khr_subgroup_shuffle_relative cl_khr_subgroup_clustered_reduce cl_intel_sp
bgroups cl_khr_spirv_no_integer_wrap_decoration cl_intel_unified_shared_memory_preview cl_khr_mipma
l_intel_planar_yuv cl_intel_packed_yuv cl_khr_int64_base_atomics cl_khr_int64_extended_atomics cl_k
images cl_khr_3d_image_writes cl_intel_media_block_io cl_intel_va_api_media_sharing cl_intel_subgro
  Platform Host timer resolution                  1ns
  Platform Extensions function suffix             INTEL

  Platform Name                                   Intel(R) OpenCL HD Graphics
Number of devices                                 1
  Device Name                                     Intel(R) Graphics [0x9a49]
  Device Vendor                                   Intel(R) Corporation
  Device Vendor ID                                0x8086
  Device Version                                  OpenCL 3.0 NEO
  Driver Version                                  20.52.18783
  Device OpenCL C Version                         OpenCL C 1.2
  Device Type                                     GPU
  Device Profile                                  FULL_PROFILE
  Device Available                                Yes
  Compiler Available                              Yes
```

Next, you can refer to Chapter 1.4 for building sample application.

1.4 Build concurrent video analytic sample application and dependent libraries

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

```
$git clone https://github.com/intel-iot-devkit/concurrent-
video-analytic-pipeline-optimization-sample-1.git cva_sample
$ cd cva_sample
$./build_and_install.sh

   [ INFO ] Working directory: /home/work/vaas_e2e_sample_1
   ************************************************************
   [ INFO ] Install required tools and create build
   environment.
   ************************************************************
   Enter the sudo password to proceed

[sudo] password for userxxx:
```

This 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. Enter the `sudo` password to continue the installation.

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.

Table 1. Steps in build_and_install.sh

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 2020.3 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 /opt/intel/svet/msdk successfully.
Apply patches under patch/ to Media SDK, then build and install MediaSDK libraries.		A symbol link ./bin/ is created which links to MediaSDK/build/__bin/release/. And Media SDK libraries are installed to /opt/intel/svet/msdk
Install MediaSDK sample_common header files and libraries. Then build video_e2e_sample.		Copy the Media SDK/sample/sample_common header files to /opt/intel/svet/msdk/include/sample_common and copy the libsample_common.a to /opt/intel/svet/msdk/lib/. A directory “build” will be created under video_e2e_sample. Then command “cmake ../; make -j4” will be run under “build” folder. After building is completed, cva_sample /bin/video_e2e_sample is the sample application binary
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

Add libva and Media SDK environment variable setting commands to current bash.	vainfo can run successfully: $ source ./svet_env_setup.sh $ /opt/intel/svet/msdk/bin/vainfo

1.5 Verify sample application's dependency

If *build_and_install.sh* and *source ./svet_env_setup.sh* run successfully, now run */opt/intel/svet/msdk/bin/vainfo* and you will see below output:

```
$ source svet_env_setup.sh

$ /opt/intel/svet/msdk/bin/vainfo

error: can't connect to X server!

libva info: VA-API version 1.9.0

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_9

libva info: va_openDriver() returns 0

vainfo: VA-API version: 1.9 (libva 2.9.0)

vainfo: Driver version: Intel iHD driver for Intel(R) Gen
Graphics - 20.3.0 (dcc5f0e)

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
```

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

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

If there is any missing library, it means the installation was not completed. Contact your account manager from Intel and provide the output from the above command.

1.6 Prepare the video clips for testing

There are two AVC clips for testing under the video folder. If you want to use mp4 video clips, you can use bellow command to extract the element stream from MP4 file:

```
$sudo apt install ffmpeg

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

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

2.0 Run sample application video_e2e_sample

2.1 Check environment variables

Before running sample application, make sure the environment variables are set correctly in the current bash.

Run below command to check whether the OpenVINO™ environment is set:

```
$echo $INTEL_OPENVINO_DIR

/opt/intel/openvino_2021
```

If `$INTEL_OPENVINO_DIR` is empty, run below command to set OpenVINO™ environment.

```
$source /opt/intel/openvino_2021/setupvars.sh
```

Run below command to set the msdk environment variables in current bash:

```
$source ./svet_env_setup.sh
```

2.2 Modify the video path in parameter file

The *build_and_install.sh* downloads two test video clips to the video folder. If you want to use your own test clip, you can modify the video path (following `-i::h264`) of **every line** in example par files under *par_file/inference/n16_1080p_face_detect_30fps.par*. See the text in the red box below.

```
-i::h264 ./video/1080p.h264 -join -hw -async 4 -dec_postproc  -
o::sink -vpp_comp_dst_x 480 -vpp_comp_dst_y 540 -vpp_comp_dst_w
480 -vpp_comp_dst_h 270 -ext_allocator -infer::fd ./model -fps
30
```

Otherwise you will see below error message when running the sample application:

```
[ERROR], sts=MFX_ERR_NULL_PTR(-2), Init, m_fSource pointer is
NULL at
/home/work/video_e2e_sample_1/MediaSDK/samples/video_e2e_sample
/src/file_and_rtsp_bitstream_reader.cpp:165
```

2.3 Enable cl_cache

The loading of inference models can take a long time. It is 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, refer to https://github.com/intel/compute-runtime/blob/master/opencl/doc/FAQ.md.

2.4 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 are active VNC sessions, close them first. The **-p** option is to keep the current user environment variables settings.

IMPORTANT NOTICE: Run *source ./svet_env_setup.sh* first when you start a new bash (or change user in bash such as run **su -p**) to run *./bin/video_e2e_sample*

If you want to run *video_e2_sample* with normal user or with X11 display, you can replace *-rdrm-DisplayPort* with *-rx11*. See *par_file/inference/n16_face_detection_1080p_x11.par* for inference.

Note: X11 rendering is not 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 of par files for several typical use cases. Refer to Chapter 2.4 for the detailed information of parameters in par files.

2.4.1 16-channel video decoding, face detection, composition, encode and display

If you have not run the following command to set the MediaSDK environment for your current bash, run it before running **video_e2e_sample** application.

Command line to set the MediaSDK environment:

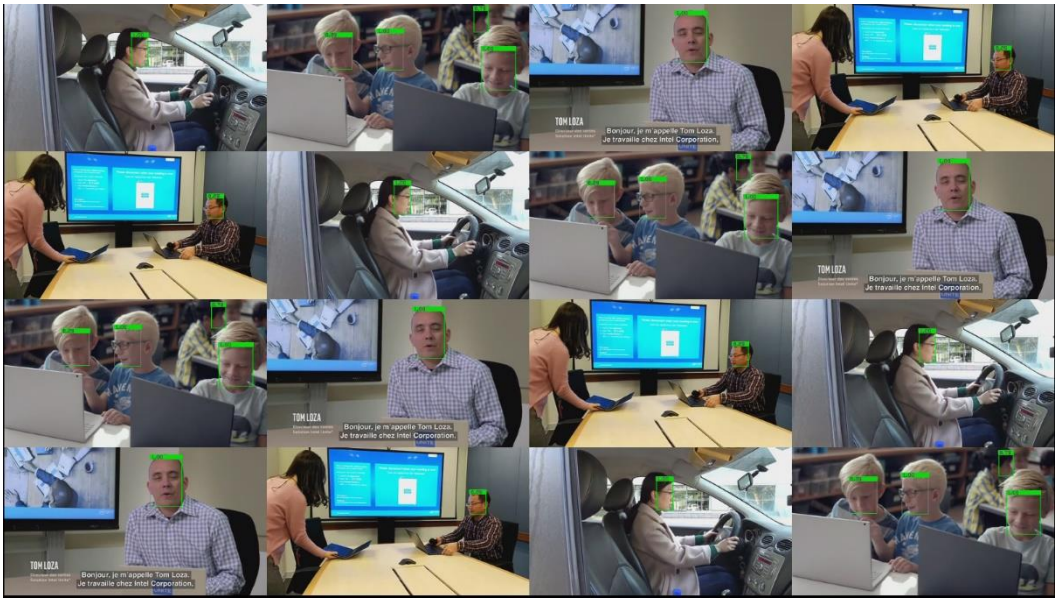
```
#source ./svet_env_setup.sh
```

Command line to run the **video_e2e_sample** application:

```
#./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 are required to wait for a minute until the video showing on the display as depicted in the following image. With **cl_cache** enabled, the next running of face detection models will be much faster, which is about 10 seconds on CFL.



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 the par files.

By default, the pipeline is running as fast as it can. If you want to limit the FPS to a certain number, add `-fps FPS_number` to every decoding sessions, which start with `-i` in the par files. Refer to *par_file/inference/ n16_1080p_face_detect_30fps.par*.

2.4.2 4-channel video decoding, human pose estimation, composition, and display

If you have not run the following command to set the MediaSDK environment for your current bash, run it before running the **video_e2e_sample** application.

Command line to set the MediaSDK environment:

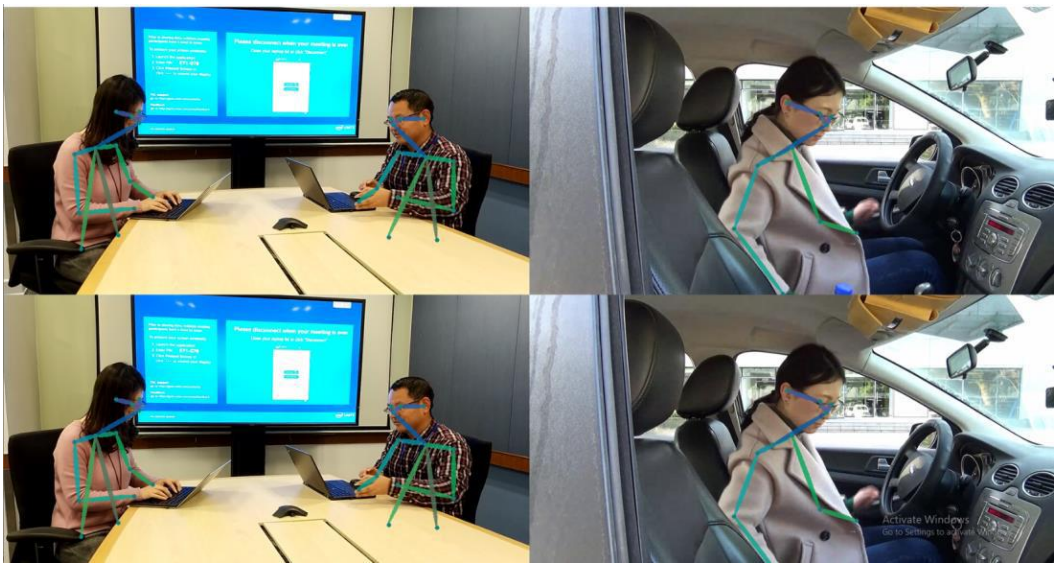
```
./source svet_env_setup.sh
```

Command line to run the **video_e2e_sample** application:

```
./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.

Below is the image of this demo.



2.4.3 4-channel video decoding, vehicle and vehicle attributes detection, composition, encode and display

If you have not run the following command to set the MediaSDK environment for your current bash, run it before running the **video_e2e_sample** application.

Command line to set the MediaSDK environment:

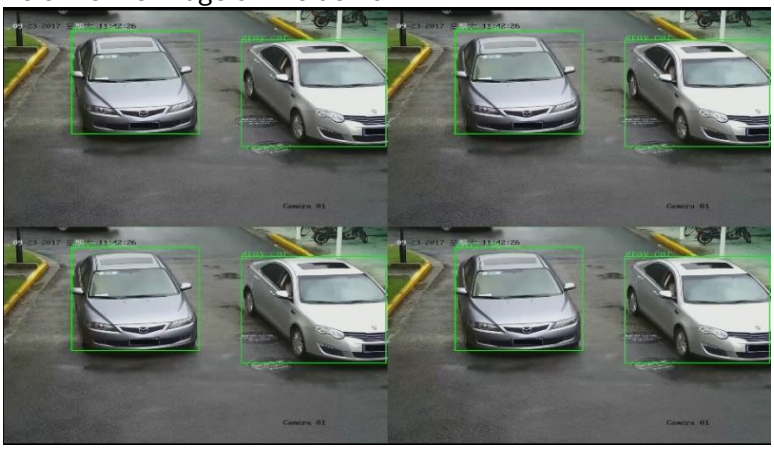
```
./source svet_env_setup.sh
```

Command line to run the **video_e2e_sample** application:

```
./bin/video_e2e_sample -par
par_file/inference/n4_vehicel_detect_1080p.par
```

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

Below is the image of this demo.



2.4.4 16-channel RTSP video decoding, face detection, composition, encode and display

If you have not run the following command to set the MediaSDK environment for your current bash, run it before running the **video_e2e_sample** application.

Command line to set the MediaSDK environment:

```
./source svet_env_setup.sh
```

Command line to run the **video_e2e_sample** application:

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

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

2.4.5 4-channel video decoding, multi objects detection/tracking, composition, and display

If you have not run the following command to set the MediaSDK environment for your current bash, run it before running the **video_e2e_sample** application.

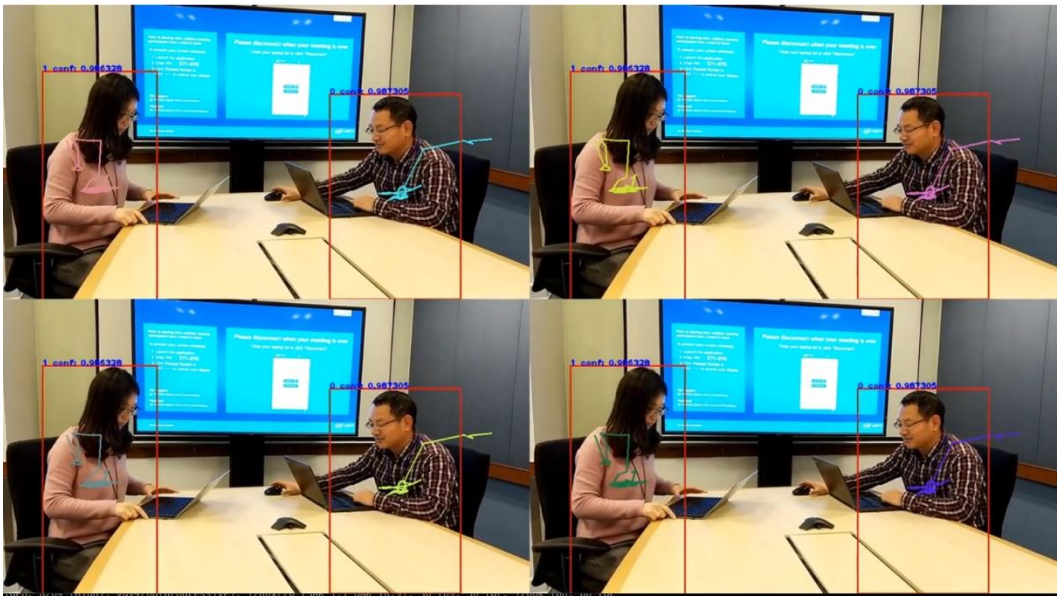
Command line to set the MediaSDK environment:

```
./source svet_env_setup.sh
```

Command line to run the **video_e2e_sample** application:

```
./bin/video_e2e_sample -par
par_file/inference/n4_multi_object_tracker.par
```

The object detection and motion tracking inference are specified by `-infer::mot ./model` in the par file. `./model` is the directory that stores objects detection and motion tracking models IR files.



2.4.6 2-Channel Video Decoding, Yolov3 Detection, Composition and Display

To convert Yolov3 model, you can refer to https://docs.openvinotoolkit.org/latest/openvino_docs_MO_DG_prepare_model_convert_model_tf_specific_Convert_YOLO_From_Tensorflow.html.

If you have not run the following command to set the MediaSDK environment for your current bash, run it before running the **video_e2e_sample** application.

Command line to set the MediaSDK environment:

```
#./source svet_env_setup.sh
```

You can edit *par_file/inference/n2_yolo_h264.par* and replace *yolofp16/yolo_v3.xml* with the yolov3 IR file path in your system.

Command line to run the **video_e2e_sample** application:

```
#./bin/video_e2e_sample -par par_file/inference/n2_yolo_h264.par
```

2.4.7 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.

2.4.8 Shared inference network instance

Starting from R3, the sessions that use same network IR files and same inference device shared one inference network instance. The benefit is that when GPU plugin is used, the network loading time decreases by 93% for 16-channel inferences.

2.4.9 16-channel RTSP video decoding, RTSP stream storing, face detection, composition, encode, and display

If you have not run the following command to set the MediaSDK environment for your current bash, run it before running the **video_e2e_sample** application.

Command line to set the MediaSDK environment:

```
#./source svet_env_setup.sh
```

Command line to run the **video_e2e_sample** application:

```
#./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.

2.4.10 2-channel RTSP stream storing

If you have not run the following command to set the MediaSDK environment for your current bash, run it before running the **video_e2e_sample** application.

Command line to set the MediaSDK environment:

```
$./source svet_env_setup.sh
```

Command line to run the **video_e2e_sample** application:

```
$./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.

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
```

2.4.11 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.

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.

If you have not run the following command to set the MediaSDK environment for your current bash, run it before running the **video_e2e_sample** application.

Command line to set the MediaSDK environment:

```
#./source svet_env_setup.sh
```

Command line to run the **video_e2e_sample** application:

```
#./bin/video_e2e_sample -par par_file/basic/n16_1080p_30fps_videowall.par par_file/basic/
n16_1080p_30fps_videowall.par
```

2.4.12 Use fake sink

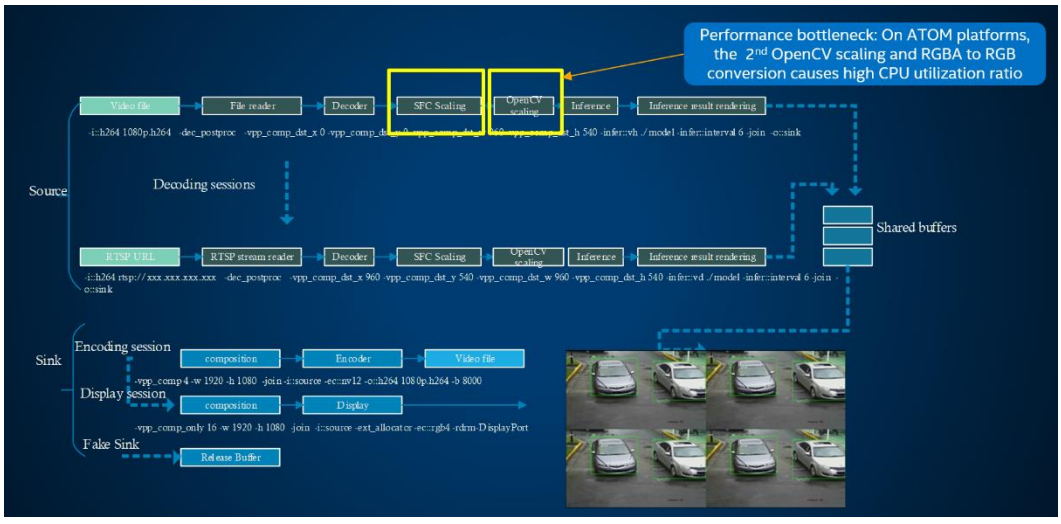
By using option `-fake_sink`, user can run the concurrent video decoding with fake sink instead of display or encoder. In this mode, the composition of decoding or inference result is disabled. Refer to example par files *n16_1080p_decode_fakesink.par* under folder **par_file/misc** and *n16_1080p_face_detection_fakesink.par*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

2.4.13 Use VPP instead of SFC in decoding session

H265 decoder doesn't support SFC, so VPP (Accelerated by Execution Unit in Intel Graphics) is used for scaling and color format convert in video decoding sessions. 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**.

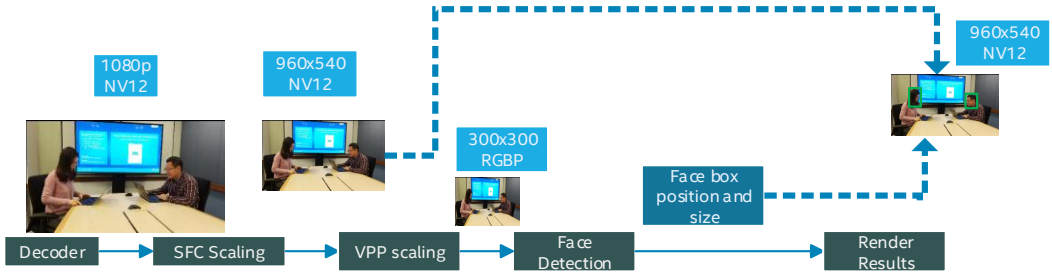
2.4.14 Enable two outputs from video decoder.

As you can see in below diagram of SVET pipeline, there are 2 scaling stages. The first one is done by GPU. The output size of first scaling is specified by vpp_comp_width and vpp_comp_height parameters in par file. The second one is done with OpenCV by CPU. And its input is the output of first scaling and its output size is set according to the input size of inference network. On ATOM platforms, we notice that the second scaling cost too much CPU computing resource and it impacts the whole pipeline performance.



Starting from R3, SVET supports enabling two outputs from video decoder by adding `-dc::rgbp` to each decoding session. As you can see in below picture, one output is from SFC with size equal to the composition input size in NV12 format. And the other is from VPP with size equal to inference input size and in RGBP format. This option only can be used together with `-infer::fd`.

With this optimization, for 4-channel face detection on APL platform, the CPU utilization ratio is reduced by half.



2.4.15 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. User can also use option `-infer::device CPU` to specify CPU as target device.

In one par file, user can use different devices for each session.

If HDDL is used as inference engine, 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.
Refer to example par file *n1_infer_options.par*.

2.4.16 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*.

2.4.17 MCU mode

MCU stands for Multiple Controller Unit. In MCU mode, SVET sample application can be used to test multiple channel video decoding, video composition and video encoding at the same time.

For example, below command can be used to test 8 1080p AVC decode, 8 1080p composition and 8 1080p AVC encoding workload:

```
$./source svet_env_setup.sh

$./bin/video_e2e_sample -par mcu1_1080p_4to4.par
mcu2_1080p_4to4.par -stat 100
```

2.4.18 Run concurrent video analytic sample application without OpenVINO™

Some of our users only care about media performance and don't need inference features. In such case, user can build SVET sample application without OpenVINO™ installation.

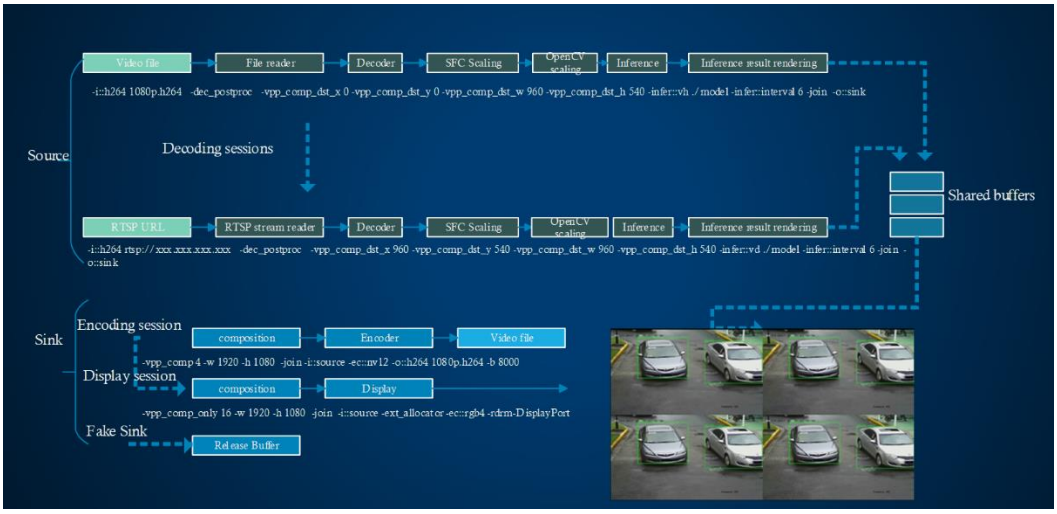
The build command is shown as below:

```
$./build_and_install.sh -b no_ocv
```

With option "-b no_ocv", the build script won't check the environment variable INTEL_OPENVINO_DIR and use a special cmake configuration file which excluded the inference related source code.

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

As you can see in below picture, 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.



2.5.1 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. Examples: -infer::fd ./model →face detection -infer::hp ./model →human pose estimation -infer:vd ./model →vehicle and vehicle attributes detection -infer:mot ./model →multi objects tracking -infer::fd ./model -infer::offline →face detection but not render the results to display -infer::fd ./model/person-detection-retail-0013.xml -> Person detection by specify the XML file directly
-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. Refer to n16_1080p_h265_fd.par and n16_h265_1080p_rtsp_simu.par.
-dc::rgbp	Enable two outputs from AVC video decoder. One is from SFC with size equal to the composition input size in NV12 format. And the other is from VPP with size equal to inference input size and in RGBP format. This option only can be used together with "-infer::fd".
-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. **Note:** "-o" option must be used together with this option, but it won't generate any output file.
-infer::device <GPU, CPU, HDDL>	Indicate the inference target device. Refer to example par file n1_infer_options.par. If this option isn't set, GPU will be used as inference engine.
-infer::interval <number>	Indicate the distance between two inference frames. 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. Refer to example par file n1_infer_options.par.
-infer::remote_blob	Enable remote_blob feature of OpenVINO™ GPU plugin. Note, if this option is set, the decoder output will be in NV12 format with size equal to inference input size. There will be no display. So this option currently only support offline inference.
-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
-vpp_comp_dump null_render	Disabling rendering after VPP Composition. This is for performance measurements. See par_file/misc/n16_1080p_decode_vpp_comp_no_display.par

-o::raw /dev/null	when use “-o::raw” with output file name “/dev/null”, application will drop the decode output frame instead of encoding or saving to local file. It’s for pure video decoding testing.

2.5.2 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

Table 2. Parameters Used in Example Par Files

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. Switch Ubuntu* to text mode(Ctrl + Alt + F3) and root user by command "su -p" before using this parameter.
-rx11	Using X11 as display. Make sure environment variable DISPLAY set correctly if run the sample application remotely in a console terminal.

2.6 Frequently Asked Questions

Q: Where can I find the description of options used in par file?

A: See chapter 2.4 in doc/svet_sample_application_user_guide_2021.1.0.pdf

Running the SVET sample applicton with option "-?" can show the usage of options.

Q: Why does the system need to be switched to text console mode before running the sample application?

A: The sample application uses libDRM to render the video directly to display, so it needs to act as master of the DRM display, which isn't allowed when X client is running. If there is any VNC session, close it. Because VNC session also starts X client.

If the par file doesn't include display session, there is no need to switch to text mode.

Q: Why does "su -p" is required to switch to root user before running the sample application?

A: To become DRM master, it needs root privileges. With option "-p", it will preserve environment variables, like LIBVA_DRIVERS_PATH, LIBVA_DRIVER_NAME and LD_LIBRARY_PATH. If without "-p", these environment variables will be reset and the sample application will run into problems.

Q: Is it possible to use X11 instead of DRM display?

A: If user doesn't want to switch to text console mode or switch to root for using DRM display, user can replace "-rdrm-DisplayPort" with "-rx11" in the par file. However, the X11 rendering isn't as effcient as DRM rendering. According to our 16-channel face detection 1080p test on CFL , the time cost of each frame increased by around 6ms. Example [par file](./par_file/inference/n16_face_detection_1080p_x11.par) using X11 as rendering method.

Q: Is there any limitation of the order of decoding, encoding and dislay sessions in the par file?

A: Yes. The decoding dessions must be descripted firstly. If there is display dession, it must be the last line in par file.

Q: The loading time of 16-channel face detection demo is too long.

A: Make sure **cl_cache** is enabled by command `echo $cl_cache_dir`. If this environment is not set, enable **cl_cache** by running command `export cl_cache_dir=/tmp/cl_cache` and `mkdir -p /tmp/cl_cache`. Then after the first running of 16-channel face detection demo, the compiled OpenCL kernles are cached and the model loading time of next running of 16-channel face detection demo will only take about 10 seconds.

More details about **cl_cache** can be found at https://github.com/intel/compute-runtime/blob/master/opencl/doc/FAQ.md

Q: Can source numbers for `-vpp_comp_only` or `-vpp_comp` be different from number of decoding sessions?

A: No. The source numbers for `-vpp_comp_only` or `-vpp_comp` must be equal to the numer of decoding sessions. Otherwise, the sample application will fail during pipeline initialization or running.

Q: How to limit the fps of whole pipeline to 30?

A: Add `-fps 30` to every decoding session.

Q: Why does `-fps 30` not working with `-fake_sink`?

A: Fake sink session does not support `-fps 30`. Add `-fps 30` to every decoding session instead.

Q: How to limit the frame number of inputs to 1000?

A: Add `-n 1000` to every decoding dessions. But do not add `-n` to encode, display and fake sink session. These sink sessions will automatically stop when the source session stops. Note, this option is not working if both `-vpp_comp_only` and `-vpp_comp` are set.

Q: Where can I find tutorials for inference engine?

A: Refer to https://docs.openvinotoolkit.org/latest/_docs_IE_DG_Deep_Learning_Inference_Engine_DevGuide.html.

Q: Why is the HDDL card usage ratio low for face detection inference?

A: It can be caused by the decoded frames that are not fed to inference engine efficiently. The default inference interval of face detection is 6. Try to set the inference interval to a lower value when using HDDL as inference target device. For example, with 3 HDDL L2 card, adding `-infer::inverval 1` to 16-channel face detection par file can increase the HDDL usage ratio to 100%.

Q: Where can I find information for the models?

A: Refer to https://github.com/opencv/open_model_zoo/tree/master/models/intel. The names of models used in sample application are

- face-detection-retail-0004
- human-pose-estimation-0001
- vehicle-attributes-recognition-barrier-0039
- vehicle-license-plate-detection-barrier-0106

Q: Can I use other OpenVINO™ version other than 2021.3?

A: Yes, but you must modify some code due to changing interfaces. And also you need to download the IR files and copy them **to ./model manually**. Refer to *script/download_and_copy_models.sh* for how to download the IR files.

Q: When run 4-channel decode plus inference and display on APL, the CPU occupy ratio is very high and fps is low?

A: You can refer to par file *par_file/inference/n4_face_detection_rgbp.par*. It uses option `-dc::rgbp` that makes the SFC outputs RGB4 for display and VPP outputs RGBP for inference input. It is not required to use OpenCV for resizing and color conversion which consume more CPU time on APL.

Note: `-dc::rgbp` only works with `-infer::fd`. More inference types will be supported in the future.

3.0 Pack video_e2e_sample Binaries and Install on Another Device

After *install_and_build.sh* script running successfully on one device, users can use scripts (*pack_binary.sh*, *install_binary.sh*) to pack and deploy the **video_e2e_sample** to other devices with binaries only.

3.1 Pack video_e2e_sample Binaries

pack_binary.sh can be used to copy the **video_e2e_sample** and other dependent binaries into a folder.

Run below commands under the source code directory and all the **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

 svet_env_setup.sh
```

3.2 Install video_e2e_sample Binaries

Users can also deploy the packed binaries with *install_binary.sh* script. Before that, make sure Ubuntu* 18.04 and OpenVINO™ have been installed on the new devices. Meanwhile on new device, the OpenVINO™ must be installed to the same path as the OpenVINO™ installation path on original device which **video_e2e_sample** is 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 the **video_e2e_sample** application with par files.

4.0 Monitor overall GPU resource usage statistics

There are some tools can be used to view GPU resource usage statistics. Refer to chapter 3.1.4 white paper CDI#621636 for additional info.

4.1 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`. *render busy* stands for the utilization of the programmable execution unit in Intel Graphics.