NVIDIA Video Effects

System Guide

DA-10469-001_v0.7.2 | January 2023

Table of Contents

Chapter 1. NVIDIA Video Effects SDK for Windows

The NVIDIA® Video Effects SDK for Windows is used to apply effect filters to videos.

The SDK is powered by NVIDIA GPUs with Tensor Cores. With Tensor Cores, algorithm throughput is greatly accelerated, and latency is reduced.

The SDK provides the following filters:

- **AI green screen (video background segmentation)**, which segments and masks the background areas in a video or image.
- **Background Blur (Beta)**, which uses the segmentation mask from the AI green screen filter, or other sources, and produces a blur effect in the background, in a video, or in an image.
- **Encoder Artifact Reduction (Beta)**, which reduces the blocking and noisy artifacts that are produced from encoding while preserving the details of the original video.
- The ArtifactReduction effect has the following modes:
  - Strength 0, which applies a weak effect.
  - Strength 1, which applies a strong effect.
- **Super resolution (Beta)**, which upscales a video and reduces encoding artifacts.

  This filter enhances the details, sharpens the output, and preserves the content. The SuperRes effect has two modes:
  - Strength 1, which applies strong enhancements.
  - Strength 0, which applies weaker enhancements while reducing encoding artifacts.
- **Upscale (Beta)**, which is a fast and light-weight method to upscale for an input video and sharpen the resulting output.

  This filter can optionally be pipelined with encoder artifact reduction to enhance the scale while reducing the video artifacts.
- **Webcam Denoising (Beta)**, which removes noise from a webcam video while preserving the texture details.

  This effect has the following modes:
  - Strength 0, which applies a weak effect.
  - Strength 1, which applies a strong effect.

1.1. Getting Started with the Video Effects SDK for Windows

The Video Effects SDK requires specific NVIDIA GPUs and a specific version of Windows OS and other associated software on which the SDK depends.

This SDK is designed and optimized for client-side application integration and for local deployment. We do not officially support the testing, experimentation, deployment of this SDK to a datacenter/cloud environment.

1.1.1. Hardware Requirements

The Video Effects SDK is compatible with GPUs that are based on the NVIDIA Turing™ or the NVIDIA Ampere™, or the NVIDIA Ada™ architecture and have Tensor Cores.

1.1.2. Software Requirements

The Video Effects SDK requires a specific version of Windows OS and other associated software on which the SDK depends.

Table 1. Software Requirements for Windows

Software	Required Version
Windows OS	64-bit Windows 10 or later
Microsoft Visual Studio	2017 or later
CMake	3.12 or later
NVIDIA Graphics Driver for Windows	511.65 or later

1.2. Installing the Video Effects SDK for Windows

Here is information about installing the SDK for Windows.

The SDK is distributed in the following forms:

- The *development* SDK package.

  The development package includes everything in the SDK including the API headers, runtime dependencies, and sample apps.
- The *redistributable* SDK package.

  The redistributable package is more convenient if your application only wants to integrate SDK API headers and ask end users to download and install the SDK runtime dependencies.

To develop applications, because the essential contents in these two packages are the same, you can use either package.

The redistributable SDK package comprises the following parts:

- An open-source repository that includes the SDK API headers, the sample applications and their dependency libraries, and a proxy file to enable compilation without the SDK DLLs.
- An installer that installs the following SDK runtime dependencies:
  - The DLLs
  - The models
  - The SDK dependency libraries

The install location is the `C:\Program Files\NVIDIA Corporation\NVIDIA Video Effects\` directory.

For an application that is built on the SDK, the developer can package the runtime dependencies into the application or require application users to use the SDK installer.

Note: The source code and sample applications are in the development package and are hosted on GitHub at https://github.com/nvidia/MAXINE-VFX-SDK.

To use the SDK redistributable package, download the source code from GitHub and install the SDK binaries.

Note: If you are using a development package, you can ignore this step.

The sample app source code demonstrates how to integrate API headers and call the SDK APIs. The sample app also includes the `NVVideoEffectsProxy.cpp` file that is used to link against the SDK DLL without requiring an import library (.lib) file. With this file, you can compile the open-source code independently of the SDK installer. However, the SDK runtime dependencies are still required to load the runtime dependencies, the DLLs, and models.

1.3. Building the Sample Applications for Windows

To demonstrate the features of the Video Effects SDK, the SDK provides sample applications as source code that you can build and as binary files that you can run without building. You can run each application from the supplied batch file or from the application binary file.

The open source repository includes the source code to build the sample application and a `NVVideoEffectsProxy.cpp` proxy file to enable compilation without explicitly linking against the SDK DLL.

Note: To download the models and runtime dependencies that are required by the features, run the SDK Installer.

1. In the root folder of the downloaded source code, start the CMake GUI and specify the source folder and a build folder for the binary files.
   a). For the source folder, ensure that the path ends in `OSS`.
   b). For the build folder, ensure that the path ends in `OSS/build`.
   c). Use CMake to configure and generate the Visual Studio solution file.
   d). Click **Configure**.
   e). When prompted to confirm that CMake can create the build folder, click **OK**.
   f). Select **Visual Studio** for the generator and x64 for the platform.
   g). To complete configuring the Visual Studio solution file, click **Finish**.
   h). To generate the Visual Studio Solution file, click **Generate**.
2. Use Visual Studio to generate the application binary.exe file from the solution file that you generated in the previous step.
3. In CMake, to open Visual Studio, click **Open Project**.
4. In Visual Studio, select **Build** > **Build Solution** .

1.4. Running the Sample Applications on Windows

This section provides information about how to run sample applications on Windows.

1.4.1. Running the AI Green Screen Application

The AI green screen application (`AigsEffectApp.exe`) demonstrates the AI green screen (video background segmentation) feature of the SDK.

The application accepts either a video file or output from a webcam as input and produces video output, which can be stored in a file or displayed in a window.

The application produces the output initially with foreground pixels highlighted. When the `--show` argument is specified, the keys described in Keyboard Controls toggle the behavior of the application.

You can run this application from the supplied batch file or from the application binary file.

When the appropriate `--comp_mode` value is passed via command line, the `AigsEffectApp` sample app also demonstrates the background blur effect.

For more ways to control the application, see the following:

- AI Green Screen Application Command-Line Reference
- Keyboard Controls

1.4.2. Running the UpscalePipeline Application

A script file, run.bat, is provided to show you how to run the UpscalePipeline application. This script file is in the `samples\UpscalePipelineApp` folder, which is under the Video Effects SDK root folder.

Note: This batch file is called `run.bat` in the redistributable installer package and is called run_local.bat in the developer package.

This file adds the paths to the required DLL files to the Path system variable and sets the options required to run the application. The input and output have been set as images in the samples/input folder, but you can set them as images or videos (MP4).

1. Open a Command Prompt window.
2. Navigate to the `samples\UpscalePipelineApp` folder.
   ```
   cd root-folder\samples\UpscalePipelineApp
   ```
3. Execute the `run.bat` or the `run_local.bat` file.

For more ways to control the application, see the following:

- UpscalePipeline Application Command-Line Reference
- Keyboard Controls

1.4.3. Running the DenoiseEffect Application

A script file, `run.bat`, is provided to show you how to run the `DenoiseEffect` application. This script file is in the `samples\DenoiseEffectApp` folder, which is under the Video Effects SDK root folder.

Note: This batch file is called `run.bat` in the redistributable installer package and is called `run_local.bat` in the developer package.

This file adds the paths to the required DLL files to the Path system variable and sets the options that are required to run the application. The input is currently set to a webcam stream and the output to a video file, but you can set them as images or as videos (MP4). When the sample app is running, you can toggle the effect on and off by pressing the **E** key.

1. Open a Command Prompt window.
2. Navigate to the `samples\DenoiseEffectApp` folder.
   ```
   cd root-folder\samples\DenoiseEffectApp
   ```
3. Execute the `run.bat` or the `run_local.bat`, file.

For more ways to control the application, see the following:

- DenoiseEffect Application Command-Line Reference
- Keyboard Controls

1.4.4. Running the VideoEffects Application

A script file, `run.bat`, is provided to show you how to run the `VideoEffects` application. This script file is in the `samples\VideoEffectsApp` folder, which is under the Video Effects SDK root folder.

The `VideoEffects` application can be used to demonstrate the following features:

- Artifact reduction
- Super Resolution
- Upscale

Note: This batch file is called `run.bat` in the redistributable installer package and is called `run_local.bat` in the developer package.

This file adds the paths to the required DLL files to the Path system variable and sets the options that are required to run the application. The input and output have been set as images in the `samples/input` folder, but you can set them as images or as videos (MP4).

1. Open a Command Prompt window.
2. Navigate to the samples\VideoEffectsApp folder.
   ```
   cd root-folder\samples\VideoEffectsApp
   ```
3. Execute the `run.bat` or the `run_local.bat` file.

For more ways to control the application, see the following:

- VideoEffects Application Command-Line Reference
- Keyboard Controls

1.5. Performance reference on Windows

This table lists performance data for features of the NVIDIA Video Effects SDK on a few supported GPU architectures.

Table 2.

	Latency (in milliseconds)		
Feature	**RTX 2060**	**RTX 3090**	**RTX 4090**
Super Resolution	3.5	1.57	0.52
Webcam Denoising	2.1	0.94	0.43
Upscale	0.17	0.07	0.03
Encoder Artifact Reduction	3.42	1.25	0.66

	Latency (in milliseconds)		
AI Green Screen	2.44	2.08	1.3

Note: Default resolution for effects is 720p.For Super Resolution and Upscale features, the data reflects performance for 2x scaling, from 360p to 720p.

Chapter 2. NVIDIA Video Effects SDK for Linux

The Video Effects SDK for Linux is used to apply effect filters to videos.

The SDK is powered by NVIDIA GPUs with Tensor Cores. With Tensor Cores, algorithm throughput is greatly accelerated, and latency is reduced.

The SDK provides the following filters:

- **AI green screen (video background segmentation)**, which segments and masks the background areas in a video or image.
- **Background Blur (Beta)**, which uses the segmentation mask from the AI green screen filter, or other sources, and produces a blur effect in the background, in a video, or in an image.
- **Encoder Artifact Reduction (Beta)**, which reduces the blocking and noisy artifacts that are produced from encoding while preserving the details of the original video.
- The ArtifactReduction effect has the following modes:
  - Strength 0, which applies a weak effect.
  - Strength 1, which applies a strong effect.
- **Super resolution (Beta)**, which upscales a video and reduces encoding artifacts.

  This filter enhances the details, sharpens the output, and preserves the content. The SuperRes effect has two modes:
  - Strength 1, which applies strong enhancements.
  - Strength 0, which applies weaker enhancements while reducing encoding artifacts.
- **Upscale (Beta)**, which is a fast and light-weight method to upscale for an input video and sharpen the resulting output.

  This filter can optionally be pipelined with encoder artifact reduction to enhance the scale while reducing the video artifacts.
- **Webcam Denoising (Beta)**, which removes noise from a webcam video while preserving the texture details.

  This effect has the following modes:
  - Strength 0, which applies a weak effect.
  - Strength 1, which applies a strong effect.

2.1. Getting Started with the Video Effects SDK on Linux

The Video Effects SDK requires specific NVIDIA GPUs, specific versions of the Linux OS, and other associated software on which the SDK depends.

This SDK is designed and optimized for server-side (datacenter/cloud) deployment. We do not officially support the testing, experimentation and production deployment of this SDK to client-side application integration and local deployment.

2.1.1. Hardware Requirements

The Video Effects SDK is compatible with GPUs that are based on NVIDIA Turing™, NVIDIA Volta™, or the NVIDIA Ampere™ architecture.

Note: For best performance with NVIDIA T4 and other server GPUs, make sure that you use a server that meets the thermal and airflow requirements for these types of products. Refer to https://www.nvidia.com/en-us/data-center/tesla/tesla-qualified-servers-catalog/ for the latest list of qualified servers.

2.1.2. Software Requirements

NVIDIA Video Effects SDK requires a specific version of Linux and other associated software on which the SDK depends.

Table 3. Software Requirements for Linux

Software	Required Version
Linux	Ubuntu 18.04, Ubuntu 20.04, or CentOS 7
CUDA	11.8.0
TensorRT	8.5.1.7 (for NVIDIA CUDA® 11.8)
cudnn	8.6.0.163 (for the appropriate NVIDIA TensorRT™/ CUDA versions)
CMake	3.10
opencv	3.2+ or 4.x (for sample apps only)
NVIDIA Graphics Driver for Linux	520.61 or later

2.2. Installing the NVIDIA Video Effects SDK for Linux

The SDK is delivered as a .tar.gz package, which is a compressed tar format.

The package contains the video effects library and header files, which need to be extracted to the `/usr/local/VideoFX` directory.

You can download the prerequisites from https://developer.nvidia.com.

Table 4. Downloading the Prerequisites

Prerequisite	Download location
CUDA	https://developer.nvidia.com/cuda
TensorRT	https://developer.nvidia.com/tensorrt
cudnn	https://developer.nvidia.com/cudnn

See Software Requirements for more information about the version numbers for <version> below).

- To install CUDA (include graphics driver):

```
$ sudo sh cuda_<version>_linux.run
```

- To install TensorRT:

```
$ sudo tar -xvf TensorRT-<version>.linux.x86_64-gnu.cuda-
<version>.cudnn<version>.tar.gz -C /usr/local
```

- To install cudnn:

```
$ tar -xvf cudnn-linux-x86_64-<version>_cuda11-archive.tar.xz
$ sudo cp cudnn-linux-x86_64-<version>_cuda11-archive/include/cudnn*.h
/usr/local/cuda/include
$ sudo cp -P cudnn-linux-x86_64-<version>_cuda11-archive/lib/libcudnn*
/usr/local/cuda/lib64
$ sudo chmod a+r /usr/local/cuda/include/cudnn*.h
/usr/local/cuda/lib64/libcudnn*
```

- To install the SDK:

```
$ sudo tar -xvf NVIDIA_VFX_SDK_<OS>_version>.tar.gz -C /usr/local
```

To install CUDA 11.8 and preserve older (supported) graphics drivers, such as versions 418 and 440, you need to carefully install the CUDA toolkit and the CUDA compatibility package. Refer to CUDA Compatibility and the `README_quickstart.md` file for more information.

Note: If you plan to use an older (supported) graphics driver, you must set `LD_LIBRARY_PATH` to include the compatibility package by running the following command:

```
export LD_LIBRARY_PATH=/usr/local/cuda/compat:$LD_LIBRARY_PATH
```

2.3. Building the Sample Applications on Linux

To demonstrate the features of the Video Effects SDK, the SDK provides sample applications as source code that you can build. You can then run each application from the supplied shell script or the application binary file.

To build the sample applications, run the following command:

```
/usr/local/VideoFX/share/build_samples.sh
```

Follow the prompts to provide an install location and build the sample apps. The script might prompt you to install the necessary prerequisites.

2.4. Running the Sample Applications on Linux

This section provides information about how to run the sample applications in Linux.

2.4.1. Running the AI Green Screen Application from the Shell Script

The AI green screen application, `AigsEffectApp`, demonstrates the AI green screen (video background segmentation) feature of the SDK. The application accepts an image, a video file, or the output from a webcam as input and produces video output, which can be stored in a file or displayed in a window.

The `~/mysamples` folder contains the following shell scripts:

- `run_aigs_webcam.sh`
- `run_aigs_image.sh`

To run the AI green screen sample application on a sample image that is included in the SDK, run the following command:

```
$ cd ~/mysamples
$ ./run_aigs_image.sh
```

This step creates an output image called `aigs_image_out.jpg`.

To run the AI green screen sample application by using a connected webcam that displays on the screen, run the following command:

```
$ cd ~/mysamples
$ ./run_aigs_webcam.sh
```

Note: To run the application directly from the command line, review the contents of the shell script.

For more ways to control the application, see the following:

- AI Green Screen Application Command-Line Reference
- Keyboard Controls

2.4.2. Running the UpscalePipeline Application from the Shell Script

Here is information about how to run the UpscalePipeline application on Linux.

To run the UpscalePipeline sample application on the sample images in the SDK, run the following command:

```
$ cd ~/mysamples
$ ./run_upscalepipeline.sh
```

Note: To run the application directly from the command line, review the contents of the shell script.

For more ways to control the application, see the following:

- UpscalePipeline Application Command-Line Reference
- Keyboard Controls

2.4.3. Running the DenoiseEffect Application from the Shell Script

The `~/mysamples` folder contains the `run_denoiseeffect.sh` shell script.

To run the DenoiseEffect sample application on the sample images in the SDK, run the following command:

```
./run_denoiseeffect.sh
```

The input is currently set to a webcam stream and the output to a video file, but you can set them as images or videos. When the sample app is running, you can toggle the effect on and off by pressing the **E** key.

Note: To run the application directly from the command line, review the contents of the shell script.

For more ways to control the application, see the following:

- DenoiseEffect Application Command-Line Reference
- Keyboard Controls

2.4.4. Running the Batch Effect Application from the Shell Script

Some of the effects can take advantage of batching to achieve higher performance. The `BatchEffectApp` contains code that illustrates the extra steps that are needed to

simultaneously process a batch of images. Unlike the other sample applications, this application accepts multiple images to be used as input.

Note: The `~/mysamples` folder contains the `run_batcheffect.sh` shell script.

To run the BatchEffectApp on a batch of sample images, run:

```
$ cd ~/mysamples
$ ./run_batcheffect.sh
```

Note: To run the application directly from the command line, review the content of the shell script.

For more ways to control the application, see the following:

- BatchEffectApp Command-Line Reference
- Keyboard Controls

2.4.5. Running the Batch Denoise Effect Application from the Shell Script

The Batch Denoise Effect Application can be used to denoise frames from multiple video streams in user-specified batches. `BatchDenoiseEffectApp` contains code that illustrates the extra steps that are needed to simultaneously process multiple video streams. The app assumes that the input videos have the same resolution and length. Unlike the other sample applications, this application accepts multiple videos to be used as input.

Note: The `~/mysamples` folder contains the `run_batchdenoiseeffect.sh` shell script.

To run the `BatchDenoiseEffectApp` on a batch of sample images, run:

```
$ cd ~/mysamples
$ ./run_batchdenoiseeffect.sh inFile1 [ ... inFileN ]
```

This process denoises the videos in the N input files, `inFile1` to `inFileN` and writes the output to the N corresponding output files.

Note: Currently sample input videos for batch denoising are not supplied with the SDK.

2.4.6. Running the Batch Aigs Effect Application from the Shell Script

The Batch Aigs Effect Application can be used to run the effect on frames from multiple video streams batches, and the batch size is equal to the number of videos. BatchAigsEffectApp contains code that illustrates the extra steps that are needed to simultaneously process multiple video streams. The app assumes that the input videos have the same resolution and

length. Unlike the other sample applications, this application accepts multiple videos to be used as input.

Note: The `~/mysamples` folder contains the `run_batchaigseffect.sh` shell script.

To run the BatchAigsEffectApp on the sample videos, run:

```
$ cd ~/mysamples
$ ./run_batchaigseffect.sh
```

This process denoises the videos in the two input files and writes the output to the two corresponding output files.

2.4.7. Running the VideoEffects Application from the Shell Script

Here is some information about running the `VideoEffects` application. The `~/mysamples` folder contains the `run_videoeffects.sh` shell script.

To run the `VideoEffects` sample application on the sample images in the SDK, run the following command:

```
./run_videoeffects.sh
```

For more ways to control the application, see the following:

- VideoEffects Application Command-Line Reference
- Keyboard Controls

Note: To run the application directly from the command line, review the contents of the shell script.

2.5. Performance reference on Linux

This table lists performance data for features of the NVIDIA Video Effects SDK on a few supported GPU architectures.

Table 5.

	Latency (in milliseconds)			Throughput (max streams to achieve 30 FPS)		
Feature	**T4**	**A10**	**A40**	**T4**	**A10**	**A40**
Super Resolution	3.16	1.43	1.1	10	23	30
Webcam Denoising	2.64	1.09	0.88	14	30	30
Upscale	0.21	0.09	0.07	157	367	471

	Latency (in milliseconds)			Throughput (max streams to achieve 30 FPS)		
Encoder Artifact Reduction	4.4	1.65	1.2	8	20	28
AI Green Screen	2.4	1.3	1.17	14	25	28

Note: Default resolution for effects is 720p.For Super Resolution and Upscale features, the data reflects performance for 2x scaling, from 360p to 720p. Throughput is defined as the maximum number of streams to achieve 30FPS real-time.

Chapter 3. Sample Applications Reference

To demonstrate the features of the Video Effects SDK, the SDK provides sample applications as source code that you can build and as binary files that you can run without building. You can run each application from the supplied shell script or the application binary file.

Important: To quit the application, you cannot just close the window. You need to enter Q or Escape.

3.1. AI Green Screen Application

This section provides command-line and keyboard control information for the AI green screen application.

3.1.1. AI Green Screen Application Command-Line Reference

Here is the command-line reference information for the AI green screen application.

```
AigsEffectApp [arguments...]
```

Here are the arguments:

--in_file=path
The image file or video file for the application to process.

--webcam[=(true|false)]
If true, use a webcam as input instead of a file.

--cam_res=[widthx]height

If `--webcam` is `true`, specify the resolution of the webcam, and *width* is optional. If omitted, *width* is computed from *height* to give an aspect ratio of 16:9. For example:

```
--cam_res=1280x720 or --cam_res=720
```

If `--webcam` is `false`, this argument is ignored.

--out_file=path
The file in which the video output is to be stored.

--show[=(true|false)]
If true, display the resulting video output in a window.

--model_dir=path
The path to the folder that contains the model files to be used for the transformation.
--codec=fourcc
The four-character code (FOURCC) of the video codec of the output video file. The default is H264.
--help
Display help information for the command.
--mode=(0|1)
Selects the mode in which to run the application:

- 0 selects the best quality.
- 1 selects the fastest performance.

--comp_mode=(0|1|2|3|4|5|6)
Where mode selects which composition mode to use:

- **0**: Displays the segmentation mask (compMatte).
- **1**: Overlays the mask on top of the image (compLight).
- **2**: Provides a composition with a BGR={0,255,0} background image (compGreen).
- **3**: Provides a composition with a BGR={255,255,255} background image (compWhite).
- **4**: No composition but displays the input image (compNone).
- **5**: Overlays the mask on the image (compBG).
- **6**: Applies a background blur filter on the input image by using the segmentation mask (`compBlur`).

3.1.2. Keyboard Controls

Here is the keyboard control information for the AI green screen application.

The sample application provides keyboard controls to change the run-time behavior of the application:

- **C**: Toggles between the following ways of rendering the image:
  - **0**: Displays the segmentation mask (`compMatte`).
  - **1**: Overlays the mask on top of the image (`compLight`).
  - **2**: Provides a composition with a `BGR={0,255,0}` background image (compGreen).
  - **3**: Provides a composition with a `BGR={255,255,255}` background image (compWhite).
  - **4**: No composition but displays the input image (`compNone`).
  - **5**: Overlays the mask on the image (`compBG`).
  - **6**: Applies a background blur filter on the input image by using the segmentation mask (`compBlur`).
- **F**: Toggles the frame rate display on and off.
- **Q** or **Escape**: exits the app and cleanly finishes writing any output file.

3.2. UpscalePipeline Application

This section provides command-line and keyboard control information for the UpscalePipeline application.

3.2.1. UpscalePipeline Application Command-Line Reference

Here is the command-line reference information for the UpscalePipeline application.

```
UpscalePipelineApp [arguments...]
```

Here are the arguments:

--in_file=path
The image file or video file for the application to process.

--out_file=path
The file in which the video output is to be stored.

--resolution=NNN
The output image/video vertical resolution, which is scaled from the input vertical resolution by 1.3333, 1.5, 2, 3, or 4.

--show[=(true|false)]
If true, displays the resulting video output in a window.

--model_dir=path
The path to the folder that contains the model files that will be used for the transformation.

--codec=fourcc
The four-character code (FOURCC) of the video codec of the output video file. The default is H264.

--ar_mode=(0|1) for ArtifactReduction
Selects the strength of the filter to be applied:

- 0 selects a weak effect.
- 1 selects a strong effect.

--upscale_strength=[0.0-1.0] for Upscale
Selects the strength of the upscale filter to be applied:

- 0 selects no enhancement.
- 1 selects the maximum crispness.

The default value is 0.4.

--progress
Show the progress.

--verbose [=(true|false)]
Verbose output.

--debug [=(true|false)]
Prints extra debugging information.

--help
Displays help information for the command.

3.2.2. Keyboard Controls

Here is the keyboard control information for the UpscalePipeline application.

The sample application provides keyboard controls for changing the run-time behavior of the application:

- **F**: Toggles the frame rate display on and off.
- **Q** or **Escape**: exits the app and cleanly finishes writing any output file.

3.3. DenoiseEffect Application

This section provides command-line and keyboard control information for the DenoiseEffect application.

3.3.1. DenoiseEffect Application Command-Line Reference

Here is the command-line reference information for the DenoiseEffect application.

```
DenoiseEffectApp [arguments...]
```

Here are the arguments:

--in_file=path
The image file or video file for the application to process.
--out_file=path
The file in which the image or the video output is to be stored.
--show[=(true|false)]
If true, displays the resulting video output in a window.
--model_dir=path

The path to the folder that contains the model files that will be used for the transformation.

--codec=fourcc

The four-character code (FOURCC) of the video codec of the output video file. The default value is `H264`.

Lossless codec, such as the Huffman Lossless Codec (FOURCC=`HFYU`), can be used to save the output video without any compression artifacts. On Windows, to use the Huffman Lossless Codec, download the `opencv_ffmpeg346_64.dll` file from https://sourceforge.net/projects/opencvlibrary/files/3.4.6/ and copy to a location on the app's search path.

--strength=(0|1)

- 0 selects a weak effect.
- 1 selects a strong effect.

--progress
Shows the progress.
--webcam
Uses the webcam as input.
--verbose [=(true|false)]
Verbose output
--debug[=(true|false)]
Prints extra debugging.
--help
Displays help information for the command.

3.3.2. Keyboard Controls

Here is the keyboard control information for the DenoiseEffect application.

The sample application provides keyboard controls for changing the run-time behavior of the application:

- **E**: Toggles the effect on and off.
- **F**: Toggles the frame rate display on and off.
- **Q** or **Escape**: exits the app and cleanly finishes writing any output file.

3.4. BatchEffectApp Application

This section provides command-line and keyboard control information for the BatchEffectApp application.

3.4.1. BatchEffectApp Command-Line Reference

Here is the command-line reference information for the BatchEffectApp application.

```
BatchEffectApp [flags ...] inFile1 [ inFileN ...]
```

Here are the flags:

--out_file=<path>
Output video files to be written, which is a pattern with a %u or %d that defaults to `"BatchOut_%02u.mp4"`.
--effect=<effect>
One of the following effects will be applied:

- Transfer
- ArtifactReduction
- SuperRes
- GreenScreen
- Upscale

--strength=<value>

The strength of an effect:

[0.0, 1.0] range for upscaling.

--scale=<scale>
The scale factor that will be applied, which are 1.3333333, 1.5, 2, 3, or 4.
--mode=<mode>
For SuperRes and ArtifactReduction, the mode values are 0 or 1:

- 0: weak effect
- 1: strong effect

--model_dir=<path>
The path to the directory that contains the models.
--verbose
Verbose output.
--help
A help message.

3.4.2. Keyboard Controls

Here is the keyboard control information for the BatchEffectApp application.

The sample application provides keyboard controls for changing the run-time behavior of the application:

- **F**: Toggles the frame rate display on and off.
- **Q** or **Escape**: exits the app and cleanly finishes writing any output file.

3.5. BatchDenoiseEffectApp Application

This section provides command-line and keyboard control information for the BatchDenoiseEffectApp application.

3.5.1. BatchDenoiseEffectApp Command-Line Reference

Here is the command-line reference information for the BatchDenoiseEffectApp application.

```
BatchDenoiseEffectApp [flags ...] inFile1 [ inFileN ...]
```

Here are the flags:

--out_file=<path>
Output video files to be written, which is a pattern with a %u or %d that defaults to `"BatchOut_%02u.mp4"`.
--strength=<value>
The values for Strength are 0 or 1.
--batchsize=<value>
The size of the batch. Default: 8

--model_dir=<path>
The path to the directory that contains the models.
--verbose
Verbose output.
--help
A help message.

3.5.2. Keyboard Controls

Here is the keyboard control information for the BatchDenoiseEffectApp application.

The sample application provides keyboard controls for changing the run-time behavior of the application:

- **F**: Toggles the frame rate display on and off.
- **Q** or **Escape**: exits the app and cleanly finishes writing any output file.

3.6. BatchAigsEffectApp Application

This section provides command-line and keyboard control information for the BatchAigsEffectApp application.

3.6.1. BatchAigsEffectApp Command-Line Reference

Here is some information about the CLI reference for the BatchAigsEffectApp.

Here is the command-line reference information for the BatchAigsEffectApp application.

```
BatchAigsEffectApp [flags ...] inFile1 [ inFileN ...]
```

Here are the flags:
--out_file=<path>
Output video files to be written, which is a pattern with a `%u` or `%d` that defaults to `"BatchOut_%02u.mp4"`.
--model_dir=<path>
The path to the directory that contains the models.
--mode=<value>
Selects the mode in which to run the application:

- `0` selects the best quality.
- `1` selects fastest performance.

--verbose
Verbose output.
--codec=<fourcc>
The fourcc code for the desired codec, for example, avc1 and h264.
--help
A help message.

3.7. VideoEffects Application

This section provides command-line and keyboard control information for the VideoEffects application.

3.7.1. VideoEffects Application Command-Line Reference

Here is the command-line reference information for the VideoEffects application.

```
VideoEffectApp [arguments...]
```

Here are the arguments:

--in_file=path
The image file or video file for the application to process.

--effect=ArtifactReduction, SuperRes, or Upscale
This argument selects the effect that will be applied:

- `ArtifactReduction`: removes the encoder artifact without changing the resolution.
- `SuperRes`: removes artifact (mode 0) and upscales to the specified output resolution.
- `Upscale`: Fast upscaler that increases the video resolution to the specified output resolution.

Note: You can also select any of the effects that are listed when you run the VideoEffectsApp with the `--help` flag.

--resolution=NNN
The desired output vertical resolution from Upscale and SuperRes, scaled 1.3333x, 1.5x, 2x, 3x or 4x times the input.

--out_file=path
The file in which the video output is to be stored.

--show[=(true|false)]
If true, displays the resulting video output in a window.

--model_dir=path
The path to the folder that contains the model files to be used for the transformation.

--codec=fourcc
The four-character code (FOURCC) of the video codec of the output video file. The default value is H264.

--strength=[0.0-1.0] for Upscale
Continuous variable values to adjust sharpness.

--mode=(0 or 1) for SuperRes or ArtifactReduction
Selects the mode of the filter to be applied:

- 0 selects a weak effect.
- 1 selects a strong effect.

--verbose [=(true|false)]
Verbose output.
--debug
Print extra debugging.
--help
Display help information for the command.

3.7.2. Keyboard Controls

Here is the keyboard control information for the VideoEffects application.

The sample application provides keyboard controls for changing the run-time behavior of the application:

- **F**: Toggles the frame rate display on and off.
- **Q** or **Escape**: exits the app and cleanly finishes writing any output file.

Chapter 4. Additional Information

This section contains additional information about using the Video Effects SDK.

4.1. Saving the Output Video in a Lossless Format

Lossless codec, such as the Ut Video Codec, can be used to save an output video from the sample applications without any compression artifacts.

For example, to save an output video with this codec, use the --codec=ULY0 option in the command-line argument of the application. On Windows, to use the Ut Video Codec, download the `opencv_ffmpeg346_64.dll` file from `https://sourceforge.net/projects/opencvlibrary/files/3.4.6/` and copy the file to a location on the application's search path.

Notice

This document is provided for information purposes only and shall not be regarded as a warranty of a certain functionality, condition, or quality of a product. NVIDIA Corporation ("NVIDIA") makes no representations or warranties, expressed or implied, as to the accuracy or completeness of the information contained in this document and assumes no responsibility for any errors contained herein. NVIDIA shall have no liability for the consequences or use of such information or for any infringement of patents or other rights of third parties that may result from its use. This document is not a commitment to develop, release, or deliver any Material (defined below), code, or functionality.

NVIDIA reserves the right to make corrections, modifications, enhancements, improvements, and any other changes to this document, at any time without notice.

Customer should obtain the latest relevant information before placing orders and should verify that such information is current and complete.

NVIDIA products are sold subject to the NVIDIA standard terms and conditions of sale supplied at the time of order acknowledgement, unless otherwise agreed in an individual sales agreement signed by authorized representatives of NVIDIA and customer ("Terms of Sale"). NVIDIA hereby expressly objects to applying any customer general terms and conditions with regards to the purchase of the NVIDIA product referenced in this document. No contractual obligations are formed either directly or indirectly by this document.

NVIDIA products are not designed, authorized, or warranted to be suitable for use in medical, military, aircraft, space, or life support equipment, nor in applications where failure or malfunction of the NVIDIA product can reasonably be expected to result in personal injury, death, or property or environmental damage. NVIDIA accepts no liability for inclusion and/or use of NVIDIA products in such equipment or applications and therefore such inclusion and/or use is at customer's own risk.

NVIDIA makes no representation or warranty that products based on this document will be suitable for any specified use. Testing of all parameters of each product is not necessarily performed by NVIDIA. It is customer's sole responsibility to evaluate and determine the applicability of any information contained in this document, ensure the product is suitable and fit for the application planned by customer, and perform the necessary testing for the application in order to avoid a default of the application or the product. Weaknesses in customer's product designs may affect the quality and reliability of the NVIDIA product and may result in additional or different conditions and/or requirements beyond those contained in this document. NVIDIA accepts no liability related to any default, damage, costs, or problem which may be based on or attributable to: (i) the use of the NVIDIA product in any manner that is contrary to this document or (ii) customer product designs.

No license, either expressed or implied, is granted under any NVIDIA patent right, copyright, or other NVIDIA intellectual property right under this document. Information published by NVIDIA regarding third-party products or services does not constitute a license from NVIDIA to use such products or services or a warranty or endorsement thereof. Use of such information may require a license from a third party under the patents or other intellectual property rights of the third party, or a license from NVIDIA under the patents or other intellectual property rights of NVIDIA.

Reproduction of information in this document is permissible only if approved in advance by NVIDIA in writing, reproduced without alteration and in full compliance with all applicable export laws and regulations, and accompanied by all associated conditions, limitations, and notices.

THIS DOCUMENT AND ALL NVIDIA DESIGN SPECIFICATIONS, REFERENCE BOARDS, FILES, DRAWINGS, DIAGNOSTICS, LISTS, AND OTHER DOCUMENTS (TOGETHER AND SEPARATELY, "MATERIALS") ARE BEING PROVIDED "AS IS." NVIDIA MAKES NO WARRANTIES, EXPRESSED, IMPLIED, STATUTORY, OR OTHERWISE WITH RESPECT TO THE MATERIALS, AND EXPRESSLY DISCLAIMS ALL IMPLIED WARRANTIES OF NONINFRINGEMENT, MERCHANTABILITY, AND FITNESS FOR A PARTICULAR PURPOSE. TO THE EXTENT NOT PROHIBITED BY LAW, IN NO EVENT WILL NVIDIA BE LIABLE FOR ANY DAMAGES, INCLUDING WITHOUT LIMITATION ANY DIRECT, INDIRECT, SPECIAL, INCIDENTAL, PUNITIVE, OR CONSEQUENTIAL DAMAGES, HOWEVER CAUSED AND REGARDLESS OF THE THEORY OF LIABILITY, ARISING OUT OF ANY USE OF THIS DOCUMENT, EVEN IF NVIDIA HAS BEEN ADVISED OF THE POSSIBILITY OF SUCH DAMAGES. Notwithstanding any damages that customer might incur for any reason whatsoever, NVIDIA's aggregate and cumulative liability towards customer for the products described herein shall be limited in accordance with the Terms of Sale for the product.

VESA DisplayPort

DisplayPort and DisplayPort Compliance Logo, DisplayPort Compliance Logo for Dual-mode Sources, and DisplayPort Compliance Logo for Active Cables are trademarks owned by the Video Electronics Standards Association in the United States and other countries.

HDMI

HDMI, the HDMI logo, and High-Definition Multimedia Interface are trademarks or registered trademarks of HDMI Licensing LLC.

ARM

ARM, AMBA and ARM Powered are registered trademarks of ARM Limited. Cortex, MPCore and Mali are trademarks of ARM Limited. All other brands or product names are the property of their respective holders. "ARM" is used to represent ARM Holdings plc; its operating company ARM Limited; and the regional subsidiaries ARM Inc.; ARM KK; ARM Korea Limited.; ARM Taiwan Limited; ARM France SAS; ARM Consulting (Shanghai) Co. Ltd.; ARM Germany GmbH; ARM Embedded Technologies Pvt. Ltd.; ARM Norway, AS and ARM Sweden AB.

OpenCL

OpenCL is a trademark of Apple Inc. used under license to the Khronos Group Inc.

NVIDIA Corporation | 2788 San Tomas Expressway, Santa Clara, CA 95051
http://www.nvidia.com

Trademarks

NVIDIA, the NVIDIA logo, and cuBLAS, CUDA, CUDA Toolkit, cuDNN, DALI, DIGITS, DGX, DGX-1, DGX-2, DGX Station, DLProf, GPU, JetPack, Jetson, Kepler, Maxwell, NCCL, Nsight Compute, Nsight Systems, NVCaffe, NVIDIA Ampere GPU architecture, NVIDIA Deep Learning SDK, NVIDIA Developer Program, NVIDIA GPU Cloud, NVLink, NVSHMEM, PerfWorks, Pascal, SDK Manager, T4, Tegra, TensorRT, TensorRT Inference Server, Tesla, TF-TRT, Triton Inference Server, Turing, and Volta are trademarks and/or registered trademarks of NVIDIA Corporation in the United States and other countries. Other company and product names may be trademarks of the respective companies with which they are associated.

Copyright

NVIDIA Corporation | 2788 San Tomas Expressway, Santa Clara, CA 95051
http://www.nvidia.com