NVIDIA OPTICAL FLOW SDK

Read Me

v | June 2021

Table of Contents

Chapter 1. Read Me

1.1. System Requirements

- NVIDIA Turing and above GPUs - Refer to the NVIDIA Optical flow developer zone web page (https://developer.nvidia.com/opticalflow-sdk) for GPUs which support Optical flow and stereo disparity hardware acceleration.
- NVIDIA Optical flow SDK. It can be downloaded from https://developer.nvidia.com/opticalflow-sdk
- Windows: Driver version 466.11 or higher
- Linux: Driver version 465.24.02 or higher

Windows Configuration Requirements

- DirectX SDK is needed. You can download the latest SDK from Microsoft's DirectX website.
- The environment variable DXSDK_DIR should be set to point to the DirectX SDK root directory, in order to build the sample applications included with the SDK
- Plus all the requirements under System Requirements and Common to all OS platforms
- Windows 10 20H1 or higher is required to used DirectX 12 interface.

Linux Configuration Requirements

- GCC 5.1 or newer is required to build and execute the sample applications.
- Building the sample applications from this SDK requires the FreeImage library to be installed. This version of the SDK has been tested against FreeImage 3.18.0. The FreeImage interface is used to read input *.png image pairs for which the optical flow needs to be calculated. It is also used to generate flow-map of the flow vectors in *.png format. End users can
  - Install the library provided by their distribution. This is the recommended approach if the version of the distribution-provided library is the same as the one used for testing this SDK, or close to to it.
  - Build and install the library from source. The source code for this library can be downloaded from http://freeimage.sourceforge.net/download.html. When compiling

FreeImage for the PowerPC architecture, users must add the line `CFLAGS += -DPNG_POWERPC_VSX_OPT=0` to the Makefile.gnu file shipped as part of FreeImage, at the end of the existing set of lines which modify CFLAGS.

- Plus all the requirements under System Requirements and Common to all OS platforms

Common to all OS platforms

- CUDA 10.2 or higher toolkit is required. It can be downloaded from http://developer.nvidia.com/cuda/cuda-toolkit
- CMake 3.14 or later. Self-extracting scripts or installers for CMake can be downloaded from https://cmake.org/download/.

1.2. Building Samples

Optical Flow SDK uses CMake for building the samples. To build the samples, follow these steps:

Windows:

1. Install all dependencies for Windows, as specified in Windows Configuration Requirements
2. Extract the contents of the SDK into a folder.
3. Create a subfolder named "build" in Optical_Flow_SDK_x.y.z/NvOFBasicSamples
4. Open a command prompt in the "build" folder and run the following command, depending upon the version of Visual Studio on your computer.
   - Visual Studio 2019: `cmake -G"Visual Studio 16 2019" -A"x64" -DCMAKE_BUILD_TYPE=Release -DCMAKE_INSTALL_PREFIX=. ..`
   - Visual Studio 2017: `cmake -G"Visual Studio 15 2017" -A"x64" -DCMAKE_BUILD_TYPE=Release -DCMAKE_INSTALL_PREFIX=. ..`
   - Visual Studio 2015: `cmake -G"Visual Studio 14 2015" -A"x64" -DCMAKE_BUILD_TYPE=Release -DCMAKE_INSTALL_PREFIX=. ..`

This command will generate the necessary Visual Studio project files in the "build" folder. You can open NvOFSamples.sln file in Visual Studio and build.

Linux:

1. Install all dependencies for Linux, as specified in Linux Configuration Requirements.
2. Extract the contents of the SDK into a folder.
3. Create a subfolder named "build" in Optical_Flow_SDK_x.y.z/NvOFBasicSamples
4. Use the following command to build samples in release mode.
   - `cmake -DCMAKE_BUILD_TYPE=Release ..`
   - `make`
   - `sudo make install`

This will build and install the binaries of the sample applications. For example, on Ubuntu, the binaries will be copied to /usr/local/bin/x64.

1.3. NvOFTracker: Build and Run Instructions

1.3.1. Prerequisites

1. CMake. Version >= 3.14
2. Visual Studio for Windows 10. Visual Studio 2019 is recommended.
3. CUDA. Version = 11.1 for Turing based GPUs. Refer Prerequisites for CUDA requirements on Nvidia Ampere GPU architecture. For linux, the recommended installation mechanism is debian installation
4. cuDNN.Version = 8.1(dev and runtime) for Turing based GPUs. Refer Prerequisites for cudnn requirements on Nvidia Ampere GPU architecture. For linux, the recommended installation mechanism is debian installation
5. TensorRT.Version = 7.2.3 for Turing based GPUs. Refer Prerequisites for TensorRT requirements on Nvidia Ampere GPU architecture. For linux, the recommended installation mechanism is debian installation. Trtexec is generally found at /usr/source/tensorrt/bin for linux
6. Video Codec SDK. Version >= 10.0
7. Git.
8. OpenCV. Refer OpenCV sub section in the Build sections

1.3.2. Windows 10 Build

Assume NvOFTracker is present here: `C:/Users/TestPC/Downloads/OpticalFlowSDK/NvOFTracker`. All paths below are relative to this path (unless specified otherwise)

CUDA, cuDNN, TensorRT(TRT)

Use the individual installation instruction for each of these libraries.

1. For cuDNN, copy each of the bin, lib and include folder contents to the corresponding folders in the cuda tool kit. This will let applications automatically search for cudnn header, libs and binaries as cuda toolkit is already in path
2. For TRT you can could do the same as above. If you choose not to, then add the lib folder (contains dlls) to path so that applications can find them at runtime.

Video Codec SDK

On downloading Video Codec SDK, if VideoCodecSDK represents the root, then add `VideoCodecSDK/Samples/External/FFmpeg/lib/x64` to path so the necessary ffmpeg dlls are found by the application at run time.

OpenCV

Use the install script(`Scripts/installOCV_Windows.sh`) to install opencv. Note that you will need Git installed and you will need to run the installation script in Git bash. When all is done, there should be an install folder in the current directory. Go to `Scripts/Install/opencv/x64/vc14/bin` and copy the entire path and add it to your system Path variable. This will help applications find the opencv related dlls at run time.

NvOFTSample and NvOFTracker:

Steps to build:

1. Do, `cd C:/Users/TestPC/Downloads/OpticalFlowSDK/NvOFTracker && mkdir build && cd build`
2. Run, `cmake -DOpenCV_DIR=opencvDir -DTRT_ROOT=trtRoot -DVIDEOCODEC_SDK_ROOT=videocodecsdkroot ..`
   - replace **opencvDir** with the directory containing OpenCVConfig.cmake (generally under `Scripts/Install/opencv` folder)
   - replace **trtRoot** with the location of TensorRT root in your downloads (For eg. `C:/Users/TestPC/Downloads/TensorRT-7.2.3.4.Windows10.x86_64.cuda-11.1.cudnn8.1/TensorRT-7.2.3.4`)
   - replace **videocodecsdk** with the location of Video Codec SDK root (necessarily the folder containing samples folder, `VideoCodecSDK/Samples`)
3. In the current directory there will be VS solution file with name `NvOFTrackerMain.sln`. Open it and build the INSTALL project.
4. The above will create a folder called bin. This folder will contain `nvoftracker.dll` library and `NvOFTSample` executable.

1.3.3. Linux Build

Assume NvOFTRacker is present here: `/home/Downloads/OpticalFlowSDK/NvOFTracker`. All paths below are relative to this path (unless specified otherwise)

CUDA, cuDNN, TensorRT

Use the individual installation instruction for each of these libraries. Use debian installation so that all paths are configured.

Video Codec SDK

Unlike windows, the ffmpeg libraries need to be built for linux. You can find the source of ffmpeg shipped as part of ffmpeg. If VideoCodecSDK is the root then `VideoCodecSDK/Samples/External/FFmpeg/src` will contain the zipped src folder. Steps to build:

1. Unzip the source folder. cd into the folder.
2. `./configure --enable-shared`
3. `make -j 8`
4. `sudo make install`

This will install the ffmpeg libraries which then can be used by app.

OpenCV

Use the install script(`Scripts/installOCV_Linux.sh`) to install opencv. Make sure ffmpeg is built before running this script. Please run `dos2unix installOCV_Linux.sh` in case there are line ending related issues.

NvOFTSample and NvOFTracker:

Steps to build:

1. Do, `cd /home/Downloads/OpticalFlowSDK/NvOFTracker && mkdir build && cd build`
2. Run, `cmake -DOpenCV_DIR=opencvDir -DVIDEOCODEC_SDK_ROOT=videocodecsdkroot ..`
   - replace **opencvDir** with the directory containing OpenCVConfig.cmake (generally under `Build/opencv` folder)
   - replace **videocodecsdk** with the location of Video Codec SDK root (necessarily the folder containing samples folder, `VideoCodecSDK/Samples`)
3. In case you followed tar installation for TensorRT then Run, `cmake -DOpenCV_DIR=opencvDir -DVIDEOCODEC_SDK_ROOT=videocodecsdkroot -DTRT_ROOT=trtRoot ..`
   - replace **opencvDir** with the directory containing OpenCVConfig.cmake (generally under `Build/opencv` folder)
   - replace **videocodecsdk** with the location of Video Codec SDK root (necessarily the folder containing samples folder, `VideoCodecSDK/Samples`)
   - replace **trtRoot** with the location of TensorRT root
4. Run `make install`
5. The above will create a folder called bin. This folder will container libnvoftracker.so library and NvOFTSample executable.

1.3.4. Running applications

Building TensorRT Detector Engine:

You will need to build Tensorrt engine(.trt file) for the detector to be used in NvOFTSample. There is an onnx model of YOLOv3 detector at the below location:

`NvOFTSample/detector/models/yolov3.onnx`

you will need to use the above onnx to generate TRT engine. Note that onnx file is platform and GPU agnostic. But that is not the case of trt engine. TRT engine is specific to Operating System and GPU being used. **Steps**:

1. Navigate to directory containing trtexec.
   - For Windows, go to your TRT download location. Navigate to bin folder which contains trtexec.
   - For Linux, suggested method is to do `sudo find / -name trtexec`. This will spew the location. Generally it is under `/usr/src/tensorrt/bin`.
2. Use the following command to create the engine file.
   - Windows `trtexec --onnx=C:/Users/TestPC/Downloads/OpticalFlowSDK/NvOFTracker/NvOFTSample/detector/models/yolov3.onnx --saveEngine="yolov3.trt"`
   - Linux `trtexec --onnx=/home/Downloads/OpticalFlowSDK/NvOFTracker/NvOFTSample/detector/models/yolov3.onnx --saveEngine="yolov3.trt"` Note that --onnx `yolov3.trt` will be created in the current directory. You can choose to provide some other location as well.

NvOFTSample

Run NvOFTSample to see the help menu. Use the engine generated above to run the samples. NvOFTSample only supports avi format for the `-o` parameter.

```
Mandatory Parameters
    -i            Input video file
    -e            TensorRT Engine file for Detector
    Optional Parameters:
    -o            Output video file
    -fT           Filename to dump tracked objects
    -dC           Dump tracked objects to Console
    -sI           Detection skip interval. Must be 0 or greater
    -g            GPU Id on which the tracker needs to run. Default is 0`
```

1.3.5. A Note on NvOFTracker and NVIDIA Ampere GPU architecture

There is a known issue where using TensorRT 7.x to build/run `yolov3.trt`(using `yolov3.onnx`) may lead to system crashes. This issue is specific to Nvidia Ampere GPU

architecture and is being actively worked upon. NvOFTracker itself works fine on Nvidia Ampere GPU architecture.

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 acknowledgment, 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.

Trademarks

NVIDIA, the NVIDIA logo, and cuBLAS, CUDA, CUDA Toolkit, cuDNN, DALI, DIGITS, DGX, DGX-1, DGX-2, DGX Station, DLProf, GPU, Jetson, Kepler, Maxwell, NCCL, Nsight Compute, Nsight Systems, NVCaffe, NVIDIA Deep Learning SDK, NVIDIA Developer Program, NVIDIA GPU Cloud, NVLink, NVSHMEM, PerfWorks, Pascal, SDK Manager, 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