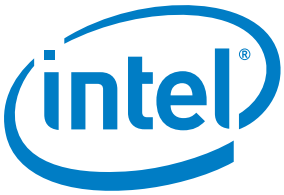


Smart Video Evaluation Toolkit – Sample Application User Guide for Windows

May 2020

Contents

Revision History

Date	Revision	Description
2020/05/26	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

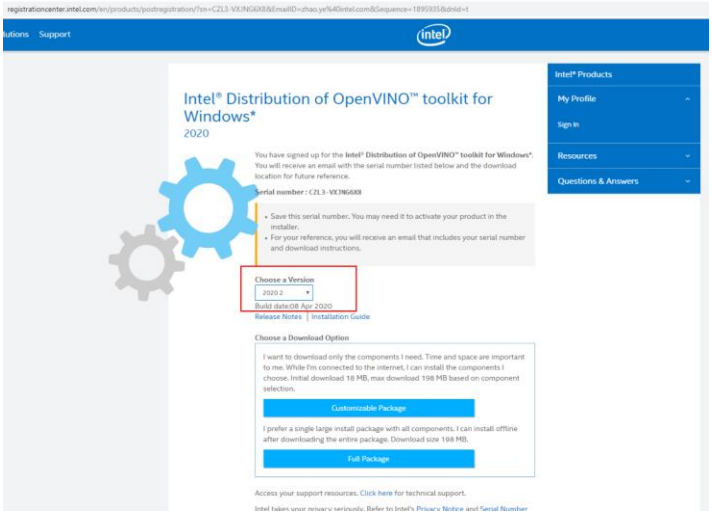
1.1 System installation

Install Windows 10 enterprise on any Intel® Core™ or Atom™ processors. (Better to be after the 5th generation Core™ and Baytrail for Atom™ processors)

Set up the network correctly

1.2 Install OpenVINO 2020.2 and Media SDK

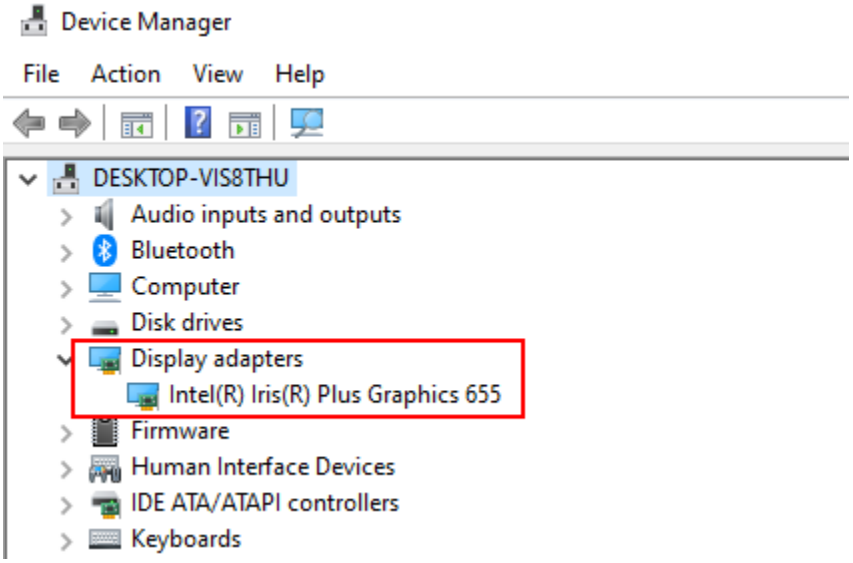
The sample application video_e2e_sample depends on OpenVINO libraries, Media SDK library and FFMPEG library. We suggest users to install OpenVINO 2020.2 package from https://software.intel.com/en-us/openvino-toolkit.



Please Install OpenVINO 2020.2 according to https://docs.openvinotoolkit.org/latest/_docs_install_guides_installing_openvino_windows.html

By default, OpenVINO 2020.2 is installed to “C:\Program Files (x86)\IntelSWTools\openvino_2020.2.117”.

Make sure the Graphic driver is installed correctly in the Windows device manager, it should look like below status.



If the driver is not installed properly, you can download the Graphic driver by below link:

https://downloadcenter.intel.com/product/80939/Graphics-Drivers?elq_cid=3226637&erpm_id=5670311

Please Install Media SDK 2019 R1 according to:

https://software.intel.com/en-us/media-sdk/choose-download/client

Media SDK installation guide as below:

1, Install the SDK and library files. (Note Installation requires full administrative rights.)

2, Set up the Microsoft Visual Studio* environment with the SDK and library directories.

3, Run the Intel_Media_SDK_20xx_xx.msi installer. The default installation directory is c:\Program Files (x86)\IntelSWTools\Intel(r)_Media_SDK_XXXX where XXXX is the year and version number.

4, After installation, restart your system to initialize INTELMEDIASDK_WINSDK_PATH and install the remaining files.

Other Media SDK Installation document, you can reference to:

https://software.intel.com/en-us/media-sdk/documentation/featured-documentation

when you build video_e2e_sample, it will depend on Media SDK sample code (sample_common and sample_plugins), which you can find in below folder, after Media SDK installed:

C:\Users\zhaoye\Documents\Intel® Media SDK 2019 R1 - Media Samples 8.4.27.25\

And for the RTSP function, this video_e2e_sample building also depends on FFMPEG library, it can be downloaded from:

https://ffmpeg.zeranoe.com/builds/win64/dev/

ffmpeg-20191224-287620f-win64-dev.zip is validated, and extracted to C:\Users\zhaoye\Desktop\ffmpeg\ffmpeg-20191224-287620f-win64-dev\ffmpeg-20191224-287620f-win64-dev\ , which will be used to compile the video_e2e_sample in visual studio.

1.3 Build SVET sample application and dependent libraries

Download svet_e2e_sample_w.zip sample package, or you can git clone the svet_e2e_sample_w code from github https://github.com/intel-iot-devkit/concurrent-video-analytic-pipeline-optimization-sample-w

and extract it to a folder, such as

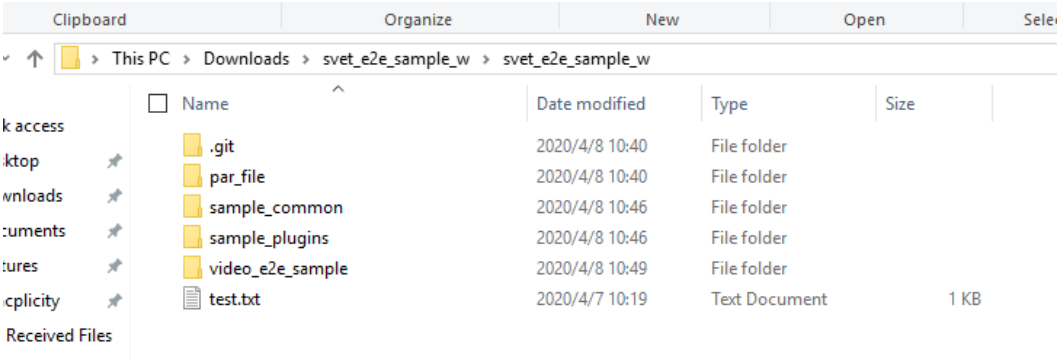
C:\Users\zhaoye\Downloads\svet_e2e_sample_w\svet_e2e_sample_w

Copy the Media SDK dependent sample code to svet_e2e_sample_w,

```
cp C:\Users\zhaoye\Documents\Intel® Media SDK 2019 R1 - Media Samples 8.4.27.25\sample_common C:\Users\zhaoye\Downloads\svet_e2e_sample_w\svet_e2e_sample_w

cp  C:\Users\zhaoye\Documents\Intel® Media SDK 2019 R1 - Media Samples 8.4.27.25\sample_plugins C:\Users\zhaoye\Downloads\svet_e2e_sample_w\svet_e2e_sample_w
```

Then the folder will look like below.



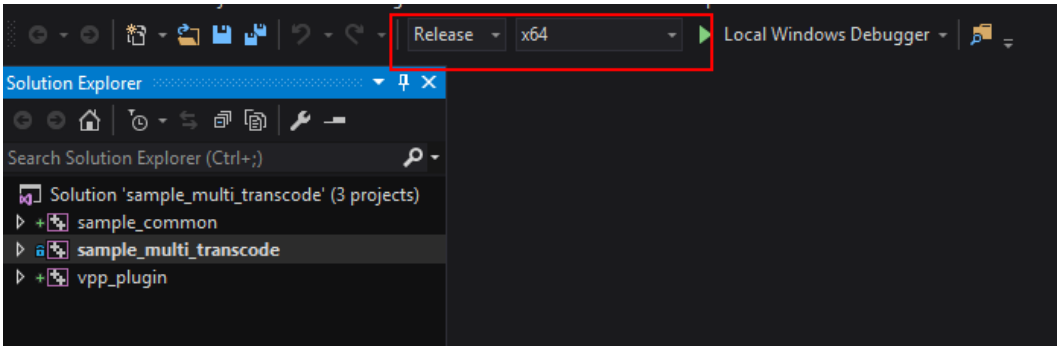
Open Visual studio (suggest to use Visual studio 2017 and Windows SDK 10), Windows SDK download link：

https://go.microsoft.com/fwlink/p/?LinkId=838916

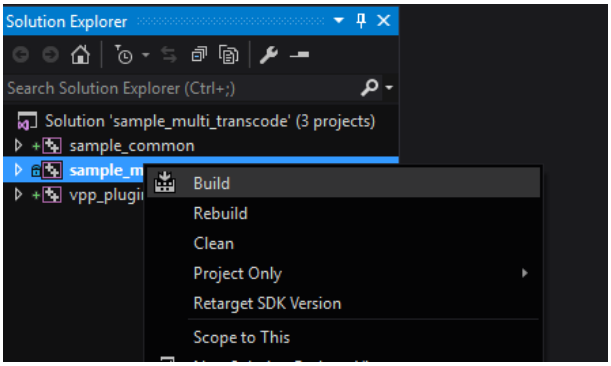
Open File->Open Project, choose:

C:\Users\zhaoye\Downloads\svet_e2e_sample_w\svet_e2e_sample_w\video_e2e_sample\sample_multi_transcode.vcxproj

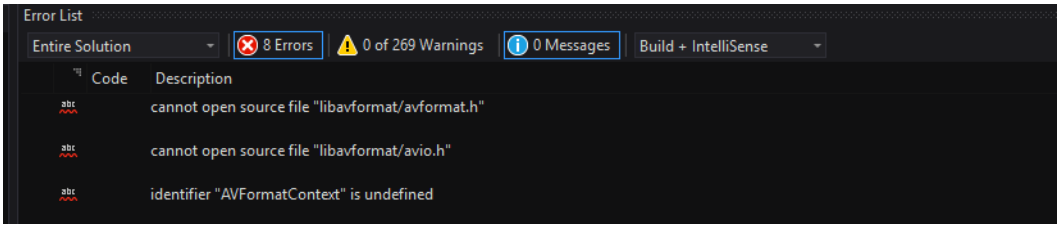
If the project load successes, it will look like below.



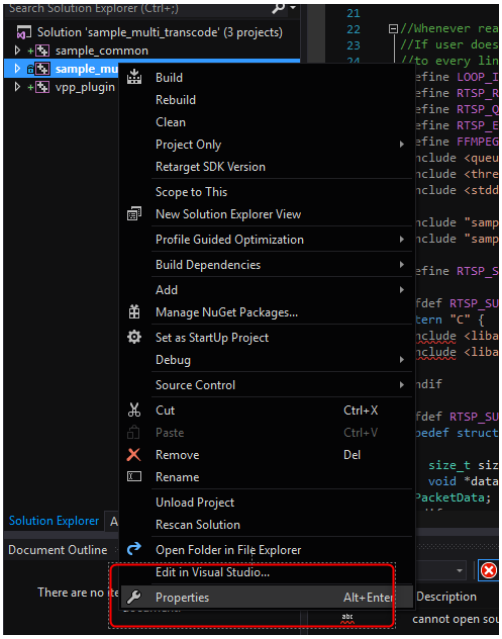
Choose “Release” and “X64”, and Right Click “Sample_multi_transcode”, choose Build, it will start building.



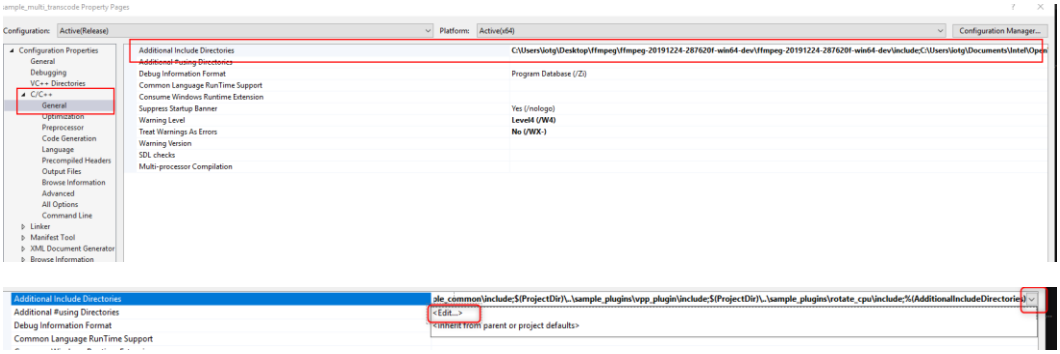
You may meet some issue like below, “cannot find source file “libavformat/avformat.h”



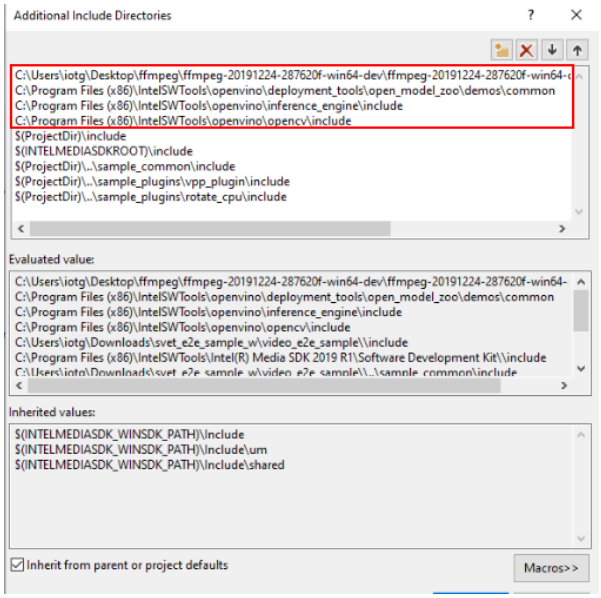
You should check the Additional Include Directories, by right clicking. “sample_multi_transcode” and choose “properties”, select C/C++->General->Additional Include Directories.



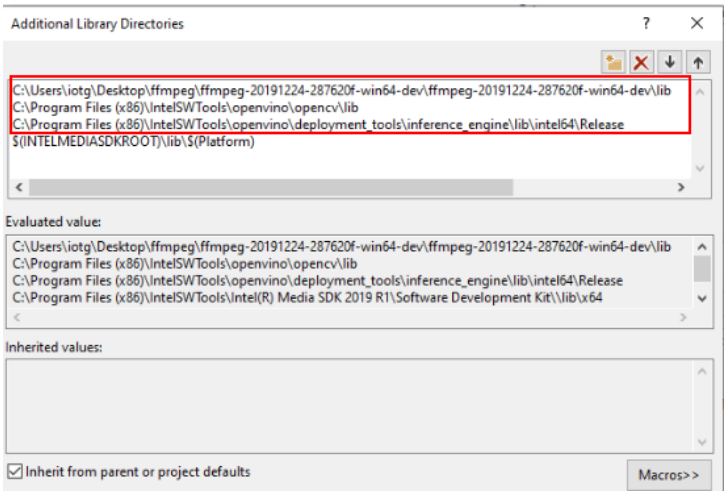
Additional Include Directories looks like below:



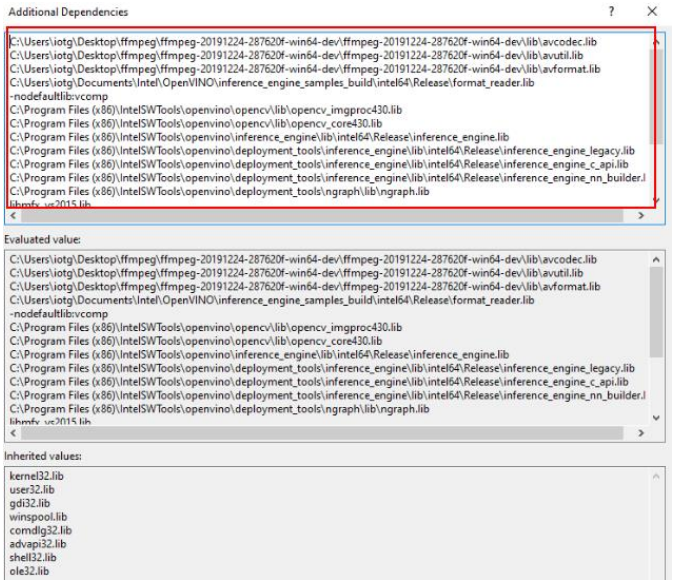
Click the down arrow at the end and Click the <Edit..> button, you will see blow dialog, please make sure, below Include folder can be found on your PC.



And the same thing for the “Additional Library Directories” and “Additional Dependencies” in Linker->General->Additional Library Directories and Linker->Input->Additional Dependencies.



Additional Dependencies, click the down arrow and <Edit..>, then a dialog jump out, it looks like below:



1.4 Prepare inference model

Download inference IR file through script\download_IR_file.py

You can execute below script in windows console (requires full administrative rights):

```
Cd svet_e2e_sample_w\script

python download_IR_file.py
```

you will find all the inference IR file in folder svet_e2e_sample_w\script\:

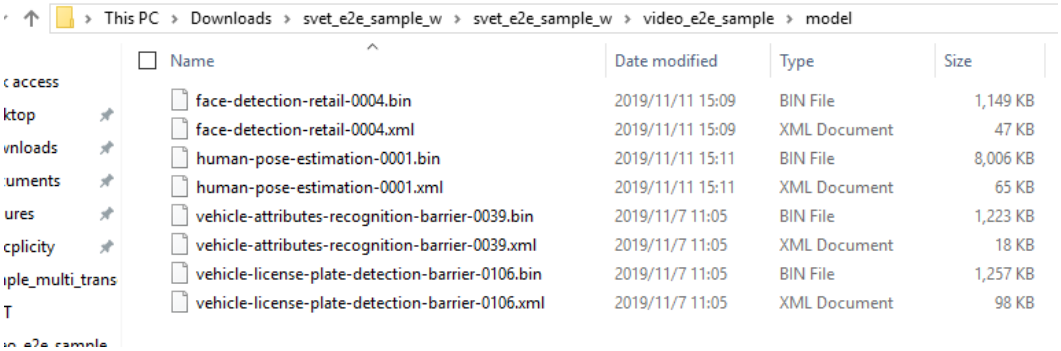
Make a “model” folder in C:\Users\zhaoye\Downloads\svet_e2e_sample_w\svet_e2e_sample_w\video_e2e_sample

And copy the inference IR file to it (take human pose estimation as example).

```
cp svet_e2e_sample_w\script\*.bin C:\Users\zhaoye\Downloads\svet_e2e_sample_w\svet_e2e_sample_w\video_e2e_sample\model\

cp svet_e2e_sample_w\script\*.xml C:\Users\zhaoye\Downloads\svet_e2e_sample_w\svet_e2e_sample_w\video_e2e_sample\model\
```

below IR file is needed for later test.



1.5 Prepare the video clips for testing

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

```
ffmpeg -i classroom.mp4 -c:v libx264 -an -bsf:v h264_mp4toannexb -r 30 -g 60 -b:v 4000k -bf 0 classroom.h264
```

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

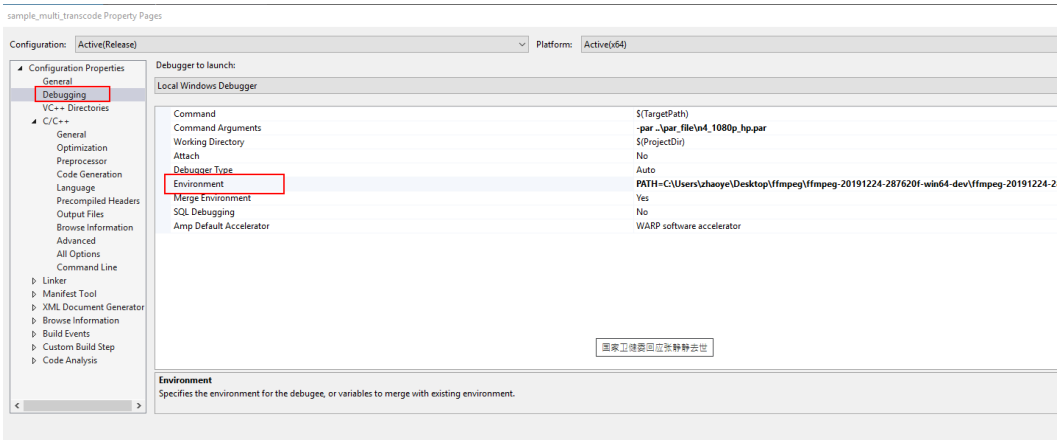
If you want to do performance test, we suggest you to format the video clips bitrate as 4Mbps. if it's 1080P video stream, GOP size as 60, reference frame as 1, no B frames, FPS as 30. Then you can get a test result near our test result.

2.0 Run SVET sample application

2.1 Check environment variables

Setting below debugging environment path to Debugging->Environment.

PATH= C:\Users\zhaoye\Desktop\ffmpeg\ffmpeg-20191224-287620f-win64-shared\ffmpeg-20191224-287620f-win64-shared\bin;C:\Program Files (x86)\IntelSWTools\openvino\deployment_tools\inference_engine\bin\intel64\Release;%PATH%



If you are running the application in console, you should set the environment path by:

```
set PATH= C:\Users\zhaoye\Desktop\ffmpeg\ffmpeg-20191224-287620f-win64-shared\ffmpeg-20191224-287620f-win64-shared\bin;C:\Program Files (x86)\IntelSWTools\openvino\deployment_tools\inference_engine\bin\intel64\Release;%PATH%
```

2.2 Modify the video path in parameter file

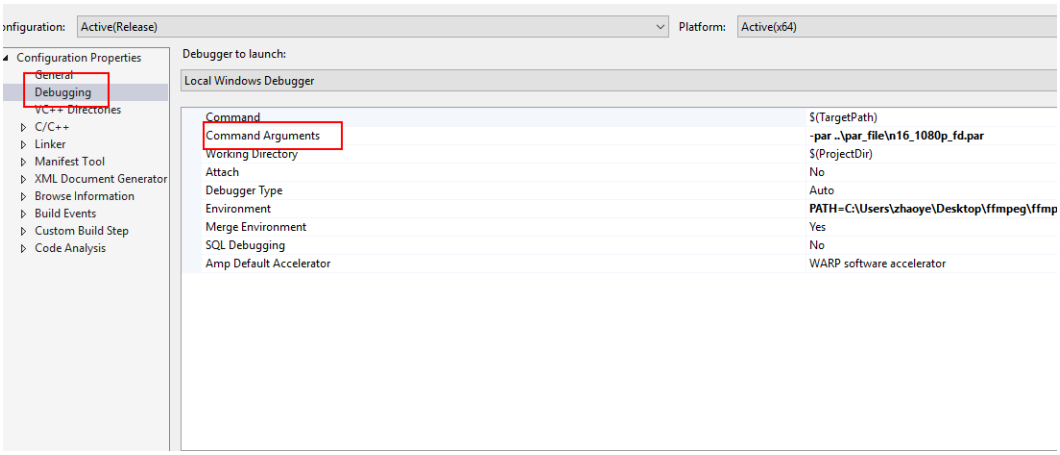
Modify the video path (following "-i::h264") of **every line** in example par file under C:\Users\zhaoye\Downloads\svet_e2e_sample_w\svet_e2e_sample_w\par_file\n4_1080p_hp.par.

```
-i::h264 content\2.h264 -dc::rgb4 -join -hw_d3d11 -threads 3 -async 10  -timeout 3600 -o::sink -vpp_comp_dst_x 0 -v
-i::h264 content\2.h264 -dc::rgb4 -join -hw_d3d11 -threads 3 -async 10  -timeout 3600 -o::sink -vpp_comp_dst_x 960
-i::h264 content\2.h264 -dc::rgb4 -join -hw_d3d11 -threads 3 -async 10  -timeout 3600 -o::sink -vpp_comp_dst_x 0 -v
-i::h264 content\2.h264 -dc::rgb4 -join -hw_d3d11 -threads 3 -async 10  -timeout 3600 -o::sink -vpp_comp_dst_x 960

-vpp_comp_only 4 -w 1920 -h 1080 -async 10 -join -hw_d3d11 -i::source -ext_allocator
```

Set the par file as Command Arguments:

-par ..\par_file\n16_1080p_fd.par



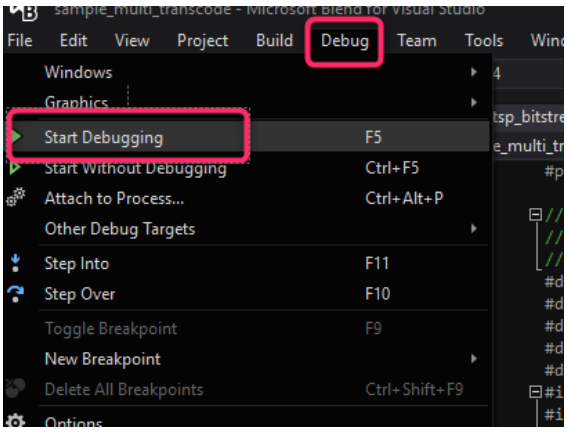
If run the application in console, you should run below command:

```
cd C:\Users\zhaoye\Downloads\svet_e2e_sample_w\svet_e2e_sample_w\video_e2e_sample

..\_build\x64\Release\video_e2e_sample.exe -par ..\par_file\n4_1080p_fd.par
```

2.3 Run video_e2e_sample application

Choose the Debug->Start Debugging, Then the application can run.



When running 16-channel inference, loading of inference models can take long time. due to the run time building of OpenCL kernel for each model. It's recommended to enable OpenCL kernel cache by make a "cl_cache" folder in C:\Users\zhaoye\Downloads\svet_e2e_sample_w\svet_e2e_sample_w\video_e2e_sample\ Then only one time OpenCL kernel building is required, after that, the built kernel can be loaded from cl_cache directory.

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

Command:

```
cd C:\Users\zhaoye\Downloads\svet_e2e_sample_w\svet_e2e_sample_w\video_e2e_sample

..\_build\x64\Release\video_e2e_sample.exe -par ..\par_file\n4_1080p_fd.par
```

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

If you want to stop the application, close the console.

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

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

Command:

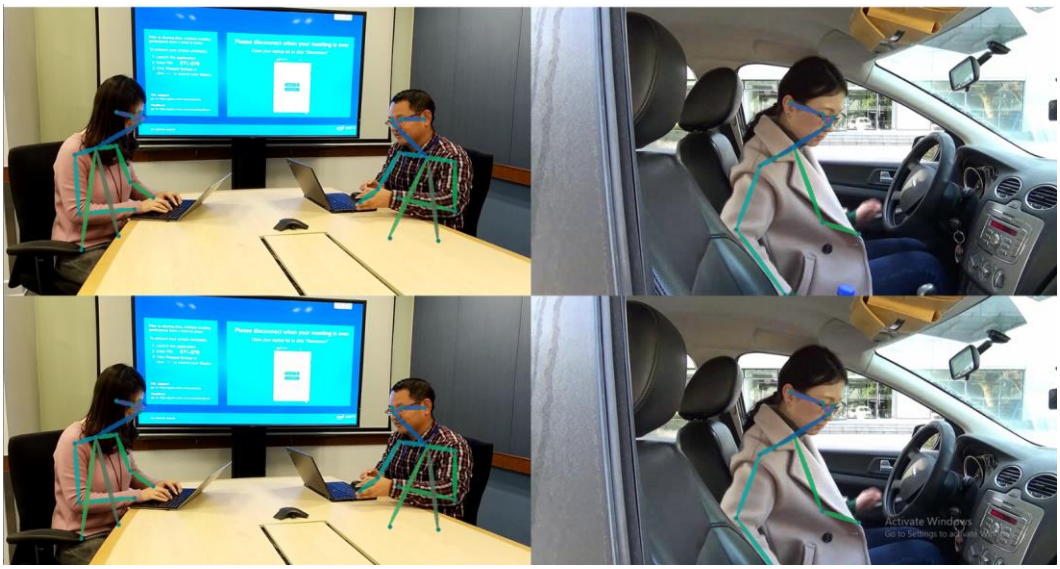
```
cd C:\Users\zhaoye\Downloads\svet_e2e_sample_w\svet_e2e_sample_w\video_e2e_sample

..\_build\x64\Release\video_e2e_sample.exe -par ..\par_file\n4_1080p_hp.par
```

Replace the "-par ..\par_file\n16_1080p_fd.par" to "-par ..\par_file\n4_1080p_hp.par"

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

Below picture is the screenshot of this demo.



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

Command:

```
cd C:\Users\zhaoye\Downloads\svet_e2e_sample_w\svet_e2e_sample_w\video_e2e_sample

..\_build\x64\Release\video_e2e_sample.exe -par ..\par_file\n4_1080p_vd.par
```

Replace the "-par ..\par_file\n16_1080p_fd.par" to "-par ..\par_file\n4_1080p_vd.par"

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

Below picture is the screenshot of this demo.

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

Command:

```
cd C:\Users\zhaoye\Downloads\svet_e2e_sample_w\svet_e2e_sample_w\video_e2e_sample

..\_build\x64\Release\video_e2e_sample.exe -par ..\par_file\n4_1080p_rtsp.par
```

Replace the "-par ..\par_file\n16_1080p_fd.par" to "-par ..\par_file\n4_1080p_rtsp.par"

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

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

2.3.5 16-channel RTSP video decoding, RTSP stream storing, face detection, composition, encode, and display

Command:

```
cd C:\Users\zhaoye\Downloads\svet_e2e_sample_w\svet_e2e_sample_w\video_e2e_sample

..\_build\x64\Release\video_e2e_sample.exe -par ..\par_file\n16_1080p_rtsp.par
```

Replace the "-par ..\par_file\n16_1080p_fd.par" to "-par ..\par_file\n16_1080p_rtsp.par"

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

2.3.6 HDDL inference, face detection and GPU composition, encode, display

To perform inference on Intel® Vision Accelerator Design with Intel® Movidius™ VPUs, the following additional installation steps are required:

1. If your Intel® Vision Accelerator Design with Intel® Movidius™ VPUs card requires SMBUS connection to PCIe slot (Raw video data card with HW version Fab-B and before), install the SMBUS driver:
    a. Go to the <INSTALL_DIR>\deployment_tools\inference-engine\external\hddl\SMBusDriver directory, where <INSTALL_DIR> is the directory in which the Intel Distribution of OpenVINO toolkit is installed.
    b. Right click on the hddlsmbus.inf file and choose **Install** from the pop up menu.

Command:

```
cd C:\Users\zhaoye\Downloads\svet_e2e_sample_w\svet_e2e_sample_w\script

Run_FD_HDDL_test.bat
```

If you want to enable HDDL human pose detection or vehicle detection, you need to change the par_file\n4_1080p_fd_HDDL.par

Replace the “-infer::fd” to “-infer::hp” or “-infer::vd”.

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

2.4.1 New parameters in Par file

Comparing to original sample_multi_transcode, we add some new parameters.

Parameter	**Usage**
-infer::infer_type ir_file_dir	Specify the inference type and directory that stores the IR files. Examples: -infer::fd ./model →face detection -infer::hp ./model →human pose estimation -infer:vd ./model →vehicle and vehicle attributes detection
-infer::Device_type	-infer::GPU →Running inference workload on GPU -infer::HDDL →Running inference workload on HDDL card

-i::h264 rtsp://url	Specify the source H264 file with RTSP URL
-rtsp_save filename.h264	Save RTSP stream to local file. This parameter must be used together with "-i::h264 rtsp://url". If the whole line of session parameters only contains "-i::h264 rtsp://url -rtsp_save filename.h264" and don't have other decoding parameters, we call such sessions as RTSP stream storing session and they must be put into a separated par file.

2.4.2 Decode, encode and display parameters

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

Parameter	Usage
-i::h264 | h264 input_video_filename	Set input file and decoder type
-o::h264 | h265 output_video_filename	Set output file and decoder type
-o::sink	The output will be passed to the sink sessions,, e.g. encoding session or composition session
-i::source	The input is coming from source sessions like decoding session
-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_d3d11	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.