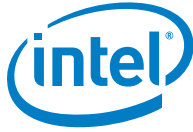


Concurrent Video Analytic Sample Application User Guide

Version 2020.3.0

Oct 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>

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 MediaSDK, OpenVINO 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.

Copyright © 2020, Intel Corporation. All rights reserved.

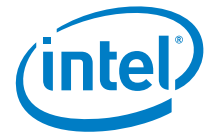


Contents

1.0	Installation Guide.....	6
1.1	System installation.....	6
1.2	TGL-U Upgrade Linux kernel and iGPU firmware	6
1.2.1	Tiger Lake U: Upgrade Linux Kernel	6
1.3	Install OpenVINO 2021.1.....	8
1.3.1	Download OpenVINO package.....	8
1.3.2	Install OpenVINO.....	10
1.3.3	Tiger Lake U: Install OpenCL NEO driver 20.25.17111	17
1.4	Build concurrent video analytic sample application and dependent libraries.....	18
1.5	Verify sample application's dependency	20
1.6	Prepare the video clips for testing.....	21
2.0	Run sample application video_e2e_sample.....	22
2.1	Check environment variables.....	22
2.2	Modify the video path in parameter file	22
2.3	Enable cl_cache.....	22
2.4	Run video_e2e_sample application	23
2.4.1	16-channel video decoding, face detection, composition, encode and display	23
2.4.2	4-channel video decoding, human pose estimation, composition, and display	24
2.4.3	4-channel video decoding, vehicle and vehicle attributes detection, composition, encode and display	25
2.4.4	16-channel RTSP video decoding, face detection, composition, encode and display	25
2.4.5	Offline inference mode.....	26
2.4.6	Shared inference network instance	26
2.4.7	16-channel RTSP video decoding, RTSP stream storing, face detection, composition, encode, and display	26
2.4.8	2-channel RTSP stream storing	26
2.4.9	Multiple displays.....	27
2.4.10	Use fake sink	27
2.4.11	Use VPP instead of SFC in decoding session	27
2.4.12	Enable two outputs from video decoder.	27
2.4.13	Configure the inference target device, inference interval and maximum object number.....	28
2.4.14	Configure the interval of JPEG encoding	29
2.5	Usage of media codec, inference and display parameters in par file	29
2.5.1	New parameters in Par file.....	30
2.5.2	Decode, encode and display parameters.....	32
3.0	Pack video_e2e_sample Binaries and Install on Another Device.....	34



3.1	Pack video_e2e_sample Binaries	34
3.2	Install video_e2e_sample Binaries	34
4.0	Monitor overall GPU resource usage statistics	35
4.1	Intel_gpu_top	35
4.2	Intel-telemetry-tool	35



Revision History

Date	Revision	Description
2020/09/23	3.0	1. Update the OpenVINO and Media SDK version 2. Add descriptions for R3 new features
2020/05/19	2.0	1. Update the OpenVINO and Media SDK version 2. Add descriptions for R2 new features
2020/03/03	1.0	1. Add new example par files 2. Add tables to explain parameters usage in par file
2019/12/26	0.5	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 18.04.02 to a Coffee Lake device (e.g. NUC8i7BEH)

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

1.2 **TGL-U Upgrade Linux kernel and iGPU firmware**

You can skip this chapter if you're using Sky Lake, Coffee Lake or Whiskey Lake U with GPU GEN 9.

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

```
$cat /sys/kernel/debug/dri/0/i915_capabilities | grep gen  
gen: 9
```

On Tiger Lake U, the GPU generation is 12.

1.2.1 **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 5.4 or above:

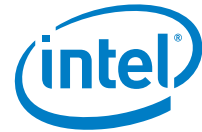
```
$uname -a
```

If not, please use below command to upgrade to kernel 5.4 and reboot the device.

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

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

```
$ wget https://github.com/intel/linux-intel-lts/archive/lts-v5.4.61-yocto-200907T010133Z.tar.gz  
$ tar -xzf lts-v5.4.61-yocto-200907T010133Z.tar.gz  
$ cd lts-v5.4.61-yocto-200907T010133Z  
$ cp /boot/config-5.4.0-51-generic .config //Copy Ubuntu default kernel config file  
$ make oldconfig //Select the default value for unset config items
```



```
$ make -j8

$ sudo make INSTALL_MOD_STRIP=1 modules_install

$ sudo 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 check /boot/grub/grub.cfg to make sure above modification effective.
```

Please install GPU firmware by below command:

```
$ 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

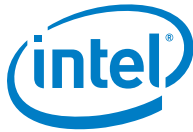
$ sudo update-initramfs

$ sync
```

After rebooting, please use below command to confirm the kernel upgrade and GPU firmware. The Firmware version requirement may be different due to the different kernel version. This snapshot comes from an older kernel version:

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

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



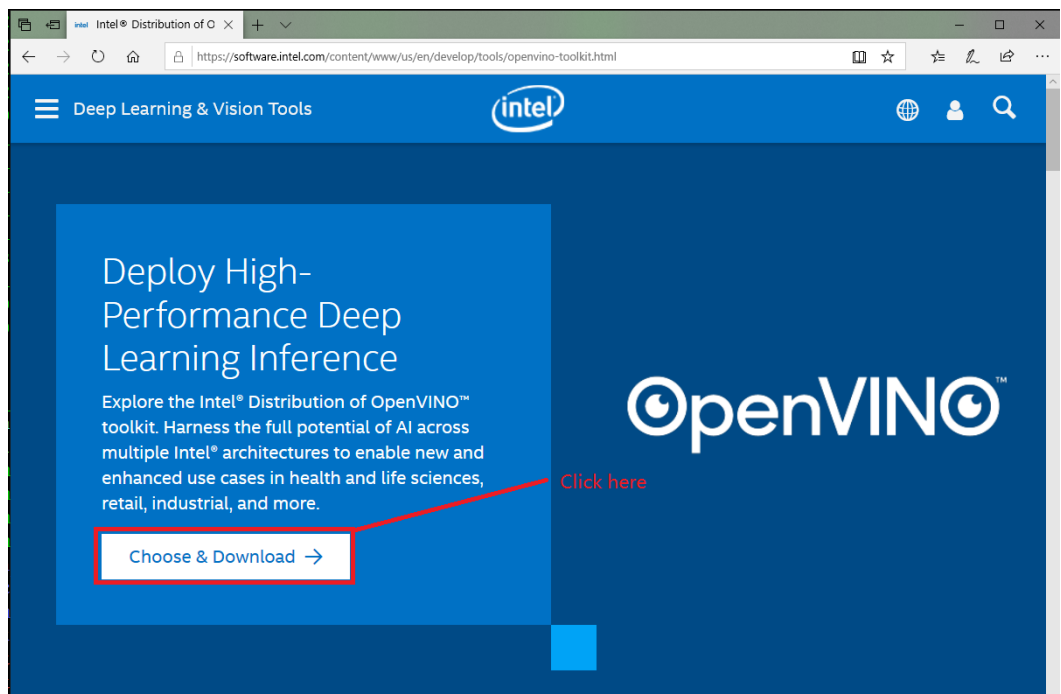
```
GuC firmware: i915/tgl_guc_43.0.1.bin
status: RUNNING
version: wanted 43.0, found 43.0
uCode: 320064 bytes
RSA: 256 bytes
HuC firmware: i915/tgl_huc_7.0.12.bin
status: RUNNING
version: wanted 7.0, found 7.0
uCode: 529984 bytes
RSA: 256 bytes
```

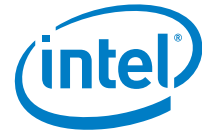
1.3 Install OpenVINO 2021.1

1.3.1 Download OpenVINO package

The sample application `video_e2e_sample` depends on OpenVINO libraries. We suggest users to install OpenVINO 2021.1 Linux package from <https://software.intel.com/en-us/openvino-toolkit>

Please use browser Edge, Chrome, Safari or Firefox to open the above URL. Click the “Choose & Download” button.





Please select “Linux”, “Web Download”, “Offline” in options. Then click “Register & Download”. See below picture.

In next page, you may need to fill out registry information if you didn't login with your Intel account.

Make Your Vision a Reality

Intel® Distribution of OpenVINO™ toolkit is built to fast-track development and deployment of high-performance computer vision and deep learning inference applications on Intel® platforms—from security surveillance to robotics, retail, AI, healthcare, transportation, and more.

- **Accelerate Performance** – Speed computer vision workloads, and enable easy execution across multiple types of Intel® processors and accelerators: CPU, GPU/Intel® Processor Graphics, VPU, and FPGA.
- **Streamline Deep Learning Deployment** – Unleash CNN-based deep learning inference using a common API, 30+ pre-trained models, and code samples. The toolkit supports more than 100 public and custom models.
- **Extend and Customize** – Use OpenCL® kernels and tools to add your own code into the workload pipeline; customize layers without the overhead of frameworks.
- **Save Time, Increase Productivity** – Develop faster with optimized OpenCV®, OpenVX®, and media encode/decode functions; 15+ samples; and more.
- **Innovate Artificial Intelligence** – Extend AI within your applications with the included Intel® Deep Learning Deployment Toolkit – optimize AI at the edge all the way to cloud.

In the OpenVINO downloading page, please make sure version “2021.1” is select. Otherwise, it might download other version and cause SVET compiling or runtime error.



Then click "Full Package" and the downloading will begin. See below picture.

Serial number:

- Save this serial number. You may need it to activate your product in the installer.
- For your reference, you will receive an email that includes your serial number and download instructions.

Select 2021.1

Choose a Version

- 2021.1
- 2021.1
- 2020.4
- 2020.3 LTS
- 2020.2
- 2020.1
- 2019 R3.1
- 2019 R3
- 2019 R2.0.1
- 2019 R2
- 2019 R1.1
- 2019 R1.0.1
- 2019 R1
- 2018 R5.0.1

Download Option

Customizable Package

Full Package

Click "Full Package"

Access your support resources. [Click here](#) for technical support.

Intel takes your privacy seriously. Refer to Intel's [Privacy Notice](#) and [Serial Number Validation Notice](#) regarding the collection and handling of your personal information, the Intel product's serial number and other information.

1.3.2 Install OpenVINO

Firstly, please use below command to uncompress the package:

```
$tar xzf l_openvino_toolkit_p_2021.1.110.tgz
```

Please uninstall the old version of OpenVINO if it has been installed before.

Then please run the installation script with sudo:

```
$cd l_openvino_toolkit_p_2021.1.110  
$sudo ./install.sh
```



```
Welcome
-----
Welcome to the Intel® Distribution of OpenVINO™ toolkit 2021.1 for Linux*
-----
The Intel installation wizard will install the Intel® Distribution of OpenVINO™
toolkit 2021.1 for Linux* to your system.

The Intel® Distribution of OpenVINO™ toolkit quickly deploys applications and
solutions that emulate human vision. Based on Convolutional Neural Networks
(CNN), the toolkit extends computer vision (CV) workloads across Intel®
hardware, maximizing performance. The Intel Distribution of OpenVINO toolkit
includes the Intel® Deep Learning Deployment Toolkit (Intel® DLDT).

Before installation please check system requirements:
https://docs.openvino toolkit.org/2021.1/\_docs\_install\_guides\_installing\_openvino
\_linux.html#system\_requirements
and run following script to install external software dependencies:

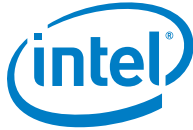
sudo -E ./install_openvino_dependencies.sh

Please note that after the installation is complete, additional configuration
steps are still required.

For the complete installation procedure, refer to the Installation guide:
https://docs.openvino toolkit.org/2021.1/\_docs\_install\_guides\_installing\_openvino
\_linux.html.

You will complete the following steps:
1. Welcome
2. End User License Agreement
3. Prerequisites
4. Configuration
5. Installation
6. First Part of Installation is Complete
-----
Press "Enter" key to continue or "q" to quit: █
```

Please following the instructions to complete the installation. Type "Enter" to continue. Then type "accept" to continue. Type "1" to continue and you'll see below picture:



```
Prerequisites > Missing Prerequisite(s)
-----
There are one or more unresolved issues based on your system configuration and
component selection.

You can resolve all the issues without exiting the installer and re-check, or
you can exit, resolve the issues, and then run the installation again.

-----
Missing optional prerequisites
-- CMake* 3.13 or higher is not installed
-- Use Intel-optimized version of OpenCV
-----

1. Skip prerequisites [ default ]
2. Show the detailed info about issue(s)
3. Re-check the prerequisites

h. Help
b. Back
q. Quit installation

-----
Please type a selection or press "Enter" to accept default choice [ 1 ]: █
```

Select "1" to skip prerequisites in this step and you'll see below installation configuration page. Since MediaSDK will be downloaded and installed by SVET script, there is no need to install MediaSDK during OpenVINO installation.



```
Inference Engine Runtime for Intel® Processor Graphics      18MB
Inference Engine Runtime for Intel® Movidius™ VPU          84MB
Inference Engine Runtime for Intel® Gaussian Neural Accelerator 13MB
Inference Engine Runtime for Intel® Vision Accelerator Design with 13MB
Intel® Movidius™ VPUs

  Model Optimizer                                           4MB
  Model Optimizer Tool                                     4MB

  Deep Learning Workbench                                  134MB
  Deep Learning Workbench                                  134MB

  OpenCV*                                                  99MB
  OpenCV* Libraries                                        88MB

  Open Model Zoo                                           111MB
  Open Model Zoo                                           111MB

  Intel(R) Media SDK                                       128MB
  Intel(R) Media SDK                                       128MB

  DL Streamer                                              389MB
  DL Streamer                                              342MB

Install space required:  868MB

-----

1. Accept configuration and begin installation [ default ]
2. Customize installation

h. Help
b. Back
q. Quit installation

-----

Please type a selection or press "Enter" to accept default choice [ 1 ]: █
```

Please select “2” to go to customized installation page. Then select “3” to change components to install.

```
Configuration
-----
Review the configuration settings below. You can customize the settings or
accept them and begin installation now.
-----

1. Accept configuration and begin installation [ default ]

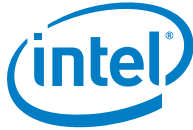
2. Change install Directory      [ /opt/intel ]
3. Change components to install  [ All ]
4. View pre-install summary

h. Help
b. Back
q. Quit installation

-----

Please type a selection or press "Enter" to accept default choice [ 1 ]: █
```

Type “7” to select MediaSDK in the component list.



```
Configuration > Component selection
-----
Select the components you want to customize.
-----

1. Accept and continue [ default ]
2. [All] Inference Engine
3. [All] Model Optimizer
4. [All] Deep Learning Workbench
5. [All] OpenCV*
6. [All] Open Model Zoo
7. [All] Intel(R) Media SDK
8. [All] DL Streamer

Install space required: 868MB
Space available: 9.4GB

a. Select/unselect all
h. Help
b. Back
q. Quit installation
-----
Please type a selection or press "Enter" to accept default choice [ 1 ]: 7
```

Then, type “2” to unselect MediaSDK from components list.

```
Configuration > Component selection
-----
You may select/unselect components that are not marked required.
-----

1. Accept and continue [ default ]
2. [x] Intel(R) Media SDK
3. [x] Intel(R) Media SDK

Install space required: 128MB
Space available: 9.4GB

h. Help
b. Back
q. Quit installation
-----
Please type a selection or press "Enter" to accept default choice [ 1 ]: 2
```

Finally, type “b” to go back and type “Enter” twice to begin installation.



```
Configuration > Component selection
-----
You may select/unselect components that are not marked required.
-----

1. Accept and continue [ default ]
2. [ ] Intel(R) Media SDK
3.   [ ] Intel(R) Media SDK

Install space required:    0MB
Space available:   9.4GB

h. Help
b. Back
q. Quit installation
-----
Please type a selection or press "Enter" to accept default choice [ 1 ]: b
```

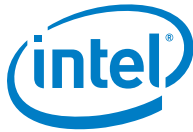
```
Configuration > Component selection
-----
Select the components you want to customize.
-----

1. Accept and continue [ default ]
2. [All] Inference Engine
3. [All] Model Optimizer
4. [All] Deep Learning Workbench
5. [All] OpenCV*
6. [All] Open Model Zoo
7. [None] Intel(R) Media SDK
8. [All] DL Streamer

Install space required:  740MB
Space available:   9.4GB

a. Select/unselect all
h. Help
b. Back
q. Quit installation
-----
Please type a selection or press "Enter" to accept default choice [ 1 ]:
```

If you had installed OpenVINO to /opt/intel/ before, it will need confirmation. Please type "y" as shown in below picture.



```
Configuration
-----
Review the configuration settings below. You can customize the settings or
accept them and begin installation now.
-----

  1. Accept configuration and begin installation [ default ]
  2. Change install Directory      [ /opt/intel ]
  3. Change components to install [ Custom ]
  4. View pre-install summary

  h. Help
  b. Back
  q. Quit installation
-----
Please type a selection or press "Enter" to accept default choice [ 1 ]:
WARNING: Destination directory already exists.
-----
Do you want to continue?
-----
  n. No
  y. Yes
-----
Please type a selection or press "Enter" to accept default choice [ n ]: y
```

```
First Part of Installation is Complete
-----
The first part of Intel® Distribution of OpenVINO™ toolkit 2021.1 for Linux*
has been successfully installed in
/opt/intel/openvino_2021.1.110.

ADDITIONAL STEPS STILL REQUIRED:

Open the Installation guide at:
https://docs.openvino toolkit.org/2021.1/\_docs\_install\_guides\_installing\_openvino\_linux.html
and follow the guide instructions to complete the remaining tasks listed below:

• Set Environment variables
• Configure Model Optimizer
• Run the Verification Scripts to Verify Installation and Compile Samples
-----
Press "Enter" key to quit: 
```

The installation will complete in a few minutes. Please run below command and also add it to ~/.bashrc. This command runs the OpenVINO environment variables setting up script. SVET build scripts depends on these environment variables.

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




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

If you're not using Tiger Lake U, make sure the OpenCL driver is installed correctly by running “sudo /opt/intel/opencvino/install_dependencies/install_NEO_OCL_driver.sh”.
If you are using Tiger Lake U, please following instructions in Chapter 1.3.3 to install OpenCL NEO driver manually.

If you see below error message during the installation of NEO OCL driver:

```
dpkg: dependency problems prevent removal of intel-igc-core:
intel-igc-openccl 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-openccl and intel-igc-core manually by bellow commands:

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

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

Run “source /opt/intel/opencvino/bin/setupvars.sh” and add “source /opt/intel/opencvino/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.

1.3.3 Tiger Lake U: Install OpenCL NEO driver 20.25.17111

If you're using Tiger Lake U, The OpenCL NEO driver need to be installed manually. Please install NEO r20.25.17111 according to [NEO r20.25.17111 release](#)



```
$ apt install clinfo
```

You will see below information if the NEO driver installed correctly.

```
root@tgl-Tiger-Lake-Client-Platform:/home/tgl/james/neo# clinfo
Number of platforms                                1
Platform Name                                     Intel(R) OpenCL HD Graphics
Platform Vendor                                   Intel(R) Corporation
Platform Version                                  OpenCL 2.1
Platform Profile                                   FULL_PROFILE
Platform Extensions                               cl_khr_byte_addressable_store cl_khr_fp16 cl_khr_glob
cl_khr_local_int32_extended_atomics cl_intel_subgroups cl_intel_required_subgroup_size cl_intel_subgroups
e_hints cl_khr_create_command_queue cl_intel_subgroups_char cl_intel_subgroups_long cl_khr_il_program c
nified_shared_memory_preview cl_khr_mipmap_image cl_khr_mipmap_image_writes cl_intel_planar_yuv cl_inte
eprh_images cl_intel_media_block_io cl_khr_3d_image_writes cl_intel_va_api_media_sharing cl_intel_subgr
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) Gen12LP HD Graphics NEO
Device Vendor                                   Intel(R) Corporation
Device Vendor ID                                0x8086
Device Version                                  OpenCL 2.1 NEO
Driver Version                                   20.18.16699
Device OpenCL C Version                          OpenCL C 2.0
```

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-
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.



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 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 /usr/lib/x86_64-linux-gnu/ 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/mediasdk/
Add libva and Media SDK environment variable setting commands to .bashrc and also run these commands in current bash.		<p>Add bellow commands to ~/.bashrc if they are not added before.</p> <p>vainfo can run successfully</p> <pre>export LD_LIBRARY_PATH=\$LD_LIBRARY_PATH:/usr/lib /x86_64-linux-gnu:/usr/lib</pre> <pre>export LIBVA_DRIVERS_PATH=/usr/lib/x86_64-linux-gnu/dri</pre> <pre>export LIBVA_DRIVER_NAME=iHD</pre> <pre>export MFX_HOME=/opt/intel/mediasdk</pre> <pre>LD_LIBRARY_PATH="\$LD_LIBRARY_PATH:/opt/intel/mediasdk/lib</pre>
Install MediaSDK sample_common header files and libraries. Then build video_e2e_sample.		<p>Copy the Media SDK/sample/sample_common header files to /opt/intel/mediasdk/include/sample_common and copy the libsample_common.a to /opt/intel/mediasdk/lib/.</p> <p>A directory "build" will be created under video_e2e_sample. Then command "cmake ../; make -j4" will be run under "build" folder.</p> <p>After building is completed, cva_sample /bin/video_e2e_sample is the sample application binary</p>



Run script/download_and_copy_models.sh to download OpenVINO face detection, human pose estimation and vehicle detection models IR files to directory model/	<pre>\$ 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</pre>
---	---

1.5 Verify sample application's dependency

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

```
l$ 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

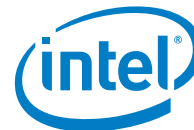
libva info: va_openDriver() returns 0

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 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

1.6 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 this sample application only supports element stream, you can use below 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 sample application video_e2e_sample*

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_2021
```

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=MXF_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_rea  
der.cpp:165
```

2.3 Enable cl_cache

The 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>



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 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.

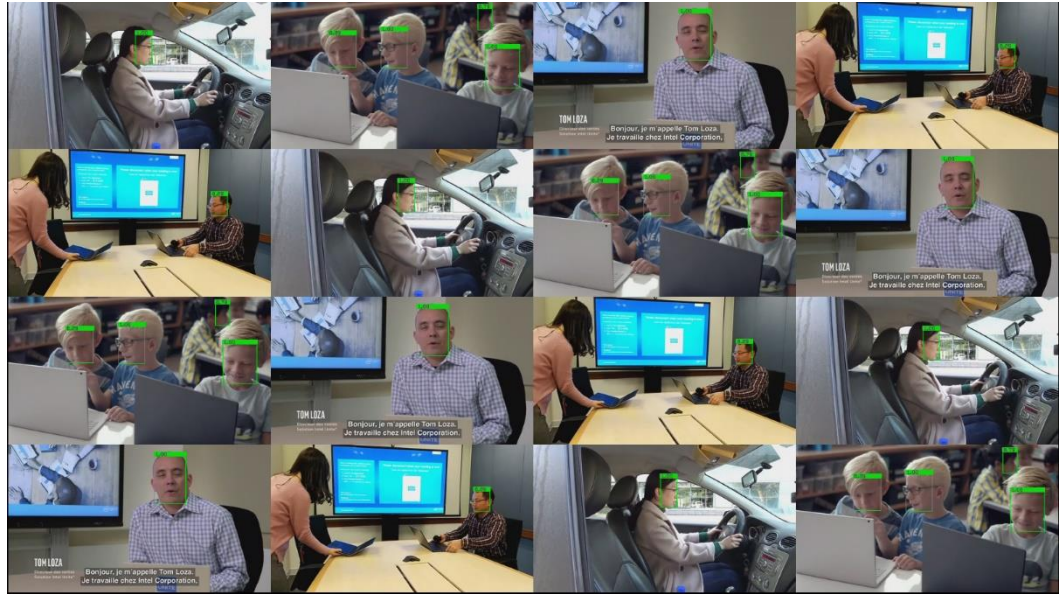
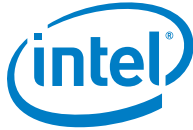
2.4.1 16-channel video decoding, face detection, composition, encode and display

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.



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 sessions, which start with “-i” in par files. Please refer to par_file/inference/n16_1080p_face_detect_30fps.par.

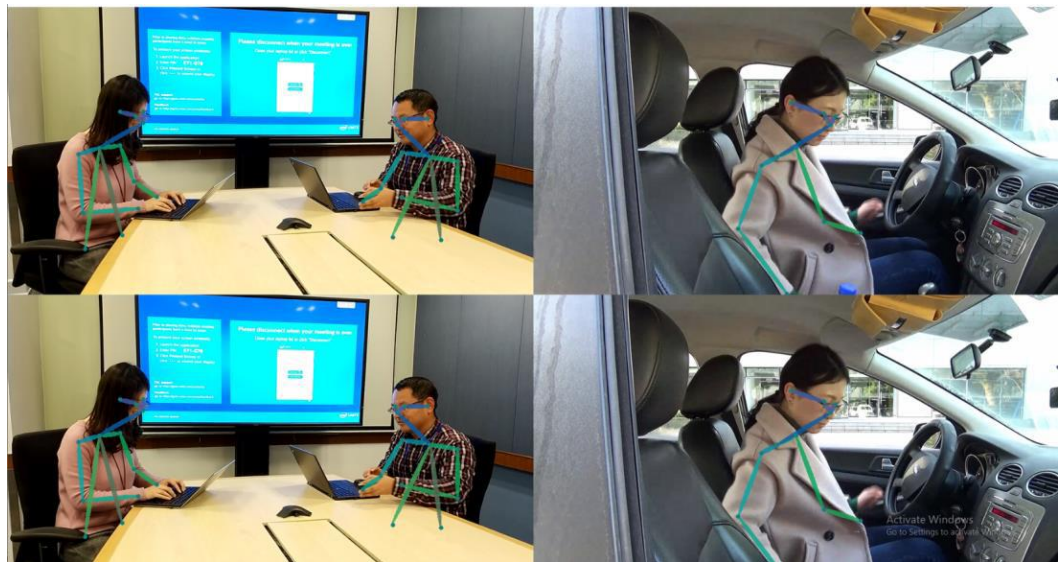
2.4.2 4-channel video decoding, human pose estimation, composition, and display

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.

Below picture is the screenshot of this demo.



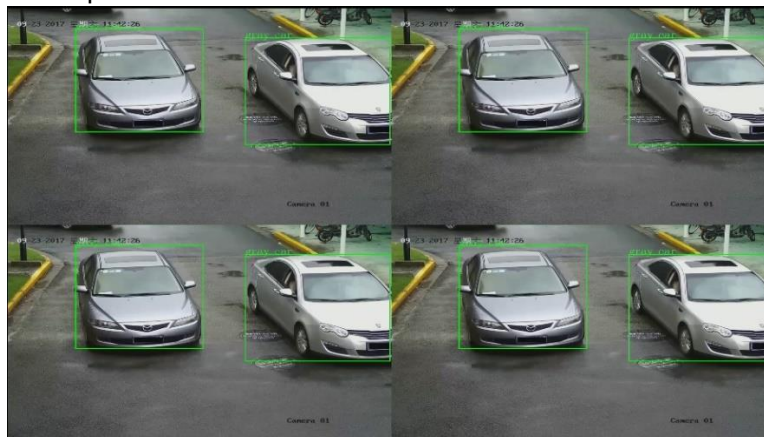
2.4.3 4-channel video decoding, vehicle and vehicle attributes detection, composition, encode and display

Command line:

```
./bin/video_e2e_sample -par par_file/inference/n4_vehicle_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.



2.4.4 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
```

2.4.5 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.

2.4.6 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.

2.4.7 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.

2.4.8 2-channel RTSP stream storing

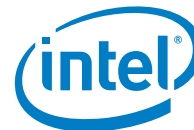
Command line:

```
./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
```

2.4.9 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, “-rdm-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
```

2.4.10 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.

2.4.11 Use VPP instead of SFC in decoding session

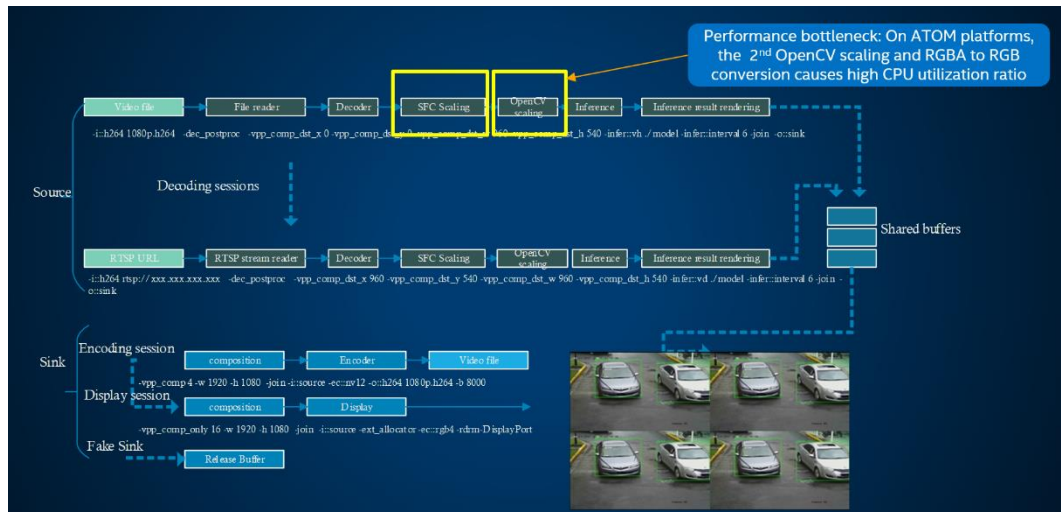
H265 decoder doesn't support SFC, so VPP(Accelerated by Execution Unit in Intel Graphics) is used for scaling and color format convert in video decoding sessions. Please refer to example par file n16_1080p_h265_fd.par under folder par_file/inference and n16_h265_1080p_rtsp_simu.par under folder par_file/rtsp.

2.4.12 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

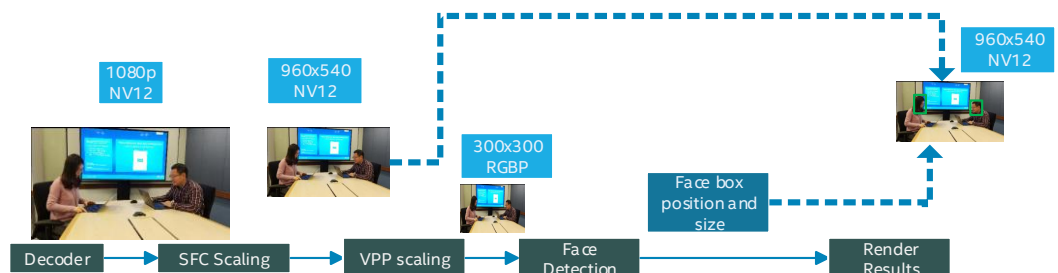


input size of inference network. On ATOM platforms, we notice that the second scaling cost too much CPU computing resource and it impacts the whole pipeline performance.



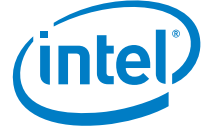
Starting from R3, SVET supports enabling two outputs from video decoder by adding “-dc::rgbp” to each decoding session. As you can see in below picture, one output is from SFC with size equal to the composition input size in NV12 format. And the other is from VPP with size equal to inference input size 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.



2.4.13 Configure 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, please firstly make sure the HDDL device has been set up successfully. See `n4_vehicle1_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.

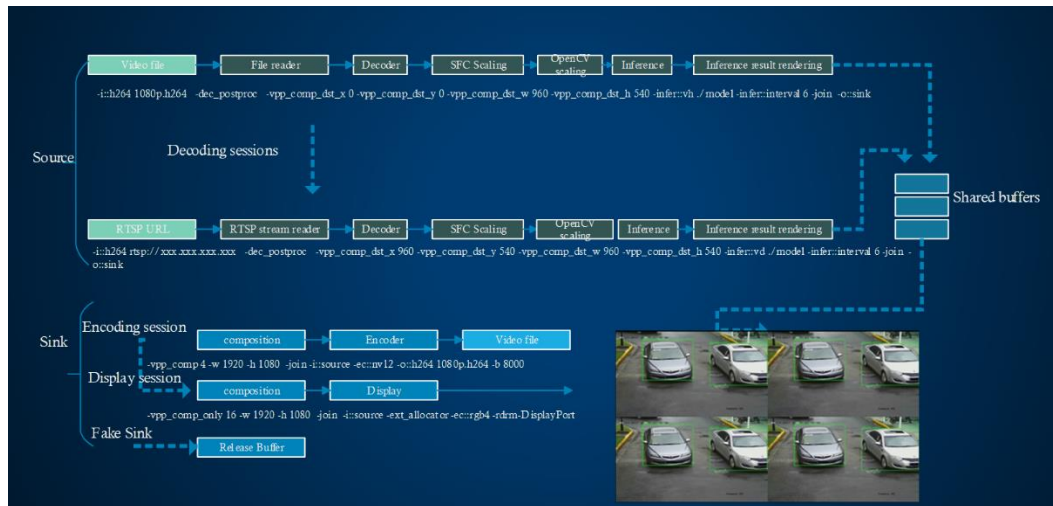
Please refer to example par file `n1_infer_options.par`.

2.4.14 Configure 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`.

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

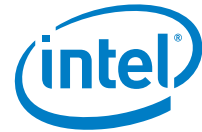
As you can see in below picture, the pipeline contains multiple sessions. Each session is defined by one line in par file. The session can be source or sink. The source session is decoding session and defined by lines starting with `"-i"`. The sink session can be encoding session that is defined `"-vpp_com"`, 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	<p>Specify the inference type and directory that stores the IR files. Can be used together with -infer::offline.</p> <p>Examples:</p> <ul style="list-style-type: none"> -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 -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	<p>Save RTSP stream to local file. This parameter must be used together with "-i::h264 rtsp://url".</p> <p>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.</p>



-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.
-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. Please 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. Please 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 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. Please 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

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. 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.



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 video_e2e_sample to other devices with binaries only

3.1 Pack video_e2e_sample Binaries

pack_binary.sh can be used 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".

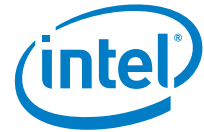
```
./script/pack_binary.sh  
  
$ls cva_e2e_sample_l/  
  
download_models.sh libva      media-driver par_file          video_e2e_sample  
install_binary.sh libva-utils MediaSDK    run_face_detection_test.sh
```

3.2 Install video_e2e_sample Binaries

Users can also deploy the packed binaries with install_binary.sh script. Before that, please 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 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. Please also refer to chapter 3.1.4 white paper [CDI#621636](#)

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.



4.2 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 VBox and VEDBox. 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 below commands:

```
$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:

