DeepSight SDK Documentation

Introduction

DeepSight SDK is a cross platform software library for automated semantic analysis of people in video and images. DeepSight SDK can be used to automatically analyze faces in real time via a simple webcam, and communicate the resulting information to the third party application.

Dependencies

Windows

- Visual Studio C++11 compiler
- OpenCL library (optional, recommended). See OpenCL acceleration OpenCL section below

Linux

- Ubuntu 18.04 and up
- C++11 compiler and build tools: sudo apt install build-essential
- Lib Poco 1.8.0 and up : sudo apt install libpoco-dev
- FFmpeg libraries: sudo apt install libavcodec-dev libavformat-dev libswscale-dev libmodplug-dev
- OpenCL library (optional, recommended). See OpenCL acceleration OpenCL section below

OpenCL acceleration

The DeepSight SDK comes with OpenCL support which can significantly increase its runtime performance while reducing CPU utilization depending on the hardware available. This was evaluated on Intel Iris+ integrated graphics hardware, for which client drivers can be installed by following the instructions for your specific operating system below. Once present on your system, the SDK will automatically detect these and make use of OpenCL acceleration when performing network inference. OpenCL drivers for other platforms (e.g. AMD) may also be compatible, but have not yet been tested. Note also that utilizing the GPU does not necessarily result in a speedup and may cause program instability depending on the specific hardware and driver combination.

- **Linux:** Follow the installation instructions at https://github.com/intel/compute-runtime/releases/latest under the "Installation procedure" section.
- **Windows:** Generally, Windows has already drivers that enable OpenCL support for the Intel integrated graphics hardware, but more performance can be obtained if the specific Intel drivers are installed.
  We have verified OpenCL functionality on NUC7i*BN and NUC8i7BE systems, for which suitable drivers can be downloaded
  here: https://downloadcenter.intel.com/download/28974/
  For the majority of other Intel processors, the Windows DCH drivers can be downloaded here: https://downloadcenter.intel.com/download/29058/Intel-Graphics-Windows-10-DCH-Drivers?product=80939

OpenCL acceleration can be enabled/disabled dynamically at runtime.

Minimum requirements

- **Windows 10** or **Ubuntu 18.04**
- Intel i5 or i7 4th generation
- 4 GB of RAM
- 500 MB of free disk space
- Input frames with a resolution of 640x480

Getting Started

After the dependencies are met, navigate to the samples directory and compile the example projects:

- **Linux** : cd samples; make
- **Windows** : Open the samples.sln sample project in the samples directory with Visual Studio and compile the examples

The example applications are a good starting point to understand the capabilities of the SDK, its usage and how to start a project with face analysis features.

DeepSight SDK runtime workflow

The typical runtime workflow when using the DeepSight SDK is:

- Create a ds::Settings instance with proper license key and neural networks path.
- Create a ds::DeepSight instance with the ds::Settings instance.
- Authenticate the SDK ds::DeepSight::authenticate().
- Analize frames with ds::DeepSight::analyze().
- Process the ds::Person instances detected in each frame.

License

The DeepSight SDK needs to be initialized with a license key in order to work. If you don't have one yet, contact sales@sightcorp.com. To allow the SDK to analyze images (eg. ds::DeepSight::analyze) a DeepSight instance needs to be authenticated first through the ds::DeepSight::authenticate() call.

Sample usage}

Tracker

The SDK is able to track people along time, interpolating information between frame sequences. In order to make use of the SDK's tracking capabilities, the ds::DeepSight::analyze() function should be used instead of ds::DeepSight::analyzeSingleImage(). Tracking enables people counting features, smoothes age and gender estimations, allows to compute ds::Person::attentionDuration, and so on.

People Counting

When using ds::DeepSight::analyze() to process a video feed, you can get the people count from the SDK by calling the function ds::DeepSight::getPeopleCount(). The counter keeps track of the number of unique IDs that were assigned to people by creating a session whenever a person walks into the field of view (FOV) of the camera. During this session, the person will be tracked within the FOV until the person leaves the FOV again. If the subject re-enters the FOV, a new session will be created and this person will be double counted.

To control the scenario in which you want to apply counting, you might need to specify a region of interest (ROI) by using a subset of the input frame to make sure you're not overcounting people that are not within your scenario (eg. counting at an entrance door). This can be done easily using OpenCV functions, for example:

```
cv::VideoCapture cap;
```

```
cap.open( 0 ); // open the default camera, lets assume a 640x480 resolution input
cv::Mat image;
cap >> image; // stream the camera input to a cv::Mat
cv::Mat roi_image = image(cv::Rect( 20, 20, 100, 100 ) ); // get a ROI from the original image
std::vector<ds::Person> people = ds.analyze( roi_image ); // analyze the ROI
```

Other parameters that can be used to control the scenario are the settings Minimum Face Size and Maximum Face Size. You can set these either using the ds::settings object or in the settings file itself if you store this to disk. The default values set for minimum and maximum face size are 0 and allow for detection of faces in all sizes. By increasing the minimum face size, you can filter the smaller faces and they will not be taken into account for counting. Similarly, by increasing the maximum face size, you can filter on the larger faces not to be counted. The minimum face size should always be smaller than the maximum face size.

You should use ds::DeepSight::analyze() when processing a video stream. If you are interested in single image analysis, you can use ds::DeepSight::analyzeSingleImage() instead. When processing single images, there is no people counting performed.

Quality of estimates

The SDK has been thoroughly tested against several datasets to maintain overall accuracy. However, in real world scenarios a lot of different factors can influence results, such as backlight, shadows or poorly positioned cameras which can lower the quality of the estimates. Please make sure to take the following into account when configuring your scenario:

Camera Settings

Depending on your camera manufacturer, it should be possible to configure camera parameters for the best results. The most important part of the configuration is to enable anti-flicker for your region (50 or 60 Hz) and to disable automatic gain, white balance and focus control. This is a fundamental step, as the software will recognize any changes in the images as a change in the face, significantly affecting the classification and tracking results.

Positioning

An ideal positioning of the camera would be aimed at a narrow space where people pass one by one, such as a doorway. You can zoom in on the upper part of the door and capture each person's face one by one while they are entering or exiting. This allows for a high face image resolution as opposed to a scenario where people can pass by alongside each other and thus a lesser level of zoom must be applied to process multiple people at once.

Angle

The best age and gender results are obtained when people are captured frontally. The bigger the angle of the camera with respect to your ROI is, the bigger the error margins are on the age and gender estimates. For example, when using a ceiling mounted camera looking at a door, make sure that you try to minimize the angle of the camera looking down at the door by moving it backwards and apply zoom on your ROI.

Distance

The further the person is from the camera, the smaller the resolution of the face in the image will be and the harder it will be to detect the person. To detect people from a distance, two things are required. Firstly, the capture resolution of the camera must be sufficiently large, 720p or 1080p is advised. Secondly, optical or digital zoom can be used in order to detect faces with sufficient resolution.

Lighting

The faces must be sufficiently and frontally lit. In case of nonlinear light patterns on the face or strong sideways illumination, the SDK will have more difficulty to correctly process the features of a person. Back illumination, such as a window behind a person, darkens the face feature and is thus less desirable. This can be counteracted to some extent by disabling automatic camera adjustment such as white-balance, focus and exposure controls.

Class List

Here are the classes, structs, unions and interfaces with brief descriptions:

[detail level 123]

▼ **Nds**

▼ **Nhelp**

CTimer	Provides utilities to measure execution time of various SDK functions

C DeepSight	DeepSight SDK engine class
▼C Person	DeepSight SDK person class
C Headpose	Headpose structure for holding head rotation information
C Settings	DeepSight SDK settings class

Class Index

Class Members