Intel® RealSense™ SDK 2014

Background Segmentation 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.

Using 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 a segmented image to remove or replace portions of the image behind a user's head and shoulders (background) using the Intel RealSense camera.

Contents

- Overview
- Creating a Session
- Initializing the Pipeline
- Capturing Background Segmented Stream
- Cleaning Up the Pipeline
- Running the Code Samples
- To learn more

Overview

The Background Segmentation module generates a segmented image per frame that can be used to remove or replace portions of the image behind the user's head and shoulders—in other words, some or all of the background.

The output image is in 32-bit BGRA (PIXEL_FORMAT_RGB32) format and contains a copy of the input color image data and a synthesized alpha channel (mask).

Pixels corresponding to the background have an alpha channel value less than 128, and the pixels corresponding to the user's head and torso contain an alpha value greater than or equal to 128.

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 segmented stream using procedural calls File: main_background_segmentation.cpp	This Tutorial. Also see User Segmentation Using the SenseManager Procedural Functions section in the SDK Reference Manual.
Capturing segmented stream using event callbacks	User Segmentation Using the SenseManager Callback Functions section in the SDK Reference Manual.

Creating a Session

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 "pxc3dseg.h"
#include "pxcsensemanager.h"
#include "util_render.h"
{
        // initialize the Util Renderer
        UtilRender *renderer = new UtilRender(L"BACKGROUND REMOVED 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. Enable the User Segmentation video module using **Enable3Dseg**.
2. Retrieve an instance of the segmentation module using **Query3DSeg**.
3. Finally, initialize the pipeline with the **init** function.

```
// Enable the User Segmentation video module
 sts = psm->Enable3DSeg();
 if (sts<PXC_STATUS_NO_ERROR){
		wprintf_s(L"Failed to Enable 3D Segmentation \n");
		return 2;
}
// Retrieve an 3D segmentation module instance
PXC3DSeg* pSeg = psm->Query3DSeg();
if (!pSeg) {
		wprintf_s(L"Unable to retrieve an 3Dseg instance\n");
		return 3;
}
// initialize the PXCSenseManager
if(psm->Init() != PXC_STATUS_NO_ERROR) return 4;
```

Capturing Background Segmented Stream

1. Create a loop to process the stream.
2. In every iteration of the loop, first use the **AcquireFrame** function:
    a. TRUE (aligned) to wait for all modules (processing and I/O) to be ready in a given frame;
    b. FALSE (unaligned) whenever any of the processing modules signal
4. Retrieve a segmented image using **AcquireSegmentedImage** on the segmentation module instance.
5. Process the segmented image, which is a raw color image with an alpha channel (mask) – 32-bit BGRA.
6. Render the processed segmented image using the **UtilRender** class, a utility class provided in the SDK for rendering.
7. Release the segmented image using **Release** on the segmented image instance.
8. Release the frame for acquiring a segmented image in the next frame using **ReleaseFrame**.

```
PXCImage* segmented_image;
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 an color Image instance with an alpha channel (mask)
        segmented_image = pSeg->AcquireSegmentedImage();
         if (segmented_image)
         {
                //lighten the background with alpha channel
                 LightenBackground(segmented_image);
                 // render the frame
                if (!renderer->RenderFrame(segmented_image)) break;
                // release the image
                segmented_image->Release();
         }

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

For an example of processing image data, please refer to the implementation of the **LightenBackground** function provided in the code sample that explains how to iterate over the pixels in the image and manipulate the pixel data.

Cleaning Up the Pipeline

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

1. Delete the UtilRender instance using **delete**.
2. Release any session and processing module instances using **Release()** on the **PXCSenseManager** instance.

```
// delete the UtilRender instance
delete renderer;

// 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 user segmented image data using the Intel RealSense camera.

Running the Code Samples

You can run this code sample two ways:

1. Build and run the **7_Background_Segmentation** sample in Visual Studio*.
2. Run the executables found in the "Release" subfolder of the code sample directory.

Figure 1 shows the output when capturing and rendering a user segmented image from the 7_Background_Removal code sample.

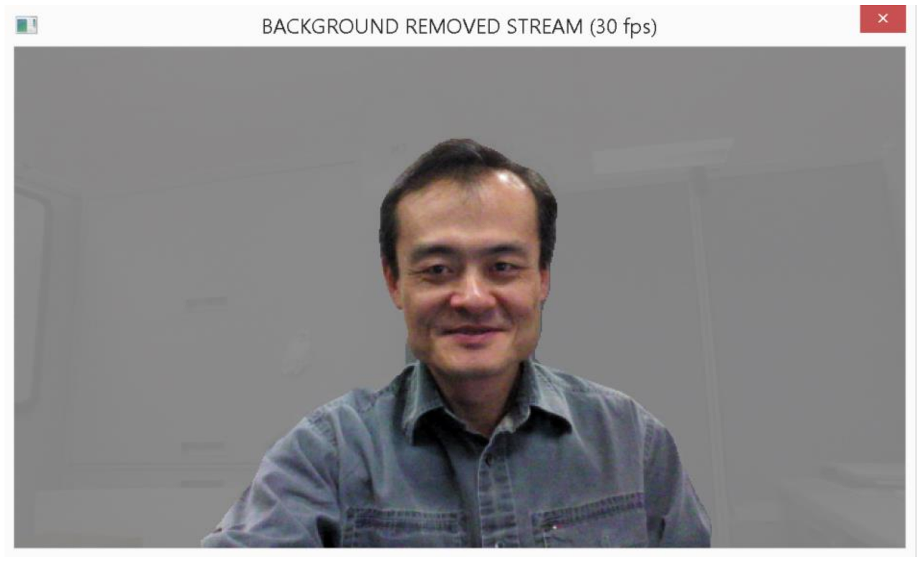

Figure 1. Processed User Segmented Stream

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 also tweak the device settings to get the best background segmentation results by changing certain device properties:

```
PXCCapture::Device* device = psm->QueryCaptureManager()->QueryDevice();
if (device)
{
   device->SetDepthConfidenceThreshold(0);
   device->SetIVCAMLaserPower(16);
   device->SetIVCAMAccuracy(PXCCapture::Device::IVCAM_ACCURACY_COARSE);
   device->SetIVCAMMotionRangeTradeOff(21);
   device->SetIVCAMFilterOption(6);
}
```