Intel® RealSense™ SDK

Capturing Raw Streams Tutorial

With the Intel® RealSense™ SDK, you have access to robust, natural human-computer interaction (HCI) algorithms such as face tracking, finger tracking, gesture recognition, speech recognition and synthesis, fully textured 3D scanning and enhanced depth augmented reality.

With the SDK you can create Windows* desktop applications that offer innovative user experiences.

In this tutorial, you'll learn how to use the SDK to capture color and depth images from your input device. An application can render image samples within a loop and output the video data streams to a screen or output file.

By the end of this tutorial you'll be ready to start using the face tracking and other algorithm modules in the SDK.

Contents

Overview

The Intel RealSense SDK supports two types of modules: input/output modules and algorithm modules. This tutorial shows you how to implement I/O modules, and later tutorials show you how to implement algorithm modules.

This tutorial shows how to capture aligned color and depth samples, but it is also possible to capture them individually (unaligned). Capturing unaligned samples may be useful if you require a high frame rate for streaming depth data.

You can use either procedural calls or event callbacks to capture data, and code samples are provided for both (see Table 1). Using event callbacks is usually preferred when developing console applications; procedural calls are often used for GUI applications. This tutorial's code samples use procedural calls.

Executable files (.exe) are provided in the Release subfolder in the code sample directory.

Table 1: Code Samples

Code Sample	For more information, see:
Capturing aligned or unaligned color and depth streams using procedural calls File: main_raw_color_depth_streams.cpp	This Tutorial. Also see Color and Depth Samples using the SenseManager sections in the SDK Reference Manual.
Capturing aligned or unaligned color and depth streams using event callbacks	Color and Depth Samples using the SenseManager Events sections in the SDK Reference Manual.

Creating a Session

Create instances of **UtilRender**; utility class to render streams.

The SDK core is represented by two interfaces:

- **PXCSession:** manages all of the modules of the SDK
- **PXCSenseManager:** organizes a pipeline by starting, stopping, and pausing the operations of its various modalities.

The first step when creating an application that uses the Intel RealSense SDK is to create a session. A session can be created explicitly by creating an instance of **PXCSession**. Each session maintains its own pipeline that contains the I/O and algorithm modules.

Another way of creating a session is by creating an instance of the **PXCSenseManager** using **CreateInstance**. The PXCSenseManager implicitly creates a session internally.

```
 #include <pxcsensemanager.h>
//SDK provided utility class used for rendering
#include "util_render.h"

{
        // initialize the UtilRender instances
        UtilRender *renderColor  = new UtilRender(L"COLOR STREAM");
        UtilRender *renderDepth  = new UtilRender(L"DEPTH STREAM");

        // create the PXCSenseManager
        PXCSenseManager *psm=0;
        psm = PXCSenseManager::CreateInstance();
        if (!psm) {
                wprintf_s(L"Unable to create the PXCSenseManager\n");
                return 1;
         }
}
```

Initializing the Pipeline

1. Add the color and depth streams to the pipeline using the **EnableStream** function as separate calls.
   a. Specify the stream types **STREAM_TYPE_COLOR** and **STREAM_TYPE_DEPTH** from **PXCCapture**.
   b. Specify the resolution (**width** and **height**) of the streams.
2. Initialize the pipeline with the **Init** function so that the requested stream samples can be processed.

- Color stream resolution can support up to 1920x1080 pixels; you can configure various frame rates as well.
- The SDK also gives you access to left, right, and IR camera feeds.

Note: If a stream is not available with the specified settings, the camera will not stream—to indicate the settings are incorrect. When the settings are correct, **Init** function will return **PXC_STATUS_NO_ERROR** status.

```
// select the color stream of size 640x480 and depth stream of size 640x480
psm->EnableStream(PXCCapture::STREAM_TYPE_COLOR, 640, 480);
psm->EnableStream(PXCCapture::STREAM_TYPE_DEPTH, 640, 480);

// initialize the PXCSenseManager
if( psm -> Init() != PXC_STATUS_NO_ERROR) {
        wprintf_s(L"Unable to Init the PXCSenseManager\n");
        return 2;
}
```

Capturing Color and Depth Streams

1. Create a loop to process streams.
2. In every iteration of the loop, first use the **AcquireFrame** function:
   a. TRUE (aligned) to wait for both color and depth samples to be ready in a given frame; else
   b. FALSE (unaligned) to return whenever either of the two samples are ready.
3. Retrieve an instance of sample from **PXCCapture::Sample** through the **QuerySample** function.
4. Retrieve individual color and depth images from the sample.
5. Render the color and depth streams using the **UtilRender** class, a utility class provided in the SDK for rendering.
6. Release the frame for reading the next samples (color + depth) through the **ReleaseFrame** function.

```
PXCImage *colorIm, *depthIm;

for (int i=0; i<MAX_FRAMES; i++) {

        // This function blocks until all streams are ready (depth and color)
        // if false streams will be Unaligned
        if (psm->AcquireFrame(true)<PXC_STATUS_NO_ERROR)  break;

        // retrieve all available image samples
        PXCCapture::Sample *sample = psm->QuerySample();

        // retrieve the image or frame by type from the sample
        colorIm = sample->color;
        depthIm = sample->depth;

        // render the frame
        if (!renderColor->RenderFrame(colorIm)) break;
        if (!renderDepth->RenderFrame(depthIm)) break;

        // release or unlock the current frame to fetch the next frame
        psm->ReleaseFrame();

}
```

Cleaning Up the Pipeline

After your application is done capturing and rendering samples, you must “clean up”:

1. Delete all UtilRender instances using **delete**.
2. Close all the last opened streams and release any session and processing module instances using **Release()** on the **PXCSenseManager** instance.

```
// delete the UtilRender instance
delete renderColor;
delete  renderDepth;

// close the last opened streams and release any session and processing module instances
psm->Release();
```

Now you have all the information to configure, capture, render, and display raw color and depth data from input streams using your device.

Running the Code Samples

You can run these code samples two ways:

1. Build and run the **2_RawDataCapture** sample in Visual Studio*.
2. Run the executables found in the “Release” subfolder of the tutorial code sample directory.

Figures 1 and 2 show the output when capturing and rendering aligned color and depth streams from the 2_RawDataCapture code sample.

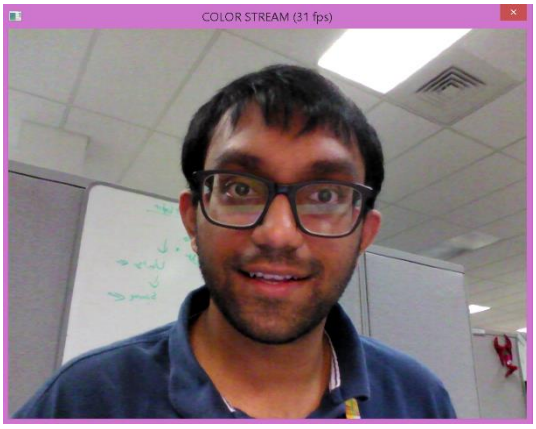

Figure 1. Rendered Color Stream

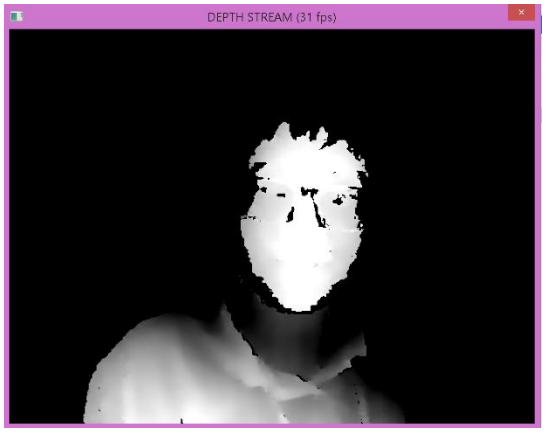

Figure 2. Rendered Depth Stream

To learn more

- The SDK Reference Manual is your complete reference guide and contains API definitions, advanced programming techniques, frameworks, and other need-to-know topics.

- You can use PXC[M]CaptureManager to query a PXC[M]Capture device in order to manipulate camera behavior such as DepthConfidenceThreshold, IVCAMAccuracy, MirrorMode, , IVCAMMotionRangeTradeOff, etc. Refer to the Interface and Function Reference : Essential section in the SDK Reference Manual.

- You can extract z-depth data from the depth samples using **data.planes[0]**. Refer to the Access Image and Audio Data section in the SDK Reference Manual.