Skeleton Tracking SDK

Getting Started

cubemos

Introduction

The cubemos SkeletonTracking SDK is a unifying approach to serve a deep learning based 2D/3D skeletal tracking functionality on Windows and Unix based systems. It is accessible by various programming interfaces. Currently C, C++, C#, Python and unity is supported.

Content

System Requirements

For Windows based operating systems
- OS: Windows 10
- C++: Microsoft Visual Studio 2017 (recommended 15.8)
- C#: Microsoft .NET Framework version 4.0
- Python: Version >= 3.6
- Unity: Unity 2018.4.x (LTS)

For Unix based operating systems
- OS: Ubuntu 18.04 (LTS) (tested),
  Other Debian based system are possible (untested)
- C++: Clang 6.0, gcc 7.4
- Python: Version >= 3.6

Hardware:
- Platform: x64
- CPUs: 6th to 10th generation Intel® Core™ and Xeon® Processors
- GPUs: Intel® Iris® Pro, Intel® HD Graphics 520, 530, 630
- VPUs: Intel® Movidius™ Neural Compute Stick 2
- 3D: 3D Supported Camera among others:
  Intel® RealSense™ D415, D435
  FRAMOS Depth Camera D435e

Windows

Installation and license activation

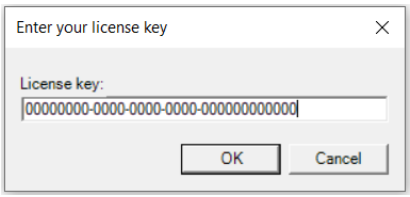
1. Download the installation package
   If you don't have one, get it here.

2. Run the installer `cubemos-SkeletonTracking_3.x.x.x.exe` as an administrator.

3. Restart your machine.

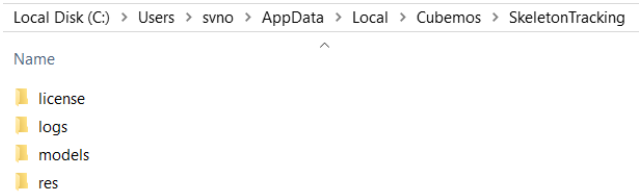
4. To activate the software for the first time run `post_installation.bat` script available in `C:\Program Files\Cubemos\SkeletonTracking\scripts`.

5. In the opened dialog enter the license key string you acquired.



The following steps take place automatically (a.,b.,c.,d.)

   a. The `post_installation.bat` script generates the license file (`cubemos_license.json`) and copies it together with the neural network models and test-images to the cubemos folder in `%LOCALAPPDATA%`.



cubemos SkeletonTracking %LOCALAPPDATA% folder after post installation step

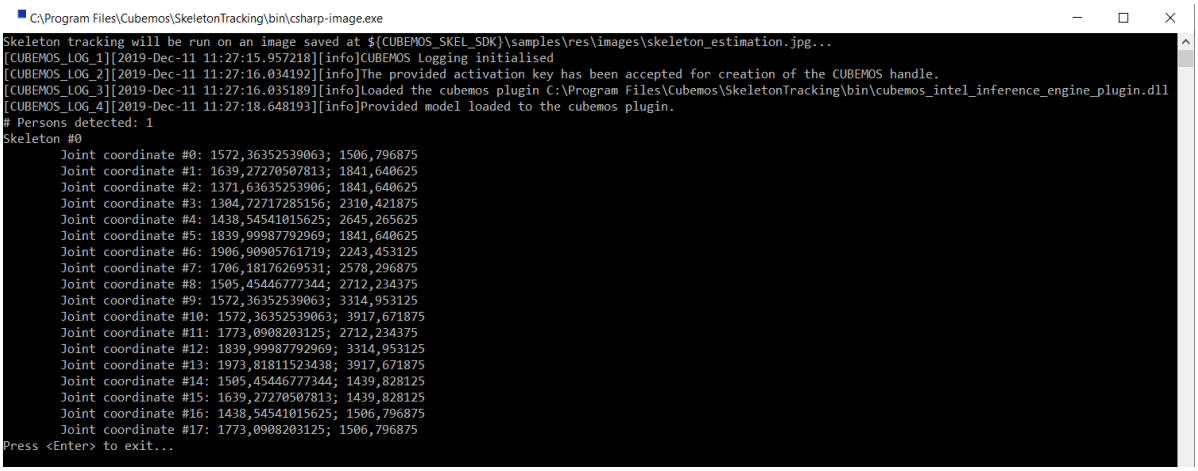
   b. By setting up %LOCALAPPDATA% folder you can access the SkeletonTracking ressources in the most flexible way without the need of permanent administrator permissions.

   c. The activation is performed.
   d. activation_key.json is saved in the license folder.
   e. (Optionally) In the post_installation.bat script you can choose to generate a Visual Studio 2017 Solution for the cubemos samples, this will copy the sample sources to C:\Users\%USERNAME%\Cubemos-Samples.

If you do not possess a valid license file, please contact cubemos at support@cubemos.com to receive a valid product license.

Follow the steps on the next page…

6. After installation, you are able to run the installed samples located in `C:\Program Files\Cubemos\SkeletonTracking\bin\`.
   If after running the `csharp-image.exe` sample, you see the output as below your installation was successful.



List of joint coordinates of example image after successful installation

For more information about the samples you can refer to the section Running cubemos Samples.

7. To integrate the SDK into your project you can choose among the following options described in this guide:
   a. C# with NuGet and Visual Studio: Windows only (page 6)
   b. C# with CMake: Windows only (page 14)
   c. Python: Windows (page 10) and Linux (page 17)
   d. C++ with CMake: Windows (page 14) and Linux (page 18)
   e. Unity: Windows only (page 12)
8. The API documentation and samples descriptions can be found in the installation folder at `%CUBEMOS_SKEL_SDK%\docs\doc_doxygen\html\index.html`

Note: during the **Windows** installation the following environment variable is created and added to your system PATH:
`%CUBEMOS_SKEL_SDK% | C:\Program Files\Cubemos\SkeletonTracking`

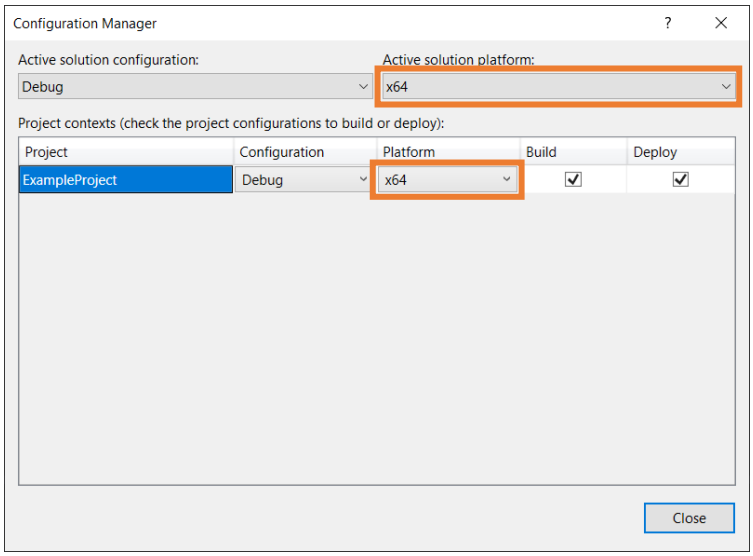
Getting Started with C# and Visual Studio

Prerequisites:

- Follow all steps of the section “Windows Installation” on page 4
- Install Microsoft Visual Studio 2017

1. Open existing project or create a **new C# project (File >> New >> Project)**
   e.g. Visual C# **>>** Windows Classic Desktop **>>** Console App (.NET Framework)

2. Ensure **x64 architecture** is set as target platform in Visual Studio.
   a. Open **Configuration Manager (Build** >> **Configuration Manager)**
   b. Select x64 as platform for all configurations as shown in the figure below
   c. Open the drop-down menu “Active solution platform”
   d. Click on **“new”**
   e. Select **“x64”**

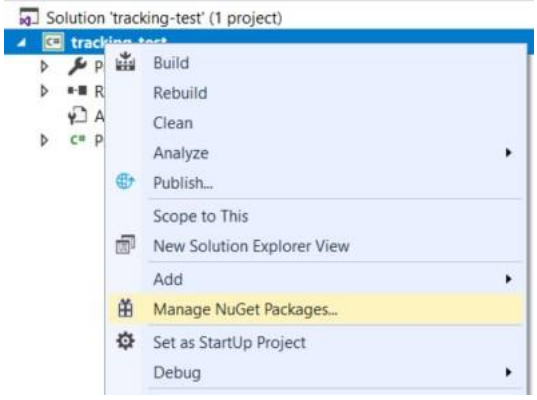


Setting the target platform to x64

3. Update NuGet package sources
   a. Open **Package Sources (Tool >> Options >> NuGet Package Manager >> Package Sources)**
   b. Add new package source with name “cubemos Skeleton Tracking Sources” which points to the NuGet folder in the installation directory of Skeleton Tracking SDK
   `(C:\Program Files\Cubemos\SkeletonTracking\wrappers\csharp\nuget)`

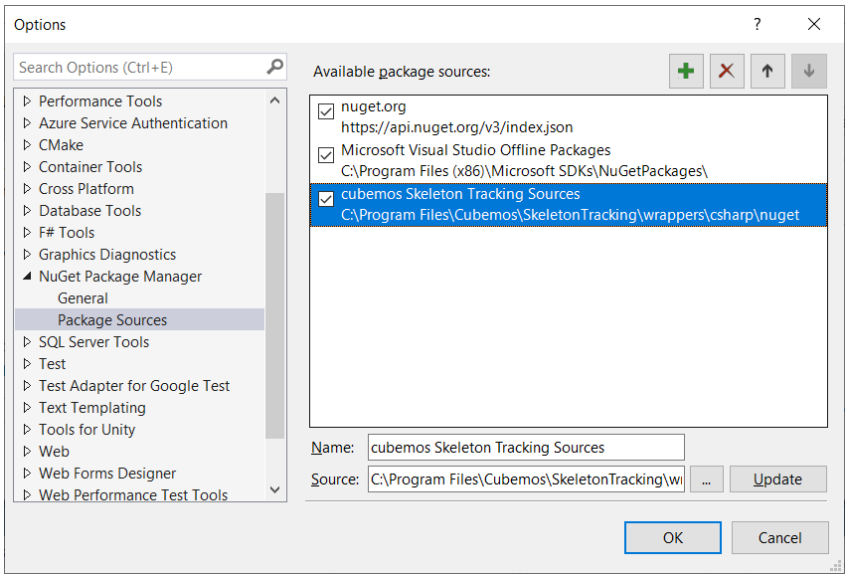
Follow the steps on the next page…

4. Include NuGet package into project
    a. Open Manage NuGet Packages as shown in the figure below



Opening "Manage NuGet Packages" window

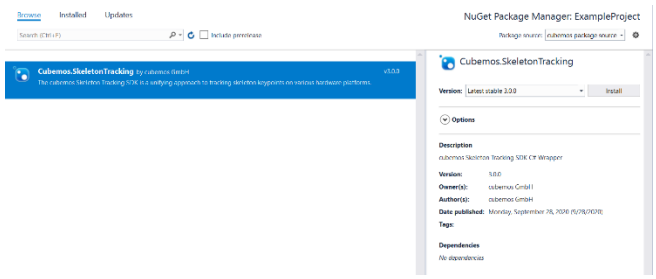
    b. Click on "**Browse**"
    c. In the "**Package Source**" drop down menu select the newly installed package source of step 3.b.
    d. Select "cubemos Skeleton Tracking Sources"



Selecting Skeleton Tracking SDK NuGet package

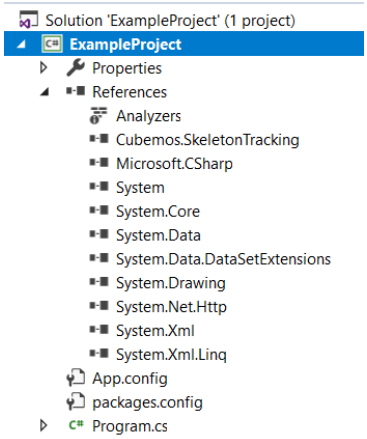
Follow the steps on the next page…

e. Select the NuGet package **Cubemos.SkeletonTracking** and click "**Install**" as shown below



Available cubemos Skeleton Tracking SDK packages

f. Check correct installation as shown in the solution explorer



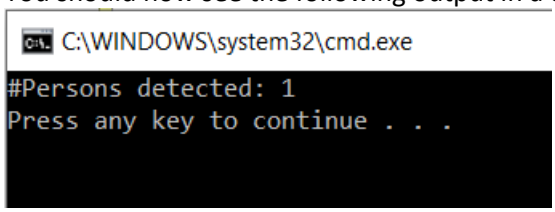
Successfully included NuGet packages

Follow the step on the next page…

g. Copy and paste the following code into your main method.

```
using System;
///
/// This sample shows how to use the cubemos SkeletonTracking C# API in a console application
///
namespace Cubemos.Samples
{
    class Program {
        static void Main(string[] args)
        {
            // Read an RGB image of any size
            String cubemosSkelData = Environment.GetFolderPath(Environment.SpecialFolder.LocalApplicationData) + "\\Cubemos\\SkeletonTracking";
            System.Drawing.Bitmap image = new System.Drawing.Bitmap(cubemosSkelData + "\\res\\images\\skeleton_estimation.jpg");

            // Decalre results container
            System.Collections.Generic.List<Cubemos.SkeletonTracking.Api.SkeletonKeypoints> skeletonKeypoints;

            // Create cubemos API handle for Intel® Inference Plugin and specify the folder with the license key
            var skeletontrackingApi = new Cubemos.SkeletonTracking.Api(cubemosSkelData + "\\license");

            // Load the CPU model
            skeletontrackingApi.LoadModel(Cubemos.TargetComputeDevice.CM_CPU,
                                cubemosSkelData + "\\models\\fp32\\skeleton-tracking.cubemos");

            // Send inference request and get the poses
            skeletontrackingApi.RunSkeletonTracking(ref image, 128, out skeletonKeypoints);

            //Output the number of detected skeletons
            Console.WriteLine("#Persons detected: " + skeletonKeypoints.Count);
        }
    }
}
```

h. Build and run the application.
You should now see the following output in a console window:



Getting Started with Python

Prerequisites:

- Follow all steps of the section “Windows Installation” on page 4
- Install Python 3.6 or 3.7 (Either as standalone package from here or create an environment with Anaconda)
- **Python 3.8 is currently not supported as pyrealsense2 is not available for this version.**

Option 1: Via PowerShell

1. Install the cubemos python modules with the following command

```
pip install --find-links=$env:CUBEMOS_SKEL_SDK\wrappers\python `
cubemos.skeletontracking
```

2. Execute the python sample

   a. Move into provided python sample folder

```
cd $env:CUBEMOS_SKEL_SDK\samples\python\
```

   b. Install the required python modules

```
pip install -r requirements.txt
```

   c. Run the sample on a local image

```
python skeleton-tracking-image.py `
..\res\images\skeleton_estimation.jpg
```

   To save the results you can use an additional parameter:

```
python skeleton-tracking-image.py -o C:\tmp\output.jpg `
  ..\res\images\skeleton_estimation.jpg
```

Option 2: Via cmd

1. Install the cubemos python modules with the following command

```
pip install --find-links="%CUBEMOS_SKEL_SDK%\wrappers\python" ^
cubemos.skeletontracking
```

2. Execute the python sample

   a. Move into provided python sample folder

```
cd "%CUBEMOS_SKEL_SDK%\samples\python\"
```

   b. Install the required python modules

```
pip install -r requirements.txt
```

   c. Run the sample on a local image:

```
python skeleton-tracking-image.py ^
..\res\images\skeleton_estimation.jpg
```

   To save the results you can use an additional parameter like below:

```
python skeleton-tracking-image.py -o C:\tmp\output.jpg ^
..\res\images\skeleton_estimation.jpg
```

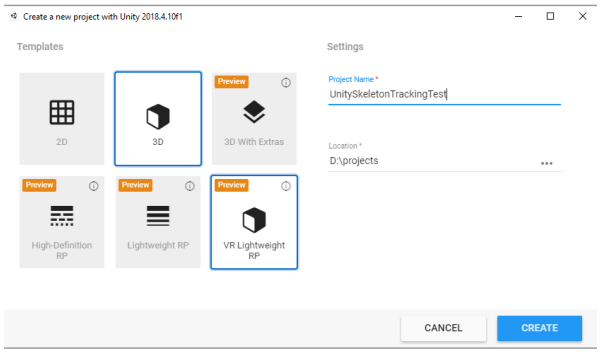
Getting Started with Unity

This section explains steps needed to get started with Skeleton Tracking in a Unity Project. The goal is to read the Skeleton Joints in the Unity Editor.

Prerequisites:

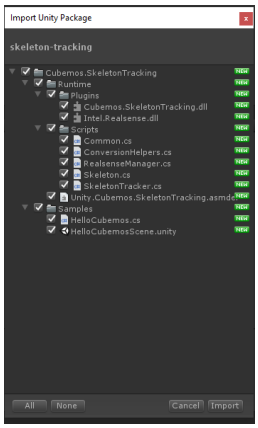
- RealSense D435 or D415 (for full compatibility with this guide)

1. Install Unity version 2018.4.x
2. Open Unity Hub.
3. Create new project: **Projects >> New** on the version 2018.4.11.
4. Select 3D project



Create unity sample project

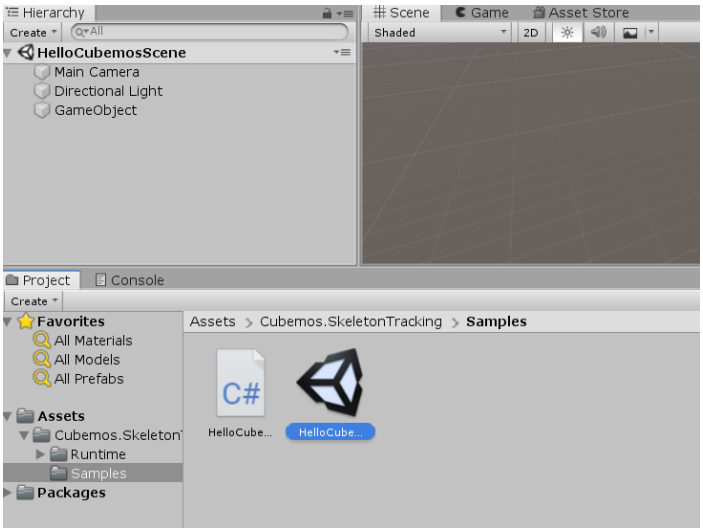
5. Import the cubemos SkeletonTracking package
    1. Go to **Assets >> Import package >> Custom package**
    2. Select `C:\Program Files\Cubemos\SkeletonTracking\wrappers\unity3D\skeleton-tracking.unitypackage`
    3. Ensure to import all available assets of the package



Select assets of cubemos SkeletonTracking SDK

Follow the steps on the next page…

6. Open the skeleton-tracking scene **Assets >> Cubemos.SkeletonTracking >> Samples**



Select HelloCubemos scene

7. Attach a RealSense D415 or D435 with USB 3.0 and position it in the view of a person.

8. Start the playback by pressing the Play button. The number of detected persons and its corresponding skeletal joints will be printed to the console.

Unity console output example

Building SkeletonTracking SDK Samples with CMake on Windows

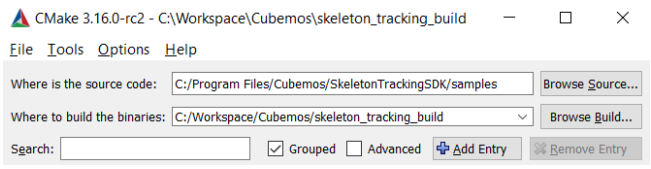
During the **Windows** installation the following environment variable is created
`%CUBEMOS_SKEL_SDK% | C:\Program Files\Cubemos\SkeletonTracking`

`%CUBEMOS_SKEL_SDK%\samples` contain the source code of the samples.

If you opted to build the samples during the activation with the post_installation.bat script the Visual Studio 2017 solution was already generated in the `C:\Users\%USERNAME%\Cubemos-Samples\build`. In that case you can open the cubemos-samples.sln solution and proceed with step 5 below.

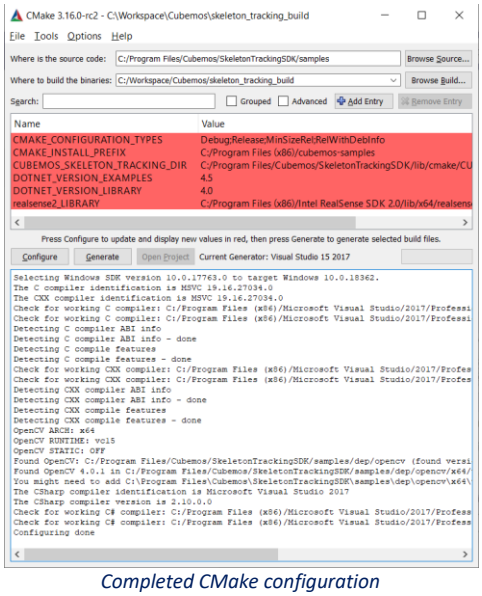
CMake 3.10.0 or newer is required.

1. Open CMake GUI
2. Select the `%CUBEMOS_SKEL_SDK%\samples` folder as source and choose a folder to save build results where you have the write permissions



Setting samples source and build folders in CMake

3. Configure and then generate VS17 solution
   (Please be aware to use the x64 platform and the Visual Studio 15 2017 generator)



Completed CMake configuration

4. Open the Visual Studio project by pressing the “Open Project” button
5. Inside Visual Studio, build the samples with **Build** >> **Build solution**
6. Run the samples binaries in the folder you specified above (for example `C:/Workspace/Cubemos/skeleton_tracking_build/cpp/image/Debug/cpp-image.exe`).

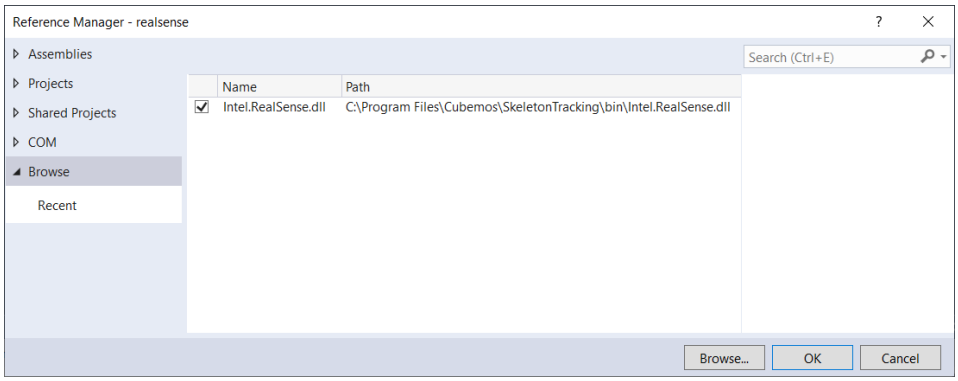
Using SkeletonTracking SDK with Intel® RealSense™ on Windows

The cubemos SkeletonTracking SDK provides samples for using the Intel® RealSense™ D415 or D435 cameras as image input device and pre-built Windows x64 binaries for librealsense 2.36. Depending on your build option you can choose one of the options below to integrate librealsense in your C# project.

Option 1: Extend SkeletonTracking SDK Samples using CMake
The easiest way to start using RealSense™ is building the cubemos samples with CMake and extending the `csharp-realsense` or `cpp-realsense` for the corresponding operating system. To create a VS2017 solution for the samples please follow the steps described on page 14.

Option 2: Use Intel® RealSense™ in C# Visual Studio Project
Add the RealSense™ package to your project references by selecting the entry **"Add Reference"** in the context menu and adding `Intel.RealSense.dll`. The SkeletonTracking SDK provides a pre-build binary `Intel.RealSense.dll` for `librealsense-2.36` in the `%CUBEMOS_SKEL_SDK%\bin` folder.

To acquire images from RealSense™ and to run human pose estimation please refer to the samples. Go to: `%CUBEMOS_SKEL_SDK%\samples\csharp\realsense\Program.cs`



For option 2 add the RealSense™ reference from the cubemos SkeletonTracking SDK installation folder

Linux

Installation and license activation

1. Download the installation package
   If you don't have one, get it here.

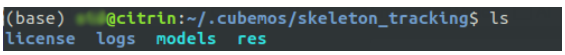
2. Install `cubemos-SkeletonTracking_3.x.x.x.deb` as root with
   `sudo apt-get install ./cubemos-SkeletonTracking_*.deb`

3. Log out and log in again to make sure that all environment variables are loaded correctly.

4. To activate the software for the first time, execute the following script which copies images and models to the user directory:
   ```
   cd /opt/cubemos/skeleton_tracking/scripts/
   bash post_installation.sh
   ```

5. In the command line dialog, enter your user password in order to execute steps necessary for installing the developer dependencies and setting up the usb device rules for the Intel Neural Compute Stick.
6. Enter your acquired license key without quotes when asked for.

7. Then the **following steps take place automatically (a., b., c., d.)**
   a. The `post_installation.sh` script generates the license file (`cubemos_license.json`) and copies it, together with the neural network models and test-images, to `/home/$USER/.cubemos/skeleton_tracking/`

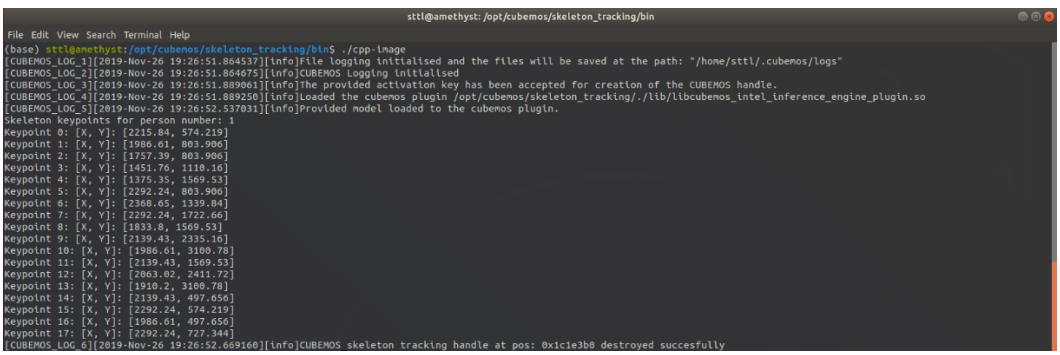


cubemos SkeletonTracking home folder after post installation step

   b. The activation is performed
   c. `activation_key.json` is saved in the license folder
   d. make files for the cubemos samples are generated and built into `/home/$USER/cubemos-samples/`

If you do not possess a valid license file, please contact cubemos at support@cubemos.com to receive a valid product license.

8. After installation, you are able to run the samples located in `/opt/cubemos/skeleton_tracking/bin/`, for example:
   `/opt/cubemos/skeleton_tracking/bin/cpp-image`



List of joint coordinates of example image after successful installation

During the **Linux** installation the following environment variable is created

`$CUBEMOS_SKEL_SDK | /opt/cubemos/skeleton_tracking`

Getting Started with Python

Prerequisites:

- Follow all steps of the section "Linux Installation" on page 19.
- Install Python >= 3.6 and pip with
  `sudo apt-get install python3-pip python3-venv`

1. Create a python virtual environment and activate it
   ```
   python3 -m venv ~/cubemos-samples/py_venv
   source ~/cubemos-samples/py_venv/bin/activate
   ```

2. Install the cubemos python modules with the following two commands
   ```
   pip3 install numpy
   pip3 install --find-links="$CUBEMOS_SKEL_SDK/wrappers/python"
   --no-index cubemos.skeletontracking
   ```

3. Execute the python sample
   a. Move into provided python sample folder
      `cd "$CUBEMOS_SKEL_SDK/samples/python/"`

   b. Install the required python modules
      `pip3 install -r requirements.txt`

   c. Run the sample on a local image with the command below:
      ```
      python3 skeleton-tracking-image.py
      ../res/images/skeleton_estimation.jpg
      ```

      To save the result image you can use an additional parameter as in the command:
      ```
      python3 skeleton-tracking-image.py -o ~/output.jpg
      ../res/images/skeleton_estimation.jpg
      ```
      The path to the output image should be in a writable location, here we would write in the home directory.

Building SkeletonTracking SDK Samples with CMake on Linux

During the **Linux** installation the following environment variable is created
`$CUBEMOS_SKEL_SDK | /opt/cubemos/skeleton_tracking`

`$CUBEMOS_SKEL_SDK/samples` contains the source code of the samples.

1. Install CMake and build tools with
   ```
   sudo apt-get install cmake build-essential
   ```

2. Create a directory where the built samples should be stored
   ```
   mkdir /path/buildDir
   cd /path/buildDir
   ```

3. Run cmake and build the samples
   ```
   cmake $CUBEMOS_SKEL_SDK/samples
   make
   ```

Using SkeletonTracking SDK with Intel® RealSense™ on Linux

The cubemos SkeletonTracking SDK provides samples for using the Intel® RealSense™ D415 or D435 cameras as image input device.

Extend SkeletonTracking SDK Samples using CMake

The easiest way to start using RealSense™ is building the cubemos samples with CMake and extending the `cpp-realsense`. To build the samples please follow the steps described in section Building SkeletonTracking SDK Samples with CMake on Linux

Developer Documentation

After installation you find a detailed description of the SkeletonTracking API in the developer documentation provided in the following path
`%CUBEMOS_SKEL_SDK%\docs\doc_doxygen\html\index.html`

The developer documentation covers the following content:

General
cubemos Skeleton Tracking API

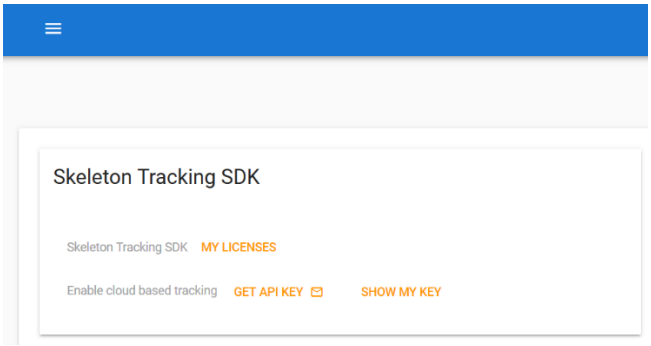
C++
Skeleton Tracking on the webcam or video input - C++
Skeleton Tracking on Intel® RealSense™ Streams - C++
Skeleton Tracking on a single image - C++

C#
cubemos Skeleton Tracking API C# Bindings
Skeleton Estimation on a local image - C#
Skeleton Estimation on Intel® RealSense™ RGB Image - C#
Skeleton Tracking on a sequence of Images read from a disk- C#
Skeleton Tracking on Intel® RealSense™ Streams with 3D - C#

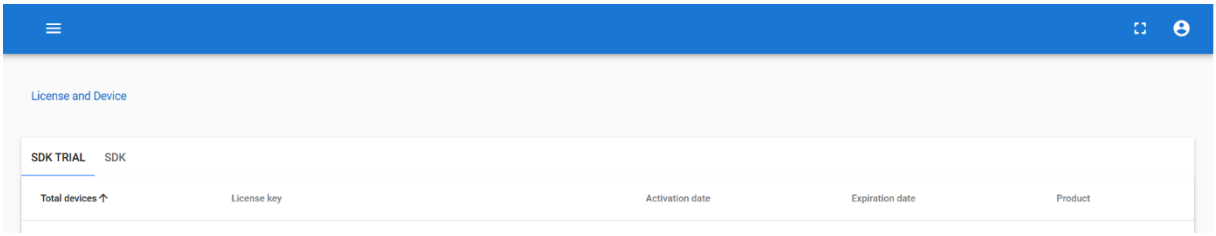
Python
cubemos Skeleton Tracking API Python Bindings
Skeleton Estimation on a single image – Python
Skeleton Tracking on a webcam or video input – Python
Skeleton Tracking on Intel® RealSense™ Streams with 3D – Python

Managing your licenses

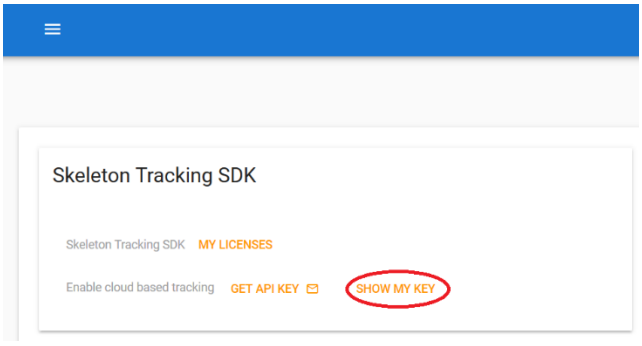
You can view your Skeleton Tracking SDK licenses in the cubemos Dashboard.
After you login with the E-Mail you used to download the Skeleton Tracking SDK you would see the options below.



Licenses and Devices.
The dashboard view under “My licenses” allows you to see what license keys you have and on how many devices they have been activated. You can also see the expiration dates of the trial licenses.

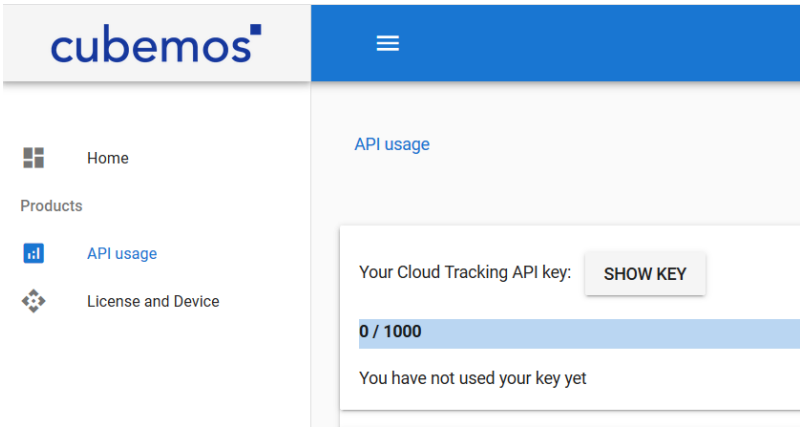


Generating keys.
By clicking on the “Show my key” you can generate a free key to use the cloud based tracking.



Quotas.
To check the remaining number of requests for your key you can go to the “API Usage” on the left side of the view. If the options on the left side are not visible you can show them by clicking on the menu  .

The “API usage” shows how many of the total requests have been used this week. The number of the available weekly requests is determined by your usage plan. In the example below 0 out of 1000 weekly requests have been used.



Running cubemos Samples

For both operating systems we provide several sample applications for the supported programming environments. For almost all samples the source code is provided as well.
Below you find a list of available sample applications for Windows and Linux.

For Windows:
The executables can be found in
`C:\Program Files\Cubemos\SkeletonTracking\bin\.`

The python script can be found in
`C:\Program Files\Cubemos\SkeletonTracking\samples\python.`

The images used in the samples are found in
`C:\Users\%USER%\AppData\Local\Cubemos\SkeletonTracking\res\images`

`csharp-interactive.exe`
A C# Interactive GUI Application to demonstrate the options and capabilities of the SDK with RealSense™.

`csharp-image.exe`
A minimal C# console sample to demonstrate the usage of SDK with image files.

`csharp-realsense.exe`
A minimal C# console sample to demonstrate the usage of SDK with Intel® RealSense™.

`csharp-videoseries.exe`
A minimal C# console sample to demonstrate the usage of tracking with sequence of consecutive images.

`cpp-image.exe`
A minimal C++ console sample to demonstrate the usage of SDK with image files.

`cpp-webcam`
A C++ console sample to demonstrate the usage of edge and cloud based tracking and asynchronous calls with a Webcam.

`cpp-realsense.exe`
A minimal C++ console sample to demonstrate the usage of SDK with Intel® RealSense™. It supports the RealSense D435, D415 and L500 (LiDAR).

`skeleton-tracking-image.py`
A minimal Python sample to demonstrate the usage of SDK with image files.

`skeleton-tracking-realsense.py`
A Python sample to demonstrate the usage of SDK with Intel® RealSense™ and performing skeleton estimation in 3D.

`skeleton-tracking-webcam-cloud.py`
A Python sample to demonstrate the usage of edge and cloud based tracking with a Webcam

For Linux
The executables can be found in:
`/opt/cubemos/skeleton_tracking/bin`

The python script can be found in:
`/opt/cubemos/skeleton_tracking/samples/python.`

The images used by the samples are found in:
`/home/$user/.cubemos/skeleton_tracking/res/images`

`cpp-image.exe`
A minimal C++ console sample to demonstrate the usage of SDK with image files.

`cpp-webcam`
A C++ console sample to demonstrate the usage of edge and cloud based tracking and asynchronous calls with a Webcam.

`cpp-realsense.exe`
A minimal C++ console sample to demonstrate the usage of SDK with Intel® RealSense™. It supports the RealSense D435, D415 and L500 (LiDAR).

`skeleton-tracking-image.py`
A minimal Python sample to demonstrate the usage of SDK with image files.

`skeleton-tracking-realsense.py`
A Python sample to demonstrate the usage of SDK with Intel® RealSense™ and performing skeleton estimation in 3D.

`skeleton-tracking-webcam-cloud.py`
A Python sample to demonstrate the usage of edge and cloud based tracking with a Webcam

Using CMake to Integrate the SDK into your Project

This section demonstrates integrating the cubemos SkeletonTracking SDK into your existing project using CMake.

The SDK contains config files for CMake which can be invoked by the CMake command `find_package()`.

The following commands are used in a CMake script to include the SkeletonTracking SDK into your application.

```
%Define project
project(example-project LANGUAGES CXX)
%Define the folder where CMake config files are stored
set(CUBEMOSSDK "$ENV{CUBEMOS_SKEL_SDK}/lib/cmake/CUBEMOS_SKELETON_TRACKING")
%Find the cubemos library (C-API)
find_package(CUBEMOS_SKELETON_TRACKING REQUIRED CONFIG PATHS "${CUBEMOSSDK}")
%Create an executable project
add_executable(${PROJECT_NAME} main.cpp)
%Link cubemos to this executable project
target_link_libraries(${PROJECT_NAME} PRIVATE cubemos_skeleton_tracking)
```

Troubleshooting

License File Error – File Missing

Error Message
fatal error:
License file cubemos_license.json provided during the software download needs to be present in the execution folder in order to perform the one time
activation over the cubemos activation server.

Possible Reason
The license file, cubemos_license.json, cannot be found.

Solution:
Make sure a valid license file `cubemos_license.json` is in the cubemos license folder.

On Windows, the license folder location is `%LOCALAPPDATA%\Cubemos\SkeletonTracking\license`.
On Linux it is `~/.cubemos/skeleton_tracking/license`.
To perform or repeat activation run `post_installation.bat` on Windows or `post_installation.sh` on Linux

---

License File Error – Trial License Expired

Error Message
error: The provided license expired. Please provide a valid license to continue using the cubemos SkeletonTracking API.

Possible Reason
The trial license has expired.

Solution:
Remove the activation key file, `activation_key.json`, and replace the expired license (`cubemos_license.json`) in the config folder with a valid license file. The license will be activated during the next use of the SDK.

On Windows, the default location is `%LOCALAPPDATA%\Cubemos\SkeletonTracking\license`.
On Linux it is `~/.cubemos/skeleton_tracking/license`.

Sample Application Start Fails – (Windows only)

Issue:
The sample does not start from the Start Menu (Windows only).

Possible Reason
The installation was interrupted or unfinished.

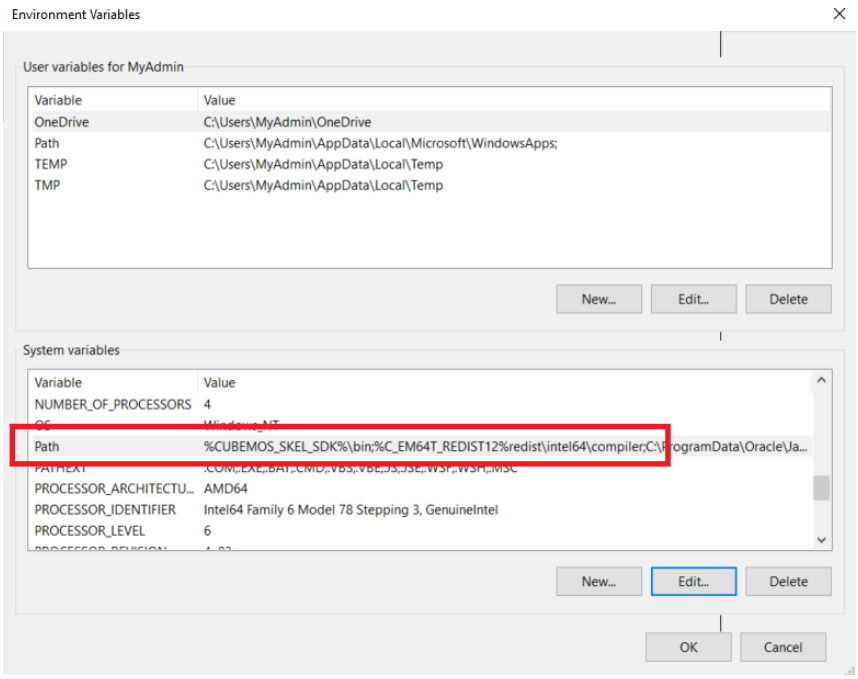
Solution:
Make sure the installer runs successfully. After installation a restart is necessary on Windows. The compiled sample is in the `%$CUBEMOS_SKEL_SDK%\bin` folder and can be run with a double-click.

Installed or built samples do not start (Windows)

Issue:
*The samples do not start and show an error message that *.dll is missing (for instance libmmd.dll or svml_dispmd.dll)*

Possible Reason
The cubemos SkeletonTracking binaries or some dependent binaries are not in the system path or the system was not restarted after the SDK installation.

Solution:
Verify that the PATH variable contains the entries: `%CUBEMOS_SKEL_SDK%\bin` and `%C_EM64T_REDIST12%redist\intel64\compiler`. A system restart might be necessary if you add these variables on your own. Correctly set system PATH should look like in the image below.



C# Sample Demo – Video freeze

Issue:
The video stream in the "csharp-interactive" sample is frozen.

Possible Reason
Connection to the camera was lost.

Solution:
Verify the physical camera connection and restart streaming with the Play button.
If that does not help:
1. Stop streaming
2. Unplug the camera
3. Reattach the camera
4. Start streaming

Cloud Tracking not working

Issue:
Sdk uses offline (edge) tracking even though cloud tracking was selected in the code.

Possible Reason (1)
No internet connection / internet connection lost during operation. Corresponding error messages in the log are:

```
[error]Exception thrown in file ***https_client.cpp and line
119 with error message: "HTTP Request failed: "Couldn't resolve
host name". Return code: 1"
```

Solution
Make sure your internet connection is working and restart the tracking.

Possible Reason (2)
Api-key is missing/invalid. Corresponding error message in the log:

```
[error] Exception thrown in file ***reid_client.cpp and line 86
with error message: "Operation ** returned with status code:
403. Details: "{"message":"Forbidden"}". Return code: 1"
```

Solution
Get a valid cloud tracking api key. See the section Managing your licenses in this document for details.

Possible Reason (3)
Api-key quota exceeded. Corresponding error message in the log:

```
[error]Exception thrown in file ***reid_client.cpp and line 120
with error message: "Operation identify returned with status
code: 429. Details: "{"message":"Limit Exceeded"}". Return
code: 1"
```

Solution
Check your api-key usage in the dashboard (see Managing your licenses) to see whether you have free requests left. If not, you can either wait until your weekly limit is renewed or purchase a new api-key.

Unity Wrapper – realsense2.dll not found

Issue:
DllNotFoundException: realsense2

Possible Reason:
`%CUBEMOS_SKEL_SDK%\bin` is not in the system path or the PC was not restarted after the installation.

Solution:
Verify that `%CUBEMOS_SKEL_SDK%\bin` was added to the path variable and restart the PC.

Unity Wrapper – cubemos_engine.dll not found

Issue:
DllNotFoundException: cubemos_engine.dll

Possible Reason:
"%C_EM64T_REDIST12%redist\intel64\compiler" is not in the system path or the PC was not restarted after the installation of the intel redistributable.

Solution:

1. Reinstall the `intel_redist_x64.msi` from the `%CUBEMOS_SKEL_SDK%\dependencies\redistributables` folder.

2. Verify that the C_EM64T_REDIST12 variable was set, default location is `%ProgramFiles(x86)%\Common Files\Intel\Shared Libraries\`

3. Verify that "%C_EM64T_REDIST12%redist\intel64\compiler" was added to the path variable.
4. If the variable is missing or is not in the PATH run setup_sdk_env.bat script.

5. Restart the PC.

Linux: Activation with post_installation.sh fails if run with sudo

Issue:
The post_installation script shows an error message that cubemos environment variables cannot be found.

Possible Reason
The environment variables are not visible if the script is started with sudo rights.

Solution:
Remove the ~/.cubemos/ with all subfolders if it exists and run post_installation.sh without sudo.

Other possible but not recommended way is to use “sudo -E post_installation.sh” to preserve the existing environment variables. In this case all samples and Skeleton Tracking applications would need to be run with sudo too!

Any other error messages

Issue:
Error message is provided, and you cannot resolve the issue.

Possible Reason:
Unknown

Solution:
Have a look in the log directory which is defined by the environment variable $CUBEMOS_LOG_DIR.
On Windows the default location for this folder is
%LOCALAPPDATA%\Cubemos\SkeletonTracking\logs.
On linux it is ~/.cubemos/skeleton_tracking/logs.

In case you cannot fix the problem on your own, please send these logs together with a description of your problem to support@cubemos.com