

## Examples Guide

## Contents

|   |           |
|---|-----------|
| <b>LUNA SDK Examples</b>  | <b>5</b>  |
| Building Examples . . . . .   | 5         |
| Running Examples . . . . .  | 5         |
| Building Examples for Windows . . . . .                               | 6         |
| Running on Windows: . . . . .   | 6         |
| Building Examples for MacOS . . . . .                                 | 7         |
| Depth example . . . . .   | 7         |
| Depth example requirements . . . . .                                  | 8         |
| Liveness Examples . . . . .   | 8         |
| Simple Liveness (example_liveness) . . . . .                          | 9         |
| Simple Liveness Requirements (simple-liveness-requirements) . . . . . | 9         |
| Run Simple Liveness . . . . .   | 9         |
| <b>Best shot example with TrackEngine</b>                             | <b>9</b>  |
| What it does . . . . .  | 9         |
| Prerequisites . . . . .   | 9         |
| Example walkthrough . . . . .   | 10        |
| How to run . . . . .  | 10        |
| Building for linux . . . . .  | 11        |
| Building for windows . . . . .  | 11        |
| Running on linux . . . . .  | 11        |
| Running on windows . . . . .  | 11        |
| <b>Depth liveness example</b>   | <b>12</b> |
| What it does . . . . .  | 12        |
| Expected output . . . . .   | 12        |
| Prerequisites . . . . .   | 12        |
| Example walkthrough . . . . .   | 12        |
| How to run . . . . .  | 12        |
| <b>Descriptor IO example</b>  | <b>13</b> |
| What it does . . . . .  | 13        |
| Prerequisites . . . . .   | 13        |
| Example walkthrough . . . . .   | 13        |
| How to run . . . . .  | 13        |
| Example output . . . . .  | 13        |
| <b>Detection example</b>  | <b>13</b> |
| What it does . . . . .  | 13        |

|                                   |           |
|-----------------------------------|-----------|
| Prerequisites . . . . .           | 13        |
| Example walkthrough . . . . .     | 13        |
| How to run . . . . .              | 14        |
| Example output . . . . .          | 14        |
| <b>Estimation example</b>         | <b>20</b> |
| What it does . . . . .            | 20        |
| Prerequisites . . . . .           | 20        |
| Example walkthrough . . . . .     | 20        |
| How to run . . . . .              | 21        |
| Example output . . . . .          | 21        |
| <b>Extraction example</b>         | <b>22</b> |
| What it does . . . . .            | 22        |
| Prerequisites . . . . .           | 22        |
| Example walkthrough . . . . .     | 22        |
| Preparations . . . . .            | 22        |
| Face detection . . . . .          | 22        |
| Descriptor extraction . . . . .   | 23        |
| Descriptor matching . . . . .     | 24        |
| Putting it all together . . . . . | 24        |
| <b>Human Detection example</b>    | <b>25</b> |
| What it does . . . . .            | 25        |
| Prerequisites . . . . .           | 25        |
| Example walkthrough . . . . .     | 25        |
| How to run . . . . .              | 25        |
| Example output . . . . .          | 25        |
| <b>Reidentification example</b>   | <b>26</b> |
| What it does . . . . .            | 26        |
| Prerequisites . . . . .           | 26        |
| Example walkthrough . . . . .     | 26        |
| Preparations . . . . .            | 27        |
| Human detection . . . . .         | 27        |
| Extraction . . . . .              | 27        |
| Descriptor matching . . . . .     | 28        |
| How to run . . . . .              | 29        |
| Example output . . . . .          | 29        |

|  |               |
|--|---------------|
| <b>Index example</b>                             | <b>29</b>     |
| What it does . . . . .                           | 29            |
| Prerequisites . . . . .                          | 29            |
| Example walkthrough . . . . .                    | 30            |
| How to run . . . . .                             | 30            |
| Example output . . . . .                         | 30            |
| <br><b>Infra Red Liveness Estimation example</b> | <br><b>30</b> |
| What it does . . . . .                           | 30            |
| Prerequisites . . . . .                          | 30            |
| How to run . . . . .                             | 31            |
| Example output . . . . .                         | 31            |
| <br><b>Liveness example</b>                      | <br><b>31</b> |
| What it does . . . . .                           | 31            |
| Prerequisites . . . . .                          | 32            |
| Example walkthrough . . . . .                    | 32            |
| Stage 0. Preparations . . . . .                  | 32            |
| Stage 1. Main cycle . . . . .                    | 32            |
| How to run . . . . .                             | 33            |
| <br><b>PPE estimator aggregation example</b>     | <br><b>33</b> |
| What it does . . . . .                           | 33            |
| Prerequisites . . . . .                          | 33            |
| Example walkthrough . . . . .                    | 33            |
| How to run . . . . .                             | 33            |
| Building for linux . . . . .                     | 34            |
| Building for windows . . . . .                   | 34            |
| Visalising output . . . . .                      | 35            |

## LUNA SDK Examples

Welcome to Example guide, which describes sample code for VisionLabs LUNA SDK. The presented examples are compatible with SDK version 5.0.0 and newer.

**Please note that while these examples are released under MIT license, the SDK itself is not. Contact us via email ([info@visionlabs.ru](mailto:info@visionlabs.ru)) for evaluation and/or licensing terms and conditions.**

Aside from examples itself, there are some supplementary materials you may find useful. Look into *cmake/* folder for a CMake find script for the SDK. CMake usage is not mandatory but we advise you to do it.

Currently we support 64 bit Windows and Linux. On Windows, everything should work with Visual Studio 2015. On Linux, we tested this code with GCC 4.8.5. The other versions may work as well. Note, that the SDK is officially supported on RedHat Linux families (RHEL, CentOS, Fedora).

## Building Examples

The build sequence for basic examples (example\_extraction, example\_estimation, example\_descriptor\_io, example\_index) is presented below. The main dependency for basic examples is LUNA SDK, these examples should work out of the box. From Luna SDK root.

```
$ mkdir build && cd build
$ cmake -DCMAKE_BUILD_TYPE=Release -DFSDK_ROOT=.. ../examples
$ make
```

Optionally you can enable complementary examples, which require additionally installed 3rd party libraries:

- Liveness examples (example\_liveness, example\_depth) - shows interface compatibility of LUNA SDK (LivenessEngine) and OpenCV library.

Please refer to requirements sections of corresponding examples.

## Running Examples

Data folder is required at <LUNA\_SDK\_root>/data on Windows and on Linux.

```
# Detecting, Extracting, Matching
$ build/example_extraction/example_extraction examples/images/Cameron_Diaz.
  ppm examples/images/Cameron_Diaz_2.ppm 0.7
$ build/example_extraction/example_extraction examples/images/Cameron_Diaz.
  ppm examples/images/Jennifer_Aniston.ppm 0.7
```

```
# Detecting, Landmarks, Estimating (Attributes, Quality, Eyes, Head pose)
$ build/example_estimation/example_estimation examples/images/portrait.ppm

# Indexing
$ build/example_index_hnsw/example_index_hnsw examples/images/Cameron_Diaz.ppm examples/images/ examples/images_lists/list.txt 0.7

# Descriptor and batch saving/loading
$ build/example_descriptor_io/example_descriptor_io examples/images/portrait.ppm
```

## Building Examples for Windows

Building command (from FSDK\_ROOT):

```
$ mkdir build install && cd build
$ cmake -G "Visual Studio 15 2017 Win64" -DCMAKE_BUILD_TYPE=Release -
  DFSDK_ROOT=.. -DCMAKE_INSTALL_PREFIX=..\install ../examples
$ cmake --build . --config Release --target install
```

After this steps You can find examples is in '`FSDK_ROOT\install\bin\`' directory.

## Running on Windows:

Manually register the *necessary dynamic libraries* (.dll) in the system (if needed).Or copy all necessary dynamic libraries (\*.dll) in the directory with an example.

Run the command from "FSDK\_ROOT".

```
$ install\bin\example_estimation.exe examples/images/Jennifer_Aniston.jpg
```

*LivenessEngineSDK.dll*, *FaceEngineSDK.dll*, *tbb.dll*, *flower.dll*, *\*TrackEngineSDK.dll*

## Running on Windows (from FSDK\_ROOT):

Note: Manually register the *necessary dynamic libraries* (.dll) in the system (if needed).Or copy all necessary dynamic libraries (\*.dll) in the directory with an example.

```
$ install\bin\example_estimation.exe examples/images/Jennifer_Aniston.jpg
```

*LivenessEngineSDK.dll*, *FaceEngineSDK.dll*, *tbb.dll*, *flower.dll*, *\*TrackEngineSDK.dll*

## Building Examples for MacOS

Building command (from FSDK\_ROOT):

```
$ mkdir build && cd build
$ cmake -DCMAKE_BUILD_TYPE=Release -DFSDK_ROOT=.. ../examples
$ make
```

If you have some problems with execution you will possibly need to re-sign all binary files. For example you can do it by such a way:

```
codesign -v --force --sign - --entitlements test.xcent lib/clang/x64/*
codesign -v --force --sign - --entitlements test.xcent bin/clang/x64/*
```

Possible content of test.xcent.

```
<?xml version="1.0" encoding="UTF-8"?>

<!DOCTYPE plist PUBLIC "-//Apple//DTD PLIST 1.0//EN" "http://www.apple.com/
  DTDs/PropertyList-1.0.dtd">

<plist version="1.0">

  <dict>

    <key>com.apple.security.get-task-allow</key>

    <true/>

  </dict>

</plist>
```

Also you possibly should to allow execution for binary file on your device: System Preference - Security & Privacy - General - Allow apps downloaded from anywhere.

## Depth example

Depth example performs liveness check based on depth frames captured from Intel Realsense Depth cameras.

## Depth example requirements

### Software requirements:

The following 3rd-party dependencies are required in order to build and run depth example: \* OpenCV

```
* core
* imgproc
* highgui
```

- Intel Realsense SDK

### Hardware requirements:

- Intel RealSense depth cameras either D400 series or the SR300 and compatible device drivers.

To verify that all realsense dependent libraries are installed correctly on the system, we recommend building librealsense SDK from source.

For build instructions please visit the following links:

- Linux: <https://github.com/IntelRealSense/librealsense/blob/master/doc/installation.md>
- Windows: [https://github.com/IntelRealSense/librealsense/blob/master/doc/installation\\_windows.md](https://github.com/IntelRealSense/librealsense/blob/master/doc/installation_windows.md)

For more information about RealSense SDK see <https://github.com/IntelRealSense/librealsense>.

Note: In order to run depth example on Windows make sure you have copied all example's dependent dlls into directory where example\_depth binary is located, or just put them into OS system's default path.

### Build command (from FSDK\_ROOT/examples):

```
$cmake -BBuild -H. -DCMAKE_INSTALL_PREFIX=./
-DOpenCV_DIR=<absolute_path_to_opencv_install_directory>
-Drealsense2_DIR=<absolute_path_to_realsense_sdk_install_directory>/lib/
cmake/realsense2
-DFSDK_ROOT=<absolute_path_to_luna_sdk_root_directory>
-DWITH_DEPTH_EXAMPLE=ON
$ cmake --build Build --config Release --target install
```

### Run depth example(from FSDK\_ROOT/examples):

```
$ bin/example_depth <absolute_path_to_luna_sdk_data_directory>
```

## Liveness Examples

For these examples DLSDK\_ROOT path should be specified. By default LivenessEngine libraries are located near the FaceEngine libraries and DLSDK\_ROOT same as DFSDK\_ROOT.



## Simple Liveness (example\_liveness)

### Simple Liveness Requirements (simple-liveness-requirements)

To build this example OpenCV library (modules listed below) and cmake `DWITH_LIVENESS_EXAMPLE` option are required to be installed. By default, `DWITH_LIVENESS_EXAMPLE` option is disabled.

OpenCV should contain the next modules:

- core
- imgproc
- highgui
- videoio

### Run Simple Liveness

#### Building command (from `FSDK_ROOT/build`):

```
$ cmake ../examples -DCMAKE_BUILD_TYPE=Release -DWITH_LIVENESS_EXAMPLE=ON -  
    DOpenCV_DIR=<path_to_opencv> -DFSDK_ROOT=.. -DLSDK_ROOT=..  
$ make
```

#### Run Simple liveness example:

```
$ build/example_liveness/example_liveness <web_cam_id> <test_number>
```

## Best shot example with TrackEngine

### What it does

This example demonstrates how to use the TrackEngine to track human face detection and to check him with eye estimation status. This example shows how to pick frame with the best quality face out of video sequence. You can find binary of this example in `FSDK_ROOT/examples/bin/possible_output_prefix`. BestShot example can display the following results:

- accepted - user blinked
- denied - user did not blink

### Prerequisites

This example assumes that you have already read the **FaceEngine Handbook** and **LivenessEngine Handbook** (or at least have them somewhere nearby for reference) and know some core concepts,

like memory management, object ownership, and life-time control. This sample will not explain these aspects in detail.

## Requirements

OpenCV library installation containing next modules:

- highgui
- imgproc
- videoio

## Example walkthrough

To get familiar with FSDK and TSDK usage and common practices, please go through example\_estimation first.

## How to run

It is required to pass the number of used cameras or a path to the video file. Built executable file placed in

```
luna-sdk_linux_*/examples/bin/*/example_bestshot_trackengine
```

works with only cameras. To work with video please build the source code of this example. Besides you have to use OpenCV with ffmpeg and GUI (for example GTK+-3.0). This example was tested with OpenCV 3.4.12.

Using statically linked ffmpeg may be against the license agreement. See <https://ffmpeg.org/legal.html> point 2.

```
./example_bestshot_trackengine <path to the video>
```

or you can pass the number of video cameras:

```
./example_bestshot_trackengine 1
```

Note: you can disable the GUI by passing an optional parameter `--no_gui`:

```
./example_bestshot_trackengine <path to the video> --no_gui
```

## Building for linux

### Building command (from FSDK\_ROOT/examples/build):

```
$ cmake -DFSDK_ROOT=<absolute_path_to_fsdk_root_dir> -DTSDK_ROOT=<
  absolute_path_to_fsdk_root_dir> -DWITH_BESTSHOT_EXAMPLE=ON -DOpenCV_DIR=<
  abs_path_to_opencv> -DCMAKE_BUILD_TYPE=Release -DCMAKE_INSTALL_PREFIX=<
  absolute_path_to_fsdk_root_dir>/examples -DCMAKE_MODULE_PATH=<
  absolute_path_to_fsdk_root_dir>/examples/cmake ../
  example_bestshot_trackengine
$ cmake --build . --config Release --target install
```

## Building for windows

### Building command (from FSDK\_ROOT/examples/build):

```
$ cmake -G "Visual Studio 14 2015 Win64" -DFSDK_ROOT=<
  absolute_path_to_fsdk_root_dir> -DTSDK_ROOT=<
  absolute_path_to_fsdk_root_dir> -DWITH_BESTSHOT_EXAMPLE=ON -DOpenCV_DIR=<
  abs_path_to_opencv> -DCMAKE_BUILD_TYPE=Release -DCMAKE_INSTALL_PREFIX=<
  absolute_path_to_fsdk_root_dir>/examples -DCMAKE_MODULE_PATH=<
  absolute_path_to_fsdk_root_dir>/examples/cmake ../
  example_bestshot_trackengine
$ cmake --build . --config Release --target install
```

## Running on linux

There is prebuilt demo, see FSDK\_ROOT/examples/bin/*possible\_output\_prefix*/ . To run it change your working directory to FSDK\_ROOT and type:

```
$ ./examples/bin/*possible_output_prefix*/./example_bestshot_trackengine
```

## Running on windows

For windows you will have to run bat script that is in the same directory as ./example\_bestshot\_trackengine.exe. It's win32\_copy\_script.bat for win32 and win64\_copy\_script.bat for win64. So for windows, to run demo you will have to type:

```
$ .\examples\bin\*possible_output_prefix*\win32_copy_script.bat
$ .\examples\bin\*possible_output_prefix*\example_bestshot_trackengine.exe
```

or:

```
$ .\examples\bin\*possible_output_prefix*\win64_copy_script.bat  
$ .\examples\bin\*possible_output_prefix*\example_bestshot_trackengine.exe
```

## Depth liveness example

### What it does

This example demonstrates depth liveness estimation using Intel RealSense cameras. It predicts whether a person on image is real or fake. Example captures frames from colored(rgb8) and depth(r16) sensors, detects face on input image and performs liveness estimation.

### Expected output

if liveness depth estimation predicted that person on image is alive bounding box is colored green, red otherwise.

### Prerequisites

This example assumes that you have already read the **FaceEngine Handbook** (or at least have it somewhere nearby for reference) and know some core concepts, like memory management, object ownership, and life-time control. This sample will not explain these aspects in detail.

### Example walkthrough

Example source code is heavily commented, you should have no problems just following through the code base to understand what is going on.

### How to run

Use the following command to run the example.

```
./example_depth <absolute_path_to_luna_sdk_data_directory>
```

## Descriptor IO example

### What it does

The example demonstrates how to save a descriptor and descriptor batch to a file.

### Prerequisites

This example assumes that you have already read the **FaceEngine Handbook** (or at least have it somewhere nearby for reference) and know some core concepts, like memory management, object ownership, and life-time control. This sample will not explain these aspects in detail.

### Example walkthrough

To get familiar with FSDK usage and common practices, please go through `example_extraction` first.

### How to run

Use the following command to run the example.

```
./example_descriptor_io <some_image.ppm>
```

### Example output

Warped images, descriptors, descriptor batch.

## Detection example

### What it does

This example demonstrates how to use the detector to get a face bbox (bounding box) and landmarks.

### Prerequisites

This example assumes that you have already read the **FaceEngine Handbook** (or at least have it somewhere nearby for reference) and know some core concepts, like memory management, object ownership, and life-time control. This sample will not explain these aspects in detail.

### Example walkthrough

To get familiar with FSDK usage and common practices, please go through `example_extraction` first.

## How to run

An example of use with one image. Batch size is an optional parameter. By default will be 1.

```
./example_detection --image <some_image.ppm>
```

An example of use with batch filled by the same image.

```
./example_detection --image <some_image.ppm> --batchSize 4
```

An example of use with the batch of images.

```
./example_detection --imageList <someImageList.txt> --batchSize 3
```

An example of use with the batch of images with default batchSize.

```
./example_detection --imageList <someImageList.txt>
```

Running the example without parameters is incorrect.

Example of a list of images:

```
/home/user/images/image_1.jpg  
/home/user/images/image_2.jpg  
/home/user/images/image_3.jpg  
/home/user/images/image_4.jpg  
/home/user/images/image_5.jpg  
/home/user/images/image_6.jpg
```

Image paths must be absolute or relative to the working directory.

## Example output

```
simpleDetect. bbox only result:
```

```
Rect:
```

```
  x = 296.598  
  y = 77.0892  
  w = 145.655  
  h = 206.526  
  score = 0.999943
```

```
simpleDetect. bbox and Landmarks5 result:
```

Rect:

x = 296  
y = 77  
w = 145  
h = 206  
score = 0.999943

Landmarks5:

x = 338.899 y = 157.901  
x = 407.017 y = 161.499  
x = 374.56 y = 199.138  
x = 341.442 y = 238.993  
x = 396.452 y = 242.443

simpleDetect. bbox, Landmarks5 and Landmarks68 result:

Rect:

x = 296  
y = 77  
w = 145  
h = 206  
score = 0.999943

Landmarks5:

x = 341 y = 157  
x = 408 y = 160  
x = 377.1 y = 216.248  
x = 342.972 y = 237.357  
x = 396.948 y = 238.631

Landmarks68:

x = 296.226 y = 154.527  
x = 296.786 y = 176.751  
x = 298.726 y = 198.489  
x = 301.404 y = 219.905  
x = 308.43 y = 241.047  
x = 319.536 y = 259.856  
x = 334.036 y = 275.579  
x = 351.577 y = 287.055  
x = 371.124 y = 289.996  
x = 388.04 y = 284.708  
x = 400.513 y = 271.055  
x = 411.994 y = 255.058  
x = 421.157 y = 237.496  
x = 427.666 y = 219.142  
x = 433.504 y = 200.254  
x = 437.435 y = 181.18  
x = 439.423 y = 161.023  
x = 306.067 y = 147.31

x = 318.217 y = 133.185  
x = 336.968 y = 128.427  
x = 356.102 y = 130.611  
x = 372.956 y = 137.662  
x = 386.046 y = 139.238  
x = 400.82 y = 133.647  
x = 416.646 y = 133.024  
x = 430.68 y = 138.276  
x = 436.744 y = 150.804  
x = 378.511 y = 156.42  
x = 379.145 y = 171.631  
x = 379.889 y = 186.692  
x = 380.242 y = 202.458  
x = 360.399 y = 210.427  
x = 368.582 y = 213.492  
x = 377.1 y = 216.248  
x = 384.595 y = 214.076  
x = 391.243 y = 211.185  
x = 327.115 y = 156.714  
x = 336.181 y = 152.848  
x = 345.985 y = 153.112  
x = 354.717 y = 159.279  
x = 345.537 y = 160.245  
x = 335.503 y = 160.111  
x = 395.182 y = 161.174  
x = 404.413 y = 155.724  
x = 413.722 y = 156.424  
x = 420.032 y = 160.665  
x = 413.331 y = 163.648  
x = 403.925 y = 163.023  
x = 342.972 y = 237.357  
x = 356.428 y = 237.022  
x = 367.702 y = 236.126  
x = 374.406 y = 238.745  
x = 381.428 y = 236.774  
x = 389.371 y = 238.498  
x = 396.948 y = 238.631  
x = 388.453 y = 246.945  
x = 379.715 y = 250.657  
x = 372.491 y = 251.414  
x = 364.784 y = 250.295  
x = 354.543 y = 246.516  
x = 347.606 y = 238.363  
x = 366.75 y = 239.997  
x = 373.865 y = 241.541



```
x = 380.56 y = 240.759
x = 392.985 y = 239.555
x = 379.898 y = 242.199
x = 372.858 y = 243.308
x = 365.794 y = 241.718
```

simpleRedetect. detect result:

Rect:

```
x = 296.598
y = 77.0892
w = 145.655
h = 206.526
score = 0.999943
```

simpleRedetect. redetect result:

Rect:

```
x = 293
y = 96
w = 150
h = 192
score = 0.99776
```

Landmarks5:

```
x = 340 y = 158
x = 407 y = 162
x = 373.659 y = 216.406
x = 341.528 y = 234.974
x = 395.089 y = 237.747
```

Landmarks68:

```
x = 298.137 y = 150.128
x = 298.055 y = 171.151
x = 299.717 y = 192.454
x = 302.577 y = 214.068
x = 308.745 y = 235.283
x = 317.092 y = 254.842
x = 328.641 y = 272.072
x = 343.394 y = 286.197
x = 363.29 y = 291.23
x = 382.517 y = 288.372
x = 397.777 y = 275.101
x = 410.911 y = 258.241
x = 421.019 y = 240.182
x = 429.028 y = 220.902
x = 434.677 y = 201.25
x = 438.724 y = 182.13
x = 440.777 y = 163.203
```

x = 314.434 y = 142.784  
x = 323.061 y = 134.318  
x = 336.166 y = 131.351  
x = 350.011 y = 133.686  
x = 362.363 y = 138.718  
x = 390.543 y = 141.037  
x = 403.225 y = 137.271  
x = 416.249 y = 136.316  
x = 427.145 y = 140.556  
x = 432.458 y = 149.122  
x = 376.602 y = 157.122  
x = 376.577 y = 171.279  
x = 376.754 y = 185.021  
x = 376.752 y = 199.582  
x = 357.403 y = 209.623  
x = 364.947 y = 213.413  
x = 373.659 y = 216.406  
x = 381.697 y = 214.331  
x = 388.521 y = 210.804  
x = 326.251 y = 157.173  
x = 335.71 y = 152.831  
x = 345.414 y = 153.109  
x = 353.374 y = 159.724  
x = 344.507 y = 161.573  
x = 334.942 y = 161.155  
x = 394.179 y = 162.185  
x = 402.952 y = 156.821  
x = 412.325 y = 157.35  
x = 419.771 y = 162.669  
x = 411.891 y = 166.165  
x = 402.491 y = 165.492  
x = 341.528 y = 234.974  
x = 353.652 y = 234.773  
x = 364.368 y = 233.824  
x = 370.725 y = 236.465  
x = 377.429 y = 234.542  
x = 386.134 y = 236.597  
x = 395.089 y = 237.747  
x = 386.147 y = 246.528  
x = 376.973 y = 251.319  
x = 369.676 y = 252.347  
x = 361.574 y = 250.524  
x = 351.74 y = 245.043  
x = 343.893 y = 236.534  
x = 362.526 y = 239.565

```
x = 370.169 y = 241.324
x = 377.229 y = 240.421
x = 392.548 y = 238.944
x = 377.549 y = 241.843
x = 369.759 y = 242.778
x = 361.78 y = 241.34
```

spanExample. Detect results for 0 image:

detected: 1 faces

next face:

Rect:

x = 296

y = 77

w = 145

h = 206

score = 0.999943

spanExample. Detect results for 1 image:

detected: 1 faces

next face:

Rect:

x = 296

y = 77

w = 145

h = 206

score = 0.999943

spanExample. Redetect results for image[0]:

next face:

Rect:

x = 293

y = 96

w = 150

h = 192

score = 0.99776

spanExample. Redetect results for image[1]:

next face:

Rect:

x = 293

y = 96

w = 150

h = 192

score = 0.99776

creationExample. Detector instance with type: 4 was created successfully!

creationExample. bbox only result:

Rect:

```
x = 289
y = 94
w = 147
h = 185
score = 0.999999
```

```
creationExample. Detector instance with type: 5 was created successfully!
creationExample. bbox only result:
```

```
Rect:
```

```
x = 297
y = 97
w = 152
h = 184
score = 0.999986
```

```
creationExample. Detector instance with type: 6 was created successfully!
creationExample. bbox only result:
```

```
Rect:
```

```
x = 296.598
y = 77.0892
w = 145.655
h = 206.526
score = 0.999943
```

## Estimation example

### What it does

This example demonstrates how to use the detector and to estimate a smile, emotions, face attributes, quality, eye and head pose on an image.

### Prerequisites

This example assumes that you have already read the **FaceEngine Handbook** (or at least have it somewhere nearby for reference) and know some core concepts, like memory management, object ownership, and life-time control. This sample will not explain these aspects in detail.

### Example walkthrough

To get familiar with FSDK usage and common practices, please go through `example_extraction` first.

## How to run

Use the following command to run the example.

```
./example_estimation <some_image.ppm>
```

## Example output

Warped images with faces.

```
Detection 1
Rect: x=277 y=426 w=73 h=94
Attribure estimate:
gender: 0.999705 (1 - man, 0 - woman)
wearGlasses: 0.000118364 (1 - person wears glasses, 0 - person doesn't wear
glasses)
age: 17.7197 (in years)
Quality estimate:
light: 0.962603
dark: 0.974558
gray: 0.980648
blur: 0.955808
quality: 0.955808
Eye estimate:
left eye state: 2 (0 - close, 1 - open, 2 - noteye)
right eye state: 2 (0 - close, 1 - open, 2 - noteye)

Detection 2
Rect: x=203 y=159 w=63 h=89
Attribure estimate:
gender: 0.0053403 (1 - man, 0 - woman)
wearGlasses: 0.000911222 (1 - person wears glasses, 0 - person doesn't wear
glasses)
age: 16.1504 (in years)
Quality estimate:
light: 0.964406
dark: 0.971644
gray: 0.981737
blur: 0.955808
quality: 0.955808
Eye estimate:
left eye state: 0 (0 - close, 1 - open, 2 - noteye)
right eye state: 2 (0 - close, 1 - open, 2 - noteye)
```

## Extraction example

### What it does

This example demonstrates how to detect a face on an image, how to extract biometric data of that face (so called descriptor) and how to compare two descriptors. As the result of its work, the example program will tell you whether people shown on two images are actually the same person or not. It will also allow you to pick a threshold for such classification.

### Prerequisites

This example assumes that you have read the **FaceEngine Handbook** already (or at least have it somewhere nearby for reference) and know some core concepts, like memory management, object ownership and life-time control. This sample will not explain these aspects in detail.

The FaceEngine SDK is not about image/video loading, format decoding and such things. It's about facial recognition. So this example will not cover topics like how to load an image. There are no such functions built into the SDK either. For demonstration purposes we've included some freeware PPM reader code into the sample. We picked PPM image format because it is simple. You can read about it here: [https://en.wikipedia.org/wiki/Netpbm\\_format](https://en.wikipedia.org/wiki/Netpbm_format). To prepare data for tests you may use such tools as ImageMagick, Gimp, XnView.

### Example walkthrough

We present a rather rich face processing pipeline. In fact, it is a bit too rich for this sample, but it was made so intentionally to give you a clue of some real world usage scenarios.

### Preparations

SDK initialization. It is pretty straightforward and self-explanatory, so we will not discuss it here.

### Face detection

This stage is implemented in `extractDescriptor` function. At this stage we have an image. Presumably, there is a face somewhere in it and we would like to find it. For that purpose, a face detector is used. We use it like so:

```
// Create default detector, see faceengine.conf - "defaultDetectorType"
auto faceDetectorRes = faceEngine->createDetector();
if (faceDetectorRes.isError()) {
    ...
}
```

```

}
auto faceDetector = faceDetectorRes.getValue();
...
// Detect no more than 10 faces in the image.
enum { MaxDetections = 10 };

// Data used for detection.
fsdk::Detection detections[MaxDetections];
int detectionsCount(MaxDetections);
fsdk::IDetector::Landmarks5 landmarks5[MaxDetections];

// Detect faces in the image.
fsdk::ResultValue<fsdk::FSDKError, int> detectorResult = faceDetector->
    detect(
        image,
        image.getRect(),
        &detections[0],
        &landmarks[0],
        detectionsCount
    );
if (detectorResult.isError()) {
    ...
}
detectionsCount = detectorResult.getValue();

```

As the result we know whether we could detect faces (and how many of them) and what prevented us from achieving that.

### Descriptor extraction

This stage is implemented in `extractDescriptor` function. When we have a reliable face detection, we need to extract some data from it that can be used for comparison or *matching*. Such data is contained in descriptor object. A descriptor may be extracted from an image using face detection and landmarks coordinates. Later, one or multiple descriptors can be matched to determine face similarity.

```

// Create face descriptor.
auto resDescriptor = faceEngine->createDescriptor();
if (!resDescriptor) {
    ...
}
fsdk::IDescriptorPtr descriptor = resDescriptor.getValue();

// Extract face descriptor.

```

```
// This is typically the most time-consuming task.
fsdk::ResultValue<fsdk::FSDKError, float> descriptorExtractorResult =
    descriptorExtractor->extract(
        image,
        detections[bestDetectionIndex],
        landmarks5[bestDetectionIndex],
        descriptor
    );
if(descriptorExtractorResult.isError()) {
    ...
}
```

## Descriptor matching

This stage is pretty simple. We match two descriptors and see how similar they are. For that we use a descriptor matcher object.

```
// Descriptors similarity.
float similarity;

// Match 2 descriptors.
// Returns similarity in range (0..1],
// where: 0 means totally different.
//       1 means totally the same.
fsdk::ResultValue<fsdk::FSDKError, fsdk::MatchingResult>
    descriptorMatcherResult =
        descriptorMatcher->match(descriptor1, descriptor2);
if (descriptorMatcherResult.isError()) {
    ...
}

similarity = descriptorMatcherResult.getValue().similarity;
```

*Similarity* score tells how similar these descriptors are. Its value is in (0..1] range. Values near 1 tell us that the descriptors are very similar.

## Putting it all together

The `main` function just calls all the above in right order. Aside from that it parses command line and loads images. After all stages finish it writes the result.



## Human Detection example

### What it does

This example demonstrates how to use the human detector to get a human bbox (bounding box).

### Prerequisites

This example assumes that you have already read the **FaceEngine Handbook** (or at least have it somewhere nearby for reference) and know some core concepts, like memory management, object ownership, and life-time control. This sample will not explain these aspects in detail.

### Example walkthrough

To get familiar with FSDK usage and common practices, please go through `example_extraction` first.

### How to run

Use the following command to run the example.

```
./example_human_detection <some_image.ppm>
```

### Example output

```
detectHumanExample. Detect results for 0 image:
  detected: 1 humans
  next human:
    Rect:
      x = 121
      y = 2
      w = 462
      h = 543
    score = 0.997865
detectHumanExample. Detect results for 1 image:
  detected: 1 humans
  next human:
    Rect:
      x = 121
      y = 2
      w = 462
      h = 543
    score = 0.997865
```

```
detectHumanExample. Redetect results for image[0]:
  next human:
    Rect:
      x = 85
      y = 19
      w = 502
      h = 490
      score = 0.997068
detectHumanExample. Redetect results for image[1]:
  next human:
    Rect:
      x = 85
      y = 19
      w = 502
      h = 490
      score = 0.997068
```

## Reidentification example

### What it does

This example demonstrates how to detect a human body on an image, how to extract data of that body (so called human descriptor) and how to compare descriptors in different images. Available human descriptor versions HDV\_TRACKER\_HUMAN\_DESCRIPTOR\_VERSION = 102 and HDV\_PRECISE\_HUMAN\_DESCRIPTOR\_VERSION = 103, HDV\_REGULAR\_HUMAN\_DESCRIPTOR\_VERSION = 104. This example demonstrates how to:

- detect a human body in an image;
- extract data of that body (so called human descriptor);
- compare descriptors in different images.

### Prerequisites

This example assumes that you have already read the **FaceEngine Handbook** (or at least have it somewhere nearby for reference) and know some core concepts, like memory management, object ownership, and life-time control. This sample will not explain these aspects in detail.

### Example walkthrough

We present a rather rich human processing pipeline. In fact, it is a bit too rich for this sample, but it was made so intentionally to give you a clue of some real world usage scenarios.

## Preparations

SDK initialization. It is pretty straightforward and self-explanatory, so we will not discuss it here.

## Human detection

At this stage we have an image. Presumably, there is a human somewhere in it and we would like to find it. For that purpose, a human detector is used. We use it like so:

```
auto resHumanDetector = faceEngine->createHumanDetector();
if (!resHumanDetector) {
    ...
}
fsdk::IHumanDetectorPtr humanDetector = resHumanDetector.getValue();
...
// Detect no more than 10 humans in the image.
const int maxDetectionCount = 10;

// Detect humans in images.
fsdk::ResultValue<fsdk::FSDKError, fsdk::Ref<fsdk::IResultBatch<fsdk::Human
    >>> detectorResult = humanDetector->detect(
    fsdk::Span<const fsdk::Image>(images.data(), images.size()),
    fsdk::Span<const fsdk::Rect>(rects.data(), rects.size()),
    maxDetectionCount);
```

## Extraction

When we have a human detection, we need to extract some data from it that can be used for comparison or *matching*. Such data is contained in descriptor object. A descriptor may be extracted from an image using human detection. Later, one or multiple descriptors can be matched to determine human similarity.

```
// Create human descriptor. Available versions are presented in fsdk::
    HumanDescriptorVersion enum
auto resDescriptor = faceEngine->createDescriptor(fsdk::
    HDV_REGULAR_HUMAN_DESCRIPTOR_VERSION);
if (!resDescriptor) {
    ...
}
fsdk::IDescriptorPtr descriptor = resDescriptor.getValue();
```

```

// Create descriptor extractor. Available versions are presented in fsdk::
// HumanDescriptorVersion enum
auto resDescriptorExtractor = engine->createExtractor(fsdk::
    HDV_REGULAR_HUMAN_DESCRIPTOR_VERSION);
if (!resDescriptorExtractor) {
    ...
}
fsdk::IDescriptorExtractorPtr descriptorExtractor = resDescriptorExtractor.
    getValue();

// Extract human descriptor.
fsdk::ResultValue<fsdk::FSDKError, float> descriptorExtractorResult =
    descriptorExtractor->extractFromWarpedImage(
        humanWarpedImage,
        descriptor
    );
if(descriptorExtractorResult.isError()) {
    ...
}

```

## Descriptor matching

This stage is pretty simple. For example for matching we take only first descriptor in the first image and all detections in other images. For that we use a descriptor matcher object.

```

// Create descriptor matcher. Available versions are presented in fsdk::
// HumanDescriptorVersion enum
auto resDescriptorMatcher = engine->createMatcher(fsdk::
    HDV_REGULAR_HUMAN_DESCRIPTOR_VERSION);
if (!resDescriptorMatcher) {
    ...
}
fsdk::IDescriptorMatcherPtr descriptorMatcher = resDescriptorMatcher.
    getValue();

// Match descriptors.
// Returns similarity in range (0..1],
// where: 0 means totally different.
//         1 means totally the same.
fsdk::ResultValue<fsdk::FSDKError, fsdk::MatchingResult>
    descriptorMatcherResult =
        descriptorMatcher->match(descriptors[0][0], descriptors[i][j]);
if (descriptorMatcherResult.isError()) {

```

```
    ...  
}  
  
float similarity = descriptorMatcherResult.getValue().similarity;
```

*Similarity* score tells how similar these descriptors are. Its value is in (0..1] range. Values near 1 tell us that the descriptors are very similar.

## How to run

Use the following command to run the example.

```
./example_human_extraction <some_image1> <some_image2>
```

## Example output

```
Similarity of current descriptor and all human descriptors in other images  
in descent order:  
0.920459 :: image number is 1, detection number is 6  
0.63473 :: image number is 1, detection number is 1  
0.572745 :: image number is 1, detection number is 5  
0.562269 :: image number is 1, detection number is 2  
0.55178 :: image number is 1, detection number is 7  
0.539377 :: image number is 1, detection number is 3  
0.528099 :: image number is 1, detection number is 4  
0.511602 :: image number is 1, detection number is 8  
0.307247 :: image number is 1, detection number is 9
```

## Index example

### What it does

This example demonstrates how to create HNSW index from the batch of descriptors and use it to search for descriptor's closest matches efficiently.

### Prerequisites

This example assumes that you have already read the **FaceEngine Handbook** (or at least have it somewhere nearby for reference) and know some core concepts, like memory management, object ownership, and life-time control. This sample will not explain these aspects in detail. It is advisable to read HNSW Index chapter before going through this example.

Hnsw index creation is not supported on embedded and 32-bit desktop platforms.

## Example walkthrough

To get familiar with FSDK usage and common practices, please go through `example_extraction` first.

## How to run

Use the following command to run the example.

```
./example_index_hnsw <image.ppm> <imagesDir> <list> <threshold>
```

## Example output

```
Images: "images/Cameron_Diaz.ppm" and "Cameron_Diaz.ppm" belong to one
person.
Images: "images/Cameron_Diaz.ppm" and "Cameron_Diaz_2.ppm" belong to one
person.
Images: "images/Cameron_Diaz.ppm" and "Jason_Statham.ppm" belong to
different persons.
Images: "images/Cameron_Diaz.ppm" and "Jason_Statham_2.ppm" belong to
different persons.
Images: "images/Cameron_Diaz.ppm" and "Jennifer_Aniston.ppm" belong to
different persons.
Images: "images/Cameron_Diaz.ppm" and "Jennifer_Aniston_2.ppm" belong to
different persons.
```

## Infra Red Liveness Estimation example

### What it does

This example demonstrates how to use the detector and to estimate a liveness on an infra red image.

### Prerequisites

This example assumes that you have already read the **FaceEngine Handbook** (or at least have it somewhere nearby for reference) and know some core concepts, like memory management, object ownership, and life-time control. This sample will not explain these aspects in detail.

## How to run

Use the following command to run the example.

```
./example_ir <some_infra_red_image>
```

## Example output

Warped images with faces.

```
[LivenessIREstimatorExample] Detecting faces.  
Found 6 face(s).  
Detection 1  
Rect: x=29.1432 y=25.6546 w=72.6785 h=80.0122  
Face is real: true score: 0.999407  
Detection 2  
Rect: x=29.8662 y=233.287 w=72.7124 h=80.0827  
Face is real: true score: 1  
Detection 3  
Rect: x=149.572 y=26.2656 w=75.1791 h=86.9817  
Face is real: true score: 0.972015  
Detection 4  
Rect: x=151.426 y=232.602 w=75.6714 h=83.8235  
Face is real: true score: 0.999995  
Detection 5  
Rect: x=302.828 y=26.9954 w=73.9233 h=84.0742  
Face is real: false score: 0.270386  
Detection 6  
Rect: x=302.983 y=216.565 w=77.1824 h=97.5578  
Face is real: true score: 0.999971
```

## Liveness example

### What it does

This example demonstrates how to capture videostream from web camera, using OpenCV, create LivenessEngine core object and Liveness tests, pass frames to ILiveness instances in order to retrieve result of the test.

## Prerequisites

This example assumes that you have already read the **FaceEngine Handbook** and **LivenessEngine Handbook** (or at least have them somewhere nearby for reference) and know some core concepts, like memory management, object ownership, and life-time control. This sample will not explain these aspects in detail.

This example requires installed OpenCV library with imgproc, highgui and videoio modules, for additional information refer to <https://opencv.org/>.

## Example walkthrough

This example consists of the following stages:

### Stage 0. Preparations

Argument parsing, SDK initialization, testing sequence preparation, video capturing.

### Stage 1. Main cycle

This stage is implemented in `while(process)` cycle. At this stage we grab frame from video capture, convert it from BGR to RGB, wrap it into FaceEngine structures, pass Image to liveness instance and output picture and action calls to user via Video window and console.

```
while(process){
    //Grab frame
    cap >> output;
    if(!output.empty()){
        cv::Mat rgb;
        cv::cvtColor(output,rgb,CV_BGR2RGB);
        //Wrap frame into fsdk container
        Image img(rgb.cols, rgb.rows, fsdk::Format::R8G8B8, rgb.data);
        ResultValue<LSDKError,bool> result = liveness->update(img);
        if((result.getResult()==lsdk::ERR_NOT_READY)&&start){
            start = false;
            //Ask for an action
            std::clog << advices[i] << std::endl;
        }
        if(result.isOk()){
            //Save result
            success = success & result.getValue();
            process = false;
        }
    }
}
```



```

        //Output video
        cv::Mat mirror;
        cv::flip(output, mirror, 1);
        cv::namedWindow("Video", CV_WINDOW_AUTOSIZE);
        cv::imshow("Video", mirror);
        cv::waitKey(1);
    } else process = false;
}

```

As the result we get flag whether or not liveness detection was successful. This cycle repeats N times (N is specified as program argument). Program result equals to Bitwise AND of every cycle iteration result. if result == true then liveness is successful, otherwise not.

### How to run

Use the following command to run the example.

```
./example_liveness <camera_number> <test_number>
```

## PPE estimator aggregation example

### What it does

Aggregation example loads video frames and starts to track multiple persons on it, estimating their body position and ppe attributes. After all tracking information for each person has been collected example aggregates these results and prints them out to the console.

### Prerequisites

This example assumes that you have read the **FaceEngine Handbook** already (or at least have it somewhere nearby for reference) and are familiar with some core concepts, like memory management, object ownership and life-time control. This sample will not explain these aspects in detail.

### Example walkthrough

To get familiar with FSDK usage and common practices, please go through example\_extraction first.

### How to run

```
./ppe_aggregation {image_list_path} [data_path]
```

Note: `image_list_path` is a text file that contains absolute paths to each video frame to process. For example, if you have frames that have prefix `img_*`, to make a text file from it you could run the following command:

```
find "$PWD" | sort | grep img_ > list.txt
```

Sample video frames for this example located in the following path

```
${FSDK_ROOT}/examples/images/ppe_frames
```

## Building for linux

### Building command (from `FSDK_ROOT/examples/build`):

```
$ cmake -DFSDK_ROOT=<absolute_path_to_fsdk_root_dir>
-DTSDK_ROOT=<absolute_path_to_tsdn_root_dir>
-DWITH_PPE_EXAMPLE=ON
-DCMAKE_INSTALL_PREFIX=<absolute_path_to_installation_dir_of_your_choise
>
-DCMAKE_MODULE_PATH=<absolute_path_to_fsdk_root_dir>/examples/cmake ../
example_ppe_aggregation

$ cmake --build . --config Release --target install
```

## Building for windows

### Building command (from `FSDK_ROOT/examples/build`):

```
$ cmake -G "Visual Studio 14 2015 Win64"
-DFSDK_ROOT=<absolute_path_to_fsdk_root_dir>
-DTSDK_ROOT=<absolute_path_to_tsdn_root_dir>
-DWITH_PPE_EXAMPLE=ON
-DCMAKE_INSTALL_PREFIX=<absolute_path_to_installation_dir_of_your_choise
>
-DCMAKE_MODULE_PATH=<absolute_path_to_fsdk_root_dir>/examples/cmake ../
example_ppe_aggregation

$ cmake --build . --config Release --target install
```

## Visualising output

Example visualisation is turned off by default to simplify demo's build step. If you want to visualise demo output you should also pass the following additional flags to cmake:

```
-DPPE_VISUALISE_DATA=ON -DOpenCV_DIR=<absolute_flag_to_opencv>
```

With visualisation turned on example saves frames processed by Track Engine in the current directory and plots bounding boxes, human keypoints and track ids on top of it.

Note: since human keypoints redetection is not supported yet, demo temporarily turns off redetection feature in the track engine.