Intel® RealSense™ SDK
Sample Applications

Table of Contents

1 Legal Information

By using this document, in addition to any agreements you have with Intel, you accept the terms set forth below.

You may not use or facilitate the use of this document in connection with any infringement or other legal analysis concerning Intel products described herein. You agree to grant Intel a non-exclusive, royalty-free license to any patent claim thereafter drafted which includes subject matter disclosed herein.

Information in this document is provided in connection with Intel products. No license, express or implied, by estoppel or otherwise, to any intellectual property rights is granted by this document. Except as provided in Intel's Terms and Conditions of Sale for such products, Intel assumes no liability whatsoever and Intel disclaims any express or implied warranty, relating to sale and/or use of Intel products including liability or warranties relating to fitness for a particular purpose, merchantability, or infringement of any patent, copyright or other intellectual property right.

Unless otherwise agreed in writing by Intel, the Intel products are not designed nor intended for any application in which the failure of the Intel product could create a situation where personal injury or death may occur.

Intel may make changes to specifications and product descriptions at any time, without notice. Designers must not rely on the absence or characteristics of any features or instructions marked "Reserved" or "Undefined." Intel reserves these for future definition and shall have no responsibility whatsoever for conflicts or incompatibilities arising from future changes to them. The information here is subject to change without notice. Do not finalize a design with this information.

The products described in this document may contain design defects or errors known as errata which may cause the product to deviate from published specifications. Current characterized errata are available on request.

Contact your local Intel sales office or your distributor to obtain the latest specifications and before placing your product order.

Copies of documents which have an order number and are referenced in this document, or other Intel literature, may be obtained by calling 1-800-548-4725, or by visiting Intel's web site http://www.intel.com.

Any software source code reprinted in this document is furnished under a software license and may only be used or copied in accordance with the terms of that license.

Intel, the Intel logo, Intel RealSense, Intel Core and Iris are trademarks or registered trademarks of Intel Corporation or its subsidiaries in the United States and other countries.

2 Introducing the SDK

The Intel® RealSense™ SDK is a library of pattern detection and recognition algorithm implementations exposed through standardized interfaces. The library aims at lowering barriers to using these algorithms and shifting the application developers' focus from coding the algorithm details to innovating on the usage of these algorithms for next generation human computer experience.

Optimization Notice

Intel's compilers may or may not optimize to the same degree for non-Intel microprocessors for optimizations that are not unique to Intel microprocessors. These optimizations include SSE2, SSE3, and SSSE3 instruction sets and other optimizations. Intel does not guarantee the availability, functionality, or effectiveness of any optimization on microprocessors not manufactured by Intel. Microprocessor- dependent optimizations in this product are intended for use with Intel microprocessors. Certain optimizations not specific to Intel microarchitecture are reserved for Intel microprocessors. Please refer to the applicable product User and Reference Guides for more information regarding the specific instruction sets covered by this notice.

Notice revision #20110804

This document describes the SDK samples and demo applications. SDK samples focus on demonstrating SDK API functions and SDK demo applications showcasing usages.

Notational Conventions

This SDK document uses the Calibri typeface for normal prose.

With the exception of section headings, captions and the table of contents, all code-related items appear in the Courier New typeface (`pxcStatus`).

Hyperlinks appear underlined in blue, such as pxcStatus.

 This is a note that provides additional information to aid your understanding of the procedure or concept.

 This is a tip that provides alternate methods or shortcuts.

 This is a result statement which indicates what you can expect to see or happen after performing a step.

3 SDK C++/C# Samples

The section describes C++/C# samples that shows the SDK interface usages.

Samples with the .cs suffix are C# samples, the rest are C++ samples.

3.1 Building Samples in Microsoft* Visual Studio* IDE

The sample files are available in the following locations:

- **C++**: `$(RSSDK_DIR)/sample`
- **C#**: `$(RSSDK_DIR)/framework/CSharp`

To build the sample with Microsoft Visual Studio 2010-2013, complete the following steps:

1. Copy the sample to a non-privileged location. It is not recommended to build the sample directly under **`%PROGRAMFILES%`**.
2. Click the sample solution file **`<sample-name>_vs20xx.sln`**. For example, the `DF_CameraViewer` sample's solution file for Microsoft Visual Studio 2012 is **`DF_CameraViewer_vs2012.sln`**.
3. Click **Build>Rebuild** to build the sample.

   The prebuilt sample binaries are under **`bin/$(PlatformName)`**.

3.2 Common Samples

The samples work for any Intel® RealSense™ camera, either the front-facing camera, model F200, or the rear-facing camera, model R200.

The sample names are prefixed with "`DF`", which stands for "`Dual Facing`".

3.2.1 Sample: DF_3DScan

The `DF_3DScan.exe` sample is a minimal C++ console application that demonstrates the capabilities of the 3D capturing and scanning. This sample can be used to generate 3D mesh data.

Launch & Build

You can launch the prebuilt sample directly from the `$(RSSDK_DIR)/bin/$(Platform)` folder of the SDK installation, or compile and execute within Microsoft Visual Studio. The project and source files are located under `$(RSSDK_DIR)/sample/DF_3DScan`.

Usage

Attach the camera, and start the sample executable.

During the sample startup sequence, optionally press and hold the `Ctrl` key to select a profile in the profile selection menu similar to what is shown in Figure 1. Currently, the module supports only a single profile.

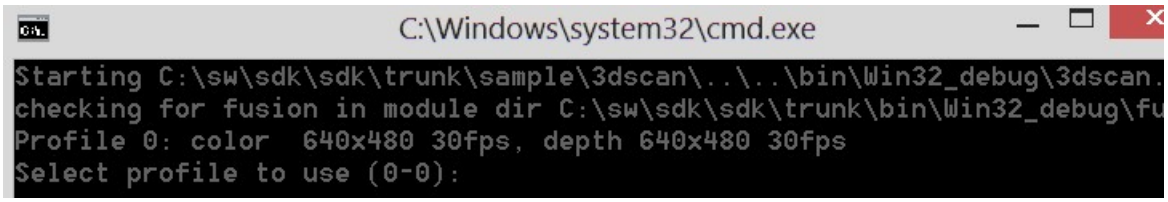


Figure 1: 3dscan profile selection menu.

Once the user selects the profile, the selected profile details are shown in stdout similar to what is shown in Figure 2.

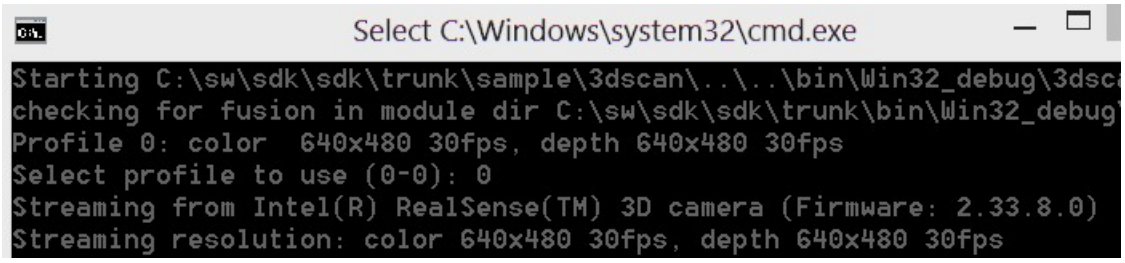


Figure 2: 3dscan showing the selected profile.

If the `Ctrl` key is not held during the startup sequence, the profile selection menu is not shown, and the first profile is used. In this case, you should see something similar to what is shown in Figure 3.

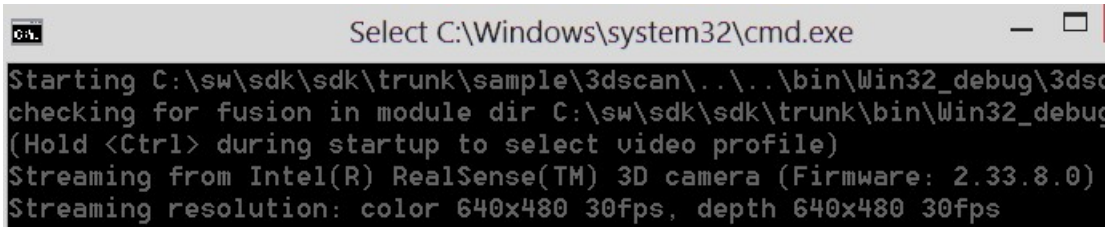


Figure 3: 3dscan starting with first profile.

After the initialization is complete, the preview window is created. An example of the preview window is shown in Figure 4.

Figure 4: 3dscan preview window when Mode is set to TARGETING. The preview window shows the extent of the scanning volume.

When the preview window is in focus, the user starts the scanning process by pressing the space bar to switching from the `TARGETING` to `SCANNING` mode. The sample is implemented to toggle between these modes using the space bar. The preview window will change to show the `SCANNING` preview similar to what is shown in Figure 5. The rendering style and size of the preview window will change to match the mode.

Figure 5: 3dscan preview when Mode is set to SCANNING. The preview window shows the accumulated data within the scanning volume.

Generating a 3D Mesh data set from the scanning volume

The scanning volume is converted into a mesh format when the window is closed. The mesh file (out.obj) is created in the current working directory. When debugging in Microsoft Visual Studio, this will be the location of the project file (i.e. `sample\3dscan\`). The OBJ output format is one of the supported standard mesh formats. These files can then be loaded into standard tools or importers for use.

Limitations

- This sample does not support simultaneous use of multiple cameras.
- This sample does not support camera detach (or attach) while it is running.
- If the camera is not attached before the sample is started, it will exit with an error code.
- If the camera is disconnected while the sample is running, it will exit with an error code.

3.2.2 Sample: DF_CameraViewer

The `DF_CameraViewer` sample is a simple C++ console application to stream and render color and/or depth streams.

The sample can be used for quickly testing certain SDK interfaces.

```
DF_CameraViewer [-load <module name>] [-sdname <device name>]
[-csize <resolution>] [-dsize <resolution>] [-file <Output Filename>] [-
listio] [-record] [-help]
```

-load Explicitly load the specified module.

-csize Specify the color stream resolution and frame rate. For example, use `640x480` to specify the resolution, or `640x480x30` to specify the resolution and the frame rate.

-dsize Specify the depth stream resolution and frame rate. For example, use `640x480` to specify the resolution, or `640x480x30` to specify the resolution and the frame rate.

-sdname Specify an input device name.

-nframes Specify the maximum number of frames to record.

-listio List the I/O devices.

-record Enable file recording. Use the -file option to specify the recording file name.

-file Specify the recording or playback file name. If -record is not specified, the sample plays back the specified file. Otherwise, the sample records the camera data to the specified file.


The SDK installation directory is privileged and not writable. For file recording, please specify a file name in a writable directory.

-help Print out the help message.

Example:

```
DF_CameraViewer -nframes 200
```


3.2.3 Sample: DF_FaceTracking[.cs]

The `DF_FaceTracking` and `DF_FaceTracking.cs` samples show how to use the SDK face detection and landmark detection interfaces.

The main window is shown as in Figure 6. From the menu, you can choose the following items:

- `Device`: Select from a list of I/O devices to be paired with the face module.
- `Module`: Select from a list of face detection and landmark detection modules.
- `Profile`: Select the algorithm configuration: 2D/3D tracking or detection.
- `Mode`: Select whether to do live streaming, recording or playback. If the playback or recording mode is selected, the sample will prompt for the playback or recording file name.

From the side buttons, you can choose the following options:

- `Scale`: Scale the image to the size of the display window, or the actual size.
- `Mirror`: Flip the image horizontally to show the camera view or the user view.
- `Detection`: Display a rectangle around any recognized faces.
- `Landmarks`: Display dots on recognized landmark positions.
- `Pose`: Display the pose information on the right of the screen.
- `Expressions`: Display the facial expression scores on the screen.
- `Recognition`: Display the face identification information on the screen.
- `Start`: Start the pipeline for face detection and landmark detection.
- `Stop`: Stop the detection.
- `Register`: Register the user for face recognition.
- `Unregister`: Unregister the user for face recognition.

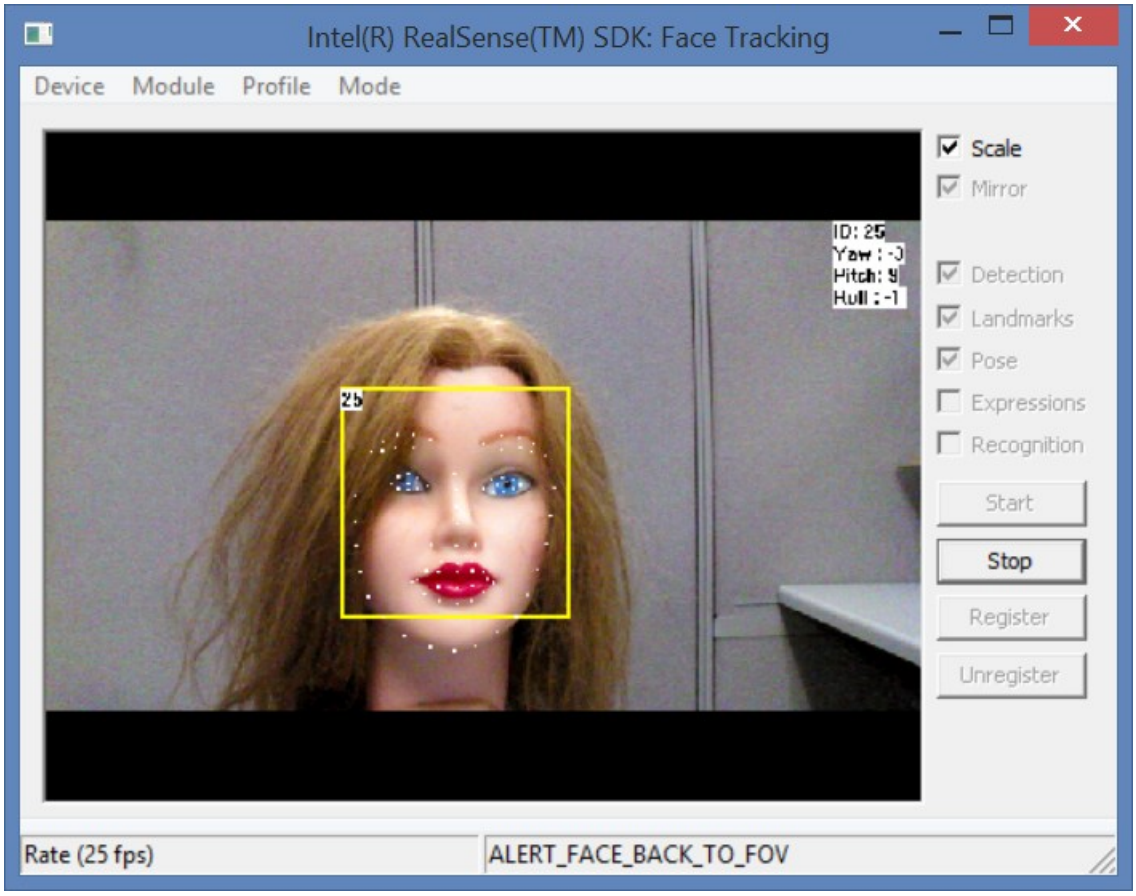


Figure 6: Face Tracking Main Window

3.2.4 Sample: DF_Projection

The `DF_Projection` sample demonstrates how to use the SDK `PXC[M]Projection` functions. The projection functions map or project coordinates among the color image coordinate system, the depth image coordinate system, and the camera coordinate system.

You can run the sample through pre-built binaries under `$(RSSDK_DIR)/bin/$(Platform)`, or build from source code, under `$(RSSDK_DIR)/sample/DF_Projection`.

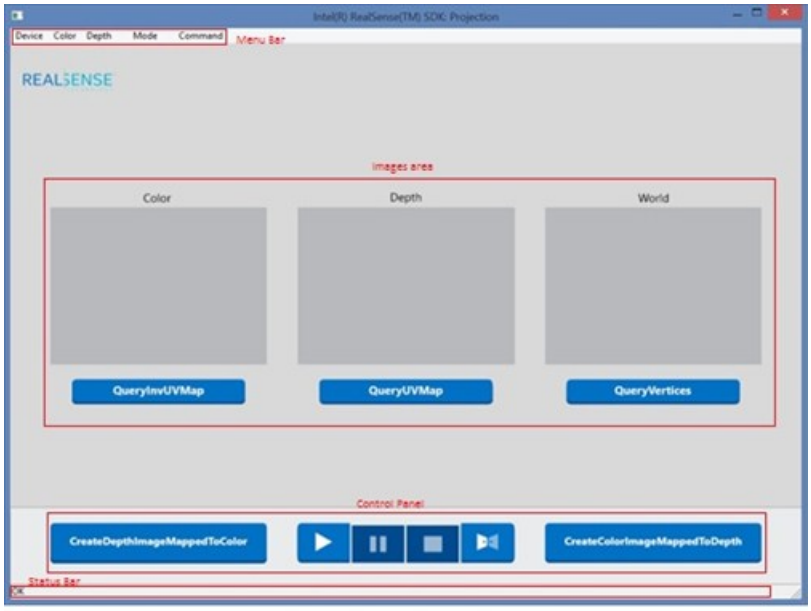


Figure 7: The Sample Main Window

The main window is shown in Figure 7. It contains the following components:

- The **`Menu`** bar contains the following tabs:
    - **Device** – Select the imaging device.
    - **`Color`**, **`Depth`** – Select the color/depth stream resolutions.
    - **`Mode`** – Select the live (default) or playback mode.
    - **`Command`** – Start or stop streaming.
- The **`Image Panels`** visualize the color, depth and vertices data as similar to Figure 8.

    Click on the `QueryInvUVMap` button to show the inverse UV map data, if not shown initially. The inverse UV map provides coordinates mapping from the color image to the depth image.

Click on the `QueryUVMap` button to show the UV map data, if not shown initially. The UV map provides coordinates mapping from the depth image to the color image.

Click on the `QueryVertices` button to show the vertices data, if not shown initially. The vertices data is in the 3D space. For rendering simplicity, the sample projects it to a 2D plane for visualization. The sample applies some lighting effect when you move your mouse to give a pseudo 3D effect.

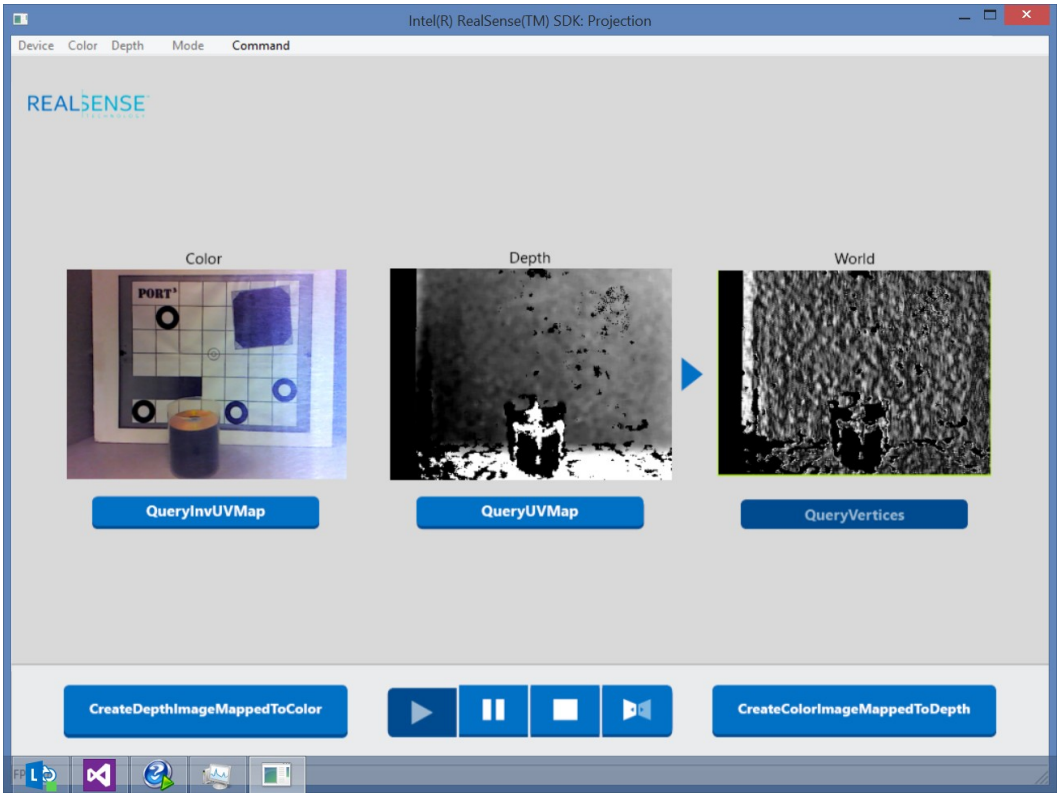


Figure 8: The Color/Depth/Vertices Visualization

You can draw on any of the three panels: color, depth and vertices. The sample maps the coordinates to the other panels and draws on them accordingly, with the mapping/projection functions marked for such operation. See Figure 9 - Figure 11.

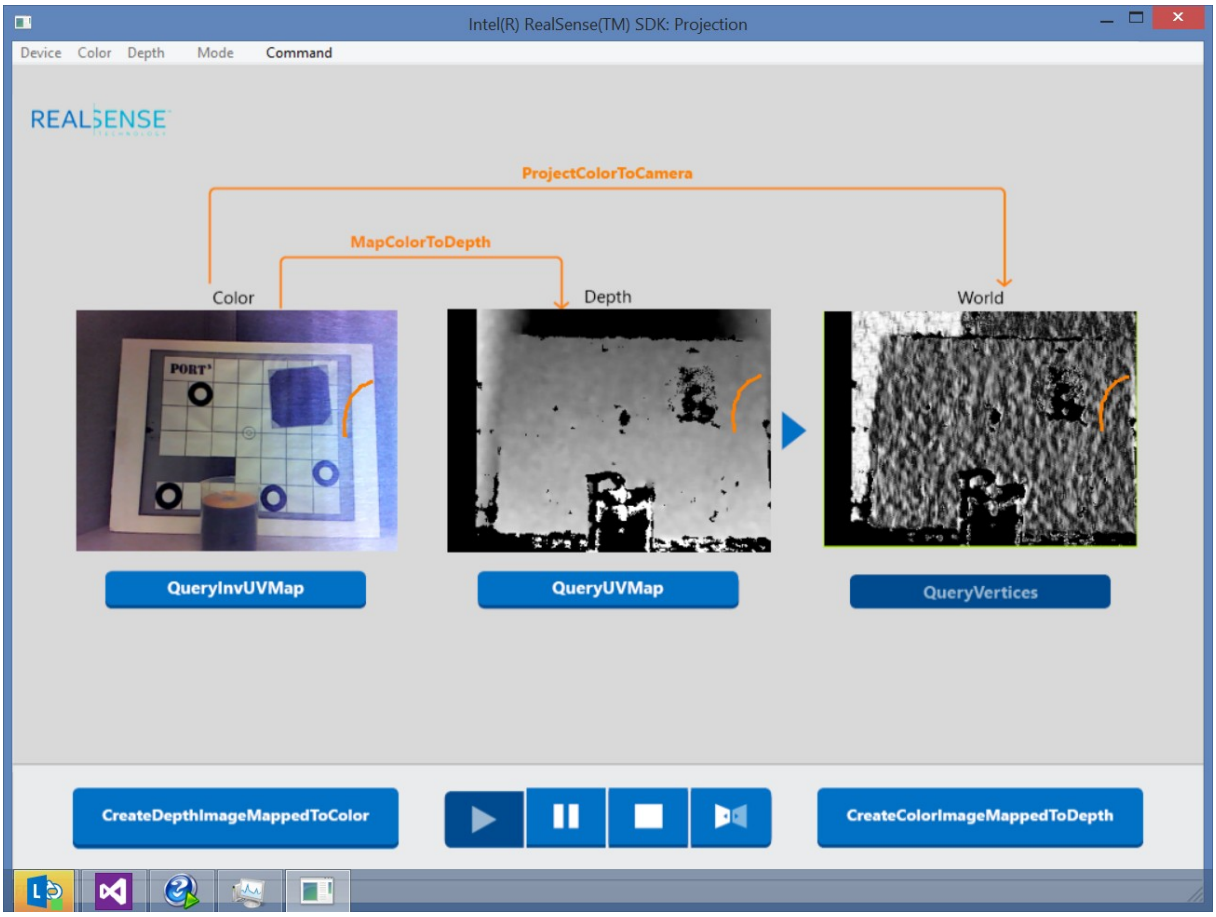


Figure 9: Map/Project Color Coordinates to Depth/Camera Coordinates

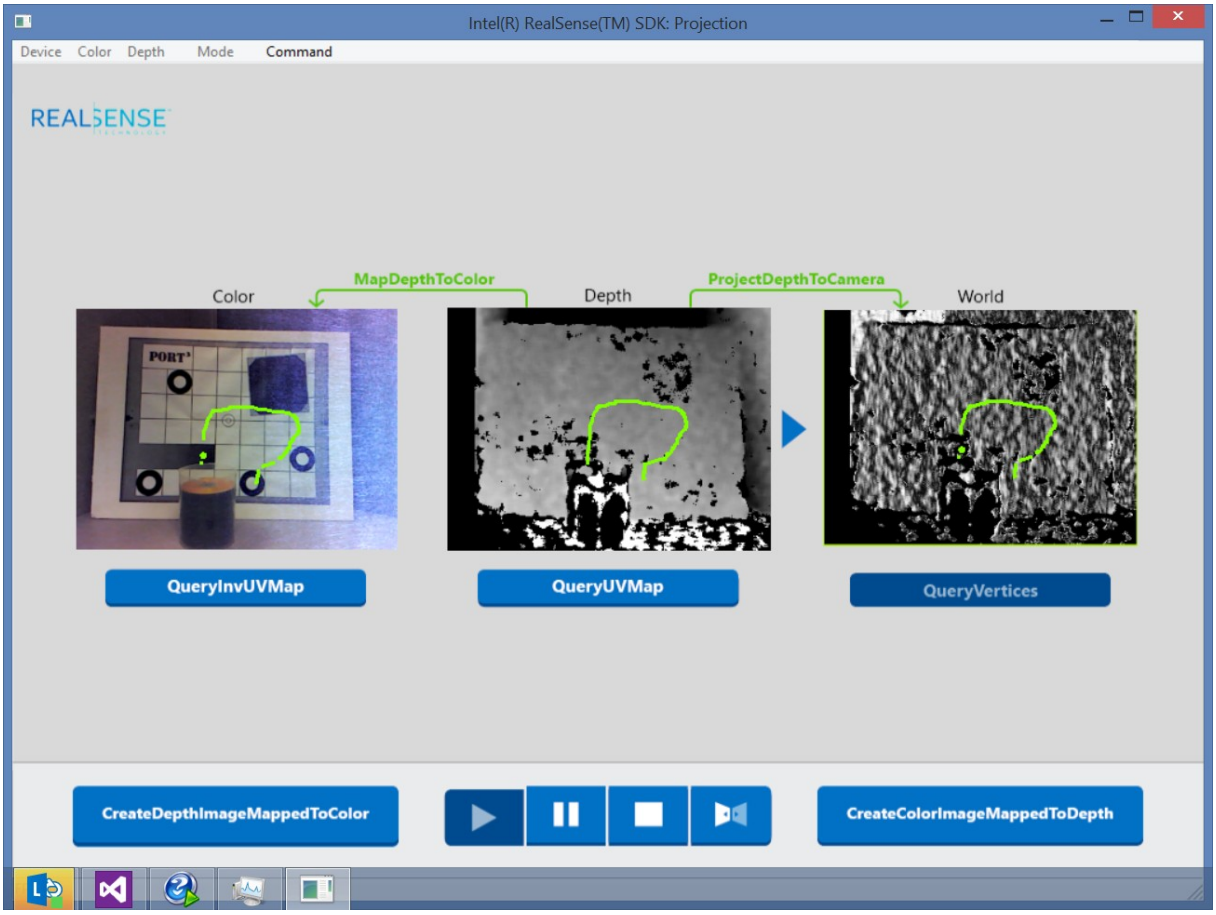


Figure 10: Map Depth Coordinates to Color and Camera Coordinates

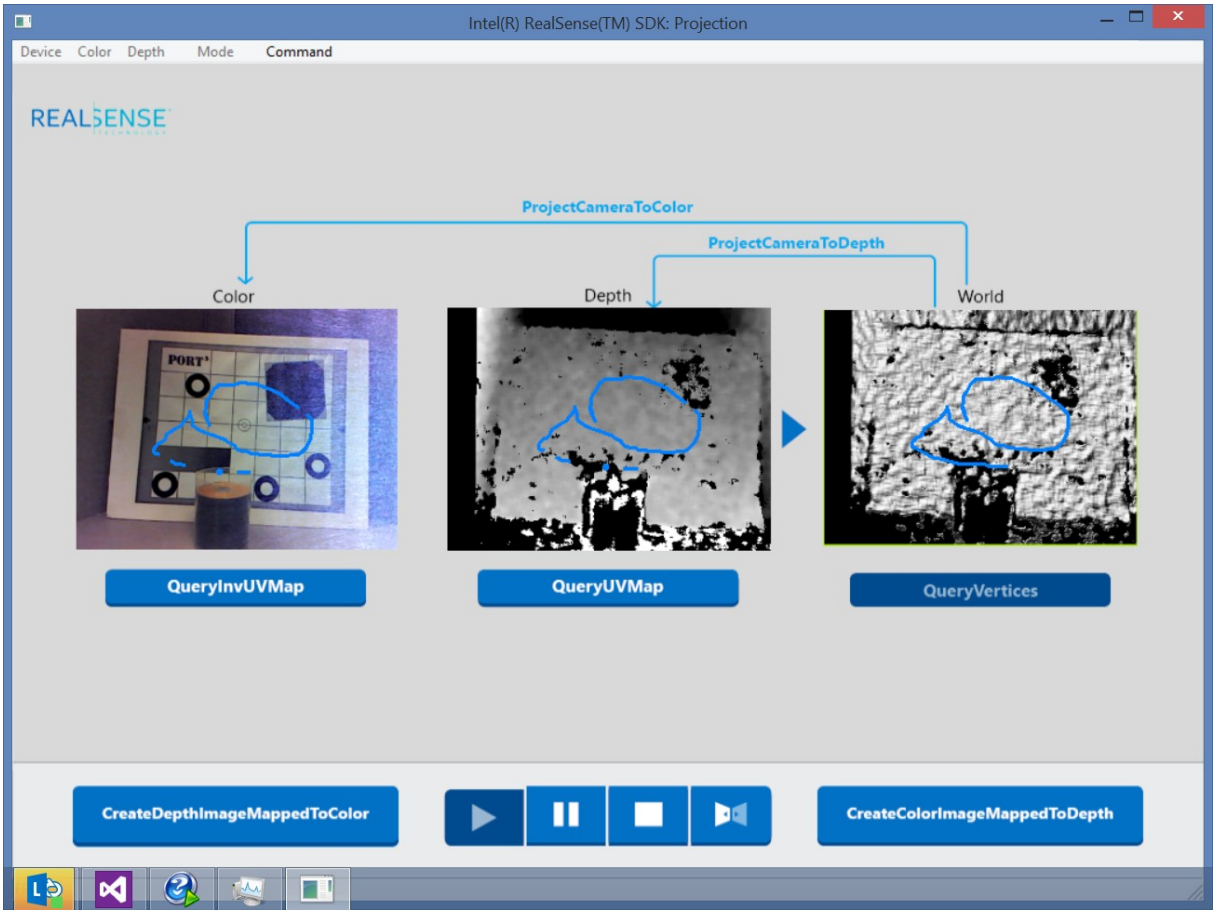


Figure 11: Project Camera Coordinates to Color and Depth Image Coordinates.

- The **`Control Panel`** contains the following buttons:

  - `CreateDepthImageMappedToColor` and `CreateColorImageMappedToDepth`: Create aligned images to the color and depth resolutions, respectively, as illustrated in Figure 12.

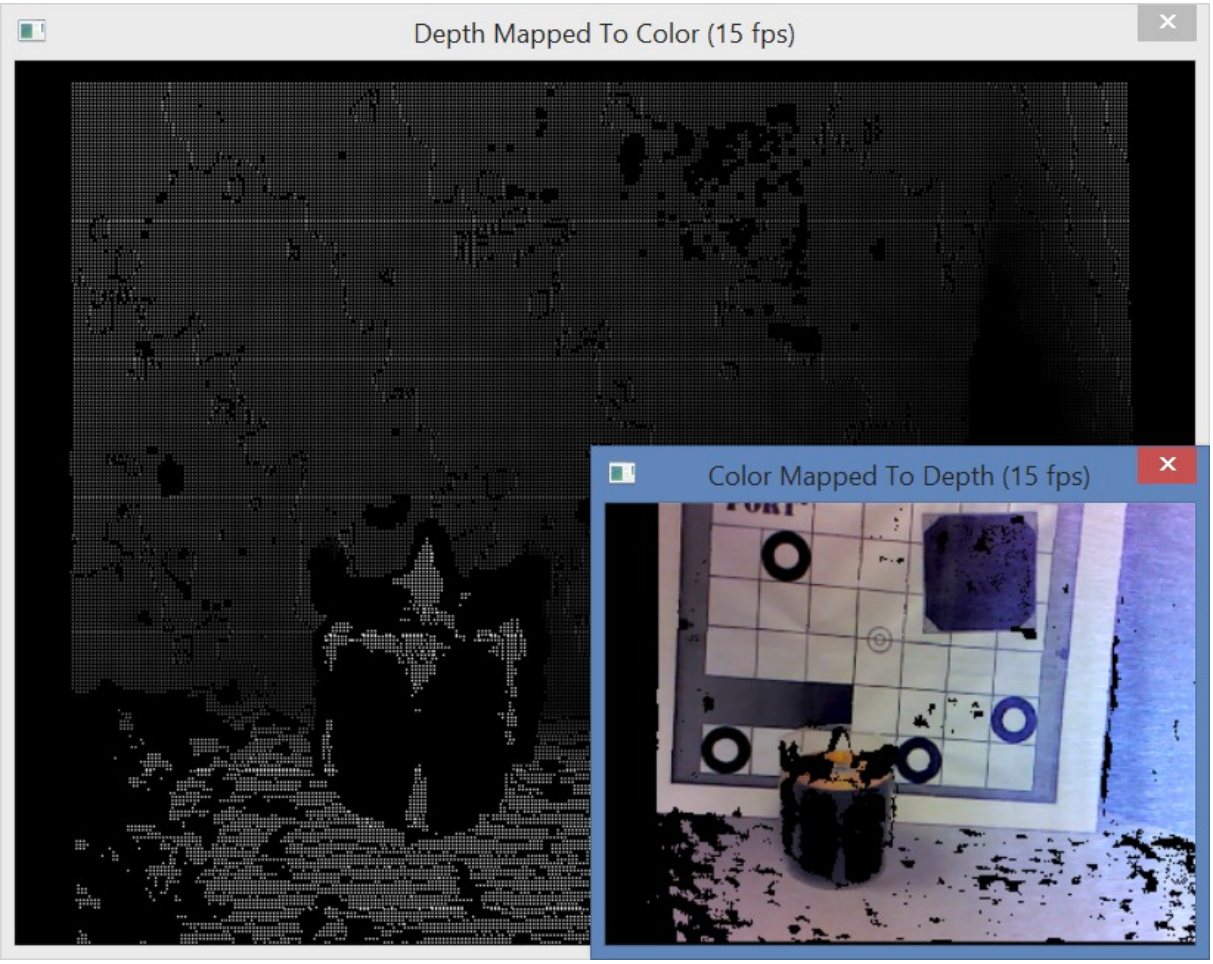


Figure 12: Create Aligned Color/Depth Images

- **`Play`** the color stream (and the depth stream.)
- **`Pause`** the streaming.
- **`Stop`** the streaming.
- **`Mirror`** the stream horizontally.

- The **`Status Bar`** provides the information during the sample operation such as selected resolution, clip and streaming status.

3.2.5 Sample: DF_RawStreams[.cs]

The `DF_RawStreams` and `DF_RawStreams.cs` samples show how to visualize raw depth and color streams and certain projection functions.

The main window is shown as in Figure 13. From the menu, you can choose the following items:

- `Device`: Select from a list of I/O devices for streaming.
- `Color`: Select the color resolution.
- `Depth`: Select the depth resolution.
- `IR`: Select the infrared stream resolution.
- `Mode`: Select whether to do live streaming, recording or playback. If the playback or recording mode is selected, the sample will prompt for the playback or recording file name.
- `C/D Sync`: Select whether to use synchronous or asynchronous color and depth streaming during visualization. The former synchronizes the color sample with the corresponding depth sample, while the latter visualizing them in their own frame rates.

From the side buttons, you can choose the following options:

- `Color`: Visualize the color stream in the main display. If picture-in-picture is enabled, the stream previously displayed in the main display goes to the picture-in-picture display.
- `Depth`: Visualize the depth stream in the main display. If picture-in-picture is enabled, the stream previously displayed in the main display goes to the picture-in-picture display.
- `IR`: Visualize the infrared stream in the main display. If picture-in-picture is enabled, the stream previously displayed in the main display goes to the picture-in-picture display.
- `UV MAP`: Map the depth coordinates to color coordinates using the `QueryUVMap` function of the `PXC[M]Projection` interface and plot the dots onto the color image.
- `Inv UV MAP`: Map the color coordinates to depth coordinates using the `QueryInvUVMap` function of the `PXC[M]Projection` interface, and plot the dots (depth value histogram) onto the color image.
- `Projection`: Map the depth to color coordinates using the `MapDepthToColor` function of the `PXC[M]Projection` interface and plot the dots on the color image.
- `Dots`: Adjust the dot size that uses to plot the mapping from depth to color coordinates. There are three choices: 1 dot, 5 dots and 9 dots.
- `Scale`: Scale the image to the size of the display window, or the actual size.

- `Mirror`: Flip the image horizontally to show the camera view or the user view.
- `PIP`: Open a picture-in-picture window to visualize the second stream in streaming. Multiple clicks can choose the window size and location.
- `Start`: Start streaming.
- `Stop`: Stop streaming.

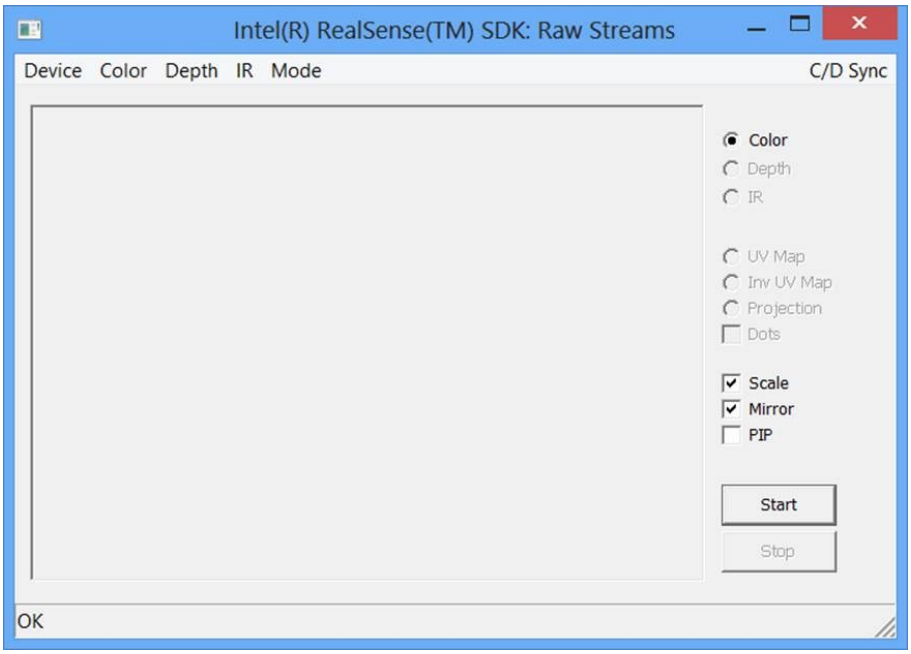


Figure 13: Raw Streams Main Window

3.3 F200 Samples

The samples work for the Intel® RealSense™ camera, model F200, a front facing camera.

The sample names are prefixed with "`FF`", which stands for "`Front Facing`".

3.3.1 Sample: FF_3DSeg[.cs]

The `FF_3DSeg` and `FF_3DSeg.cs` are two sample applications to demonstrate the user segmentation feature.

The C++ `FF_3DSeg` sample is a console application, which supports the following command line options:

`-load`	Explicitly load the specified module.
`-nframes`	Specify the maximum number of frames to record.
`-record`	Enable file recording. Use the -file option to specify the recording file name.
`-file`	Specify the recording or playback file name. If -record is not specified, the sample plays back the specified file. Otherwise, the sample records the camera data to the specified file. The SDK installation directory is privileged and not writable. For file recording, please specify a file name in a writable directory.
`-help`	Print out the help message.

Run the sample in a command window. The sample displays the segmented image in a pop up window, as similar to Figure 14:

Figure 14: User Segmentation Window

The `FF_3DSeg.cs` sample is a C# GUI application.

From the main menu, as illustrated in Figure 15, you can select the following items:

- `Device`: Select the input device.
- `Color/Depth`: Select the input stream resolutions.
- `Mode`: Select the operating mode as follows:
  - `Live`: Streaming from the live camera.
  - `Playback`: Streaming from a file.
  - `Record`: Streaming from the live camera and save the content to a file.

There are a set of controls on the right panel:

- `Segmented`: Display the segmented image in the display panel.
- `Color`: Display the color image in the display panel.
- `Depth`: Display the depth image in the display panel.
- `Scale`: scale the image to the display panel resolution
- `PIP`: show the last two image types selected in the picture in picture mode.

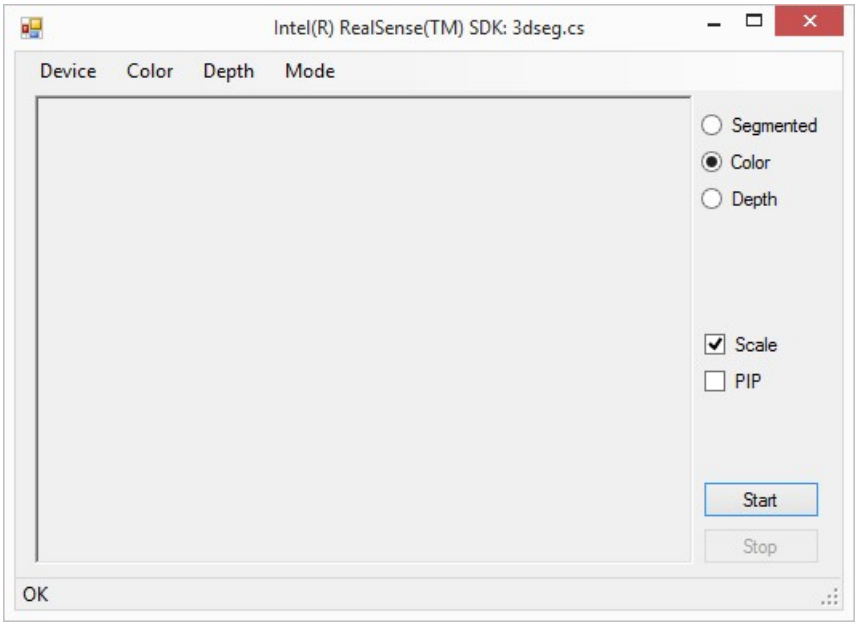


Figure 15: The 3dseg.cs Sample Main Window

Click the `Start` button to start streaming and the Stop button to stop streaming. The sample shows the segmented image (or any other image type selected) in the display panel, similar to Figure 16:

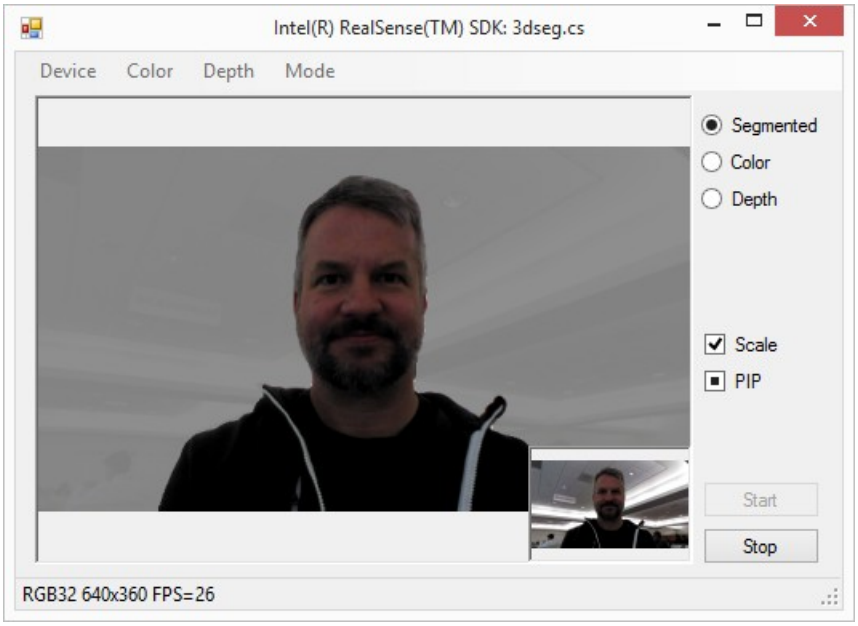


Figure 16: The 3dseg.cs Sample Streaming in Process

3.3.2 Sample: FF_BlobViewer[.cs]

The `FF_BlobViewer` and `FF_BlobViewer.cs` samples show how to use the SDK utility algorithms for blob and contour extraction.

The main window is shown as in Figure 17. From the menu, you can choose the following items:

- `Device`: Select from a list of I/O devices for streaming.
- `Mode`: Select whether to do live streaming or playback. If the playback mode is selected, the sample will prompt for the playback file name.

From the side buttons, you can choose the following options:

- `Show Blobs`: Display the detected blob images.
- `Show Contours`: Display the detected contour lines.
- `Blob Data`: You can fine-tune to the blob data detection parameters such as access order, smoothing parameters and max range.
- `Contour Data`: You can fine-tune the contour detector configuration parameters such as the smoothing parameter.
- `Start`: Start streaming.
- `Stop`: Stop streaming.

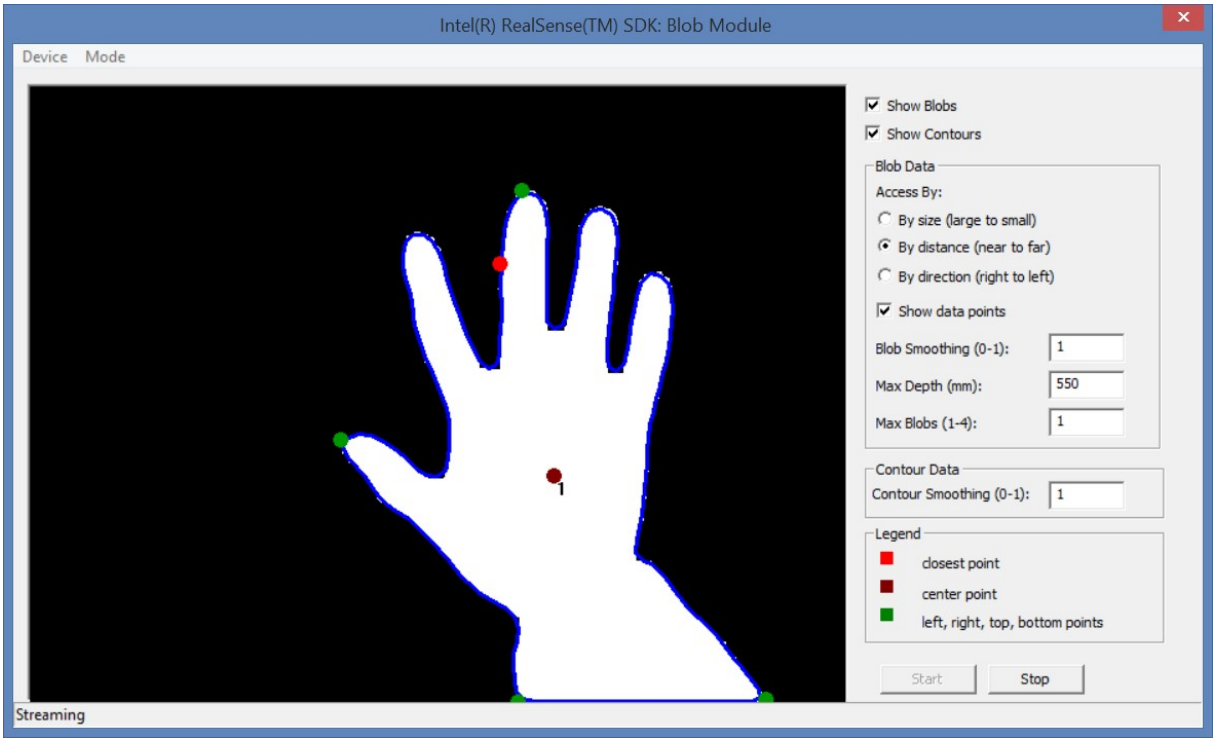


Figure 17: Blob Viewer Main Window

3.3.3 Sample: FF_EmotionViewer[.cs] [Preview]

Introduction

The `FF_EmotionViewer[.cs]` samples are GUI sample applications that demonstrate the features of the SDK emotion detection module. The samples track faces and display the face emotion states next to the tracked faces. The sample with the `.cs` suffix is written in C#. The sample without is written in C++.

Launch

You can launch the prebuilt samples directly from the `$(RSSDK_DIR)/bin/$(Platform)` folder of the SDK installation, or compile and execute within the Microsoft Visual Studio. The project and source files are located under `$(RSSDK_DIR)/sample/FF_EmotionViewer` (C++) or `$(RSSDK_DIR)/framework/CSharp/FF_EmotionViewer.cs` (C#).

Menu Options

From the menu, you can select the input device, the emotion module, and the operating mode (live streaming mode, playback mode, and recording mode), as shown in Figure 18:

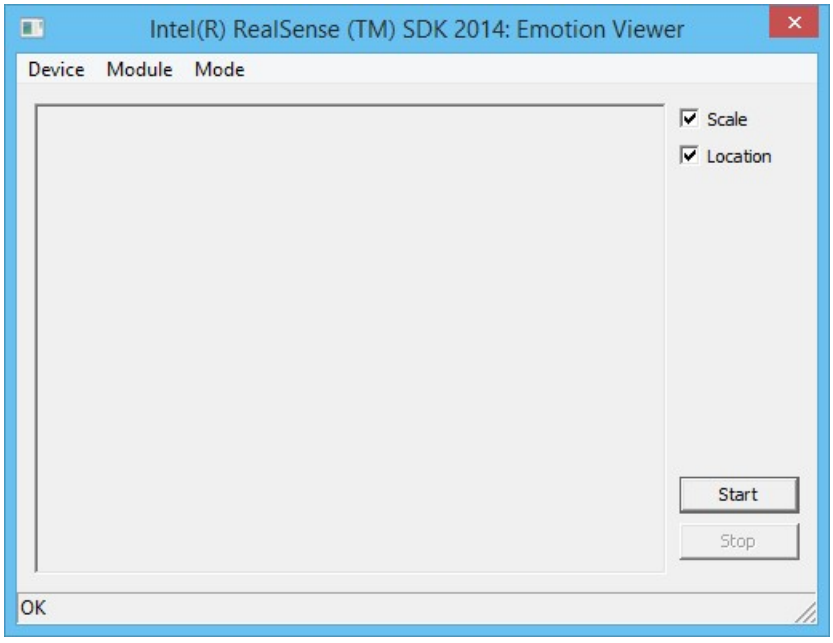


Figure 18: Sample emotion_viewer Menus

GUI Options

Click the `Start` button to load and activate the emotion viewer module.

Check the `Scale` option to scale the images to the size of the display panel, and the

`Location` option to toggle the box on and off around each detected face. This option does not turn off emotion captions.

Emotions

There are seven emotion labels (`Anger`, `Contempt`, `Disgust`, `Fear`, `Joy`, `Sadness`, and `Surprise`), and three sentiment labels (`Negative`, `Positive`, and `Neutral`). The detected emotion caption will display next to the face rectangle, as illustrated in Figure 19.

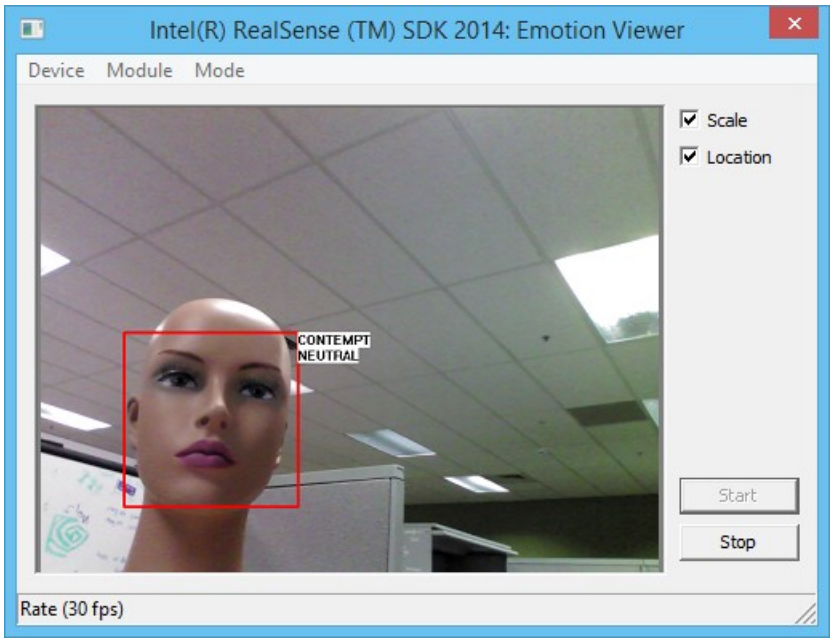


Figure 19: Sample emotion_viewer Menus

3.3.4 Sample: FF_HandsViewer[.cs]

Introduction

The `FF_HandsViewer[.cs]` samples are GUI applications that demonstrate the SDK hand tracking feature. The samples demonstrate, among other, the tracked hand skeleton (joints and bones), the segmentation mask (separating hand from background) and the gesture recognition mechanism. Two versions of this sample are available, one using the C++ API and the other on top of the C# API.

Launch

The samples can be launched directly from the `bin` folder of the SDK installation, or compiled and executed using Microsoft Visual Studio. The project and source files are located inside the `sample/FF_HandsViewer` or `framework/CSharp/FF_HandsViewer.cs` folders, respectively.

Menu Options

From the menu, the user can select the input device, the hand tracking module, and the operating mode (live streaming mode, playback mode, and recording mode), as shown in Figure 20:

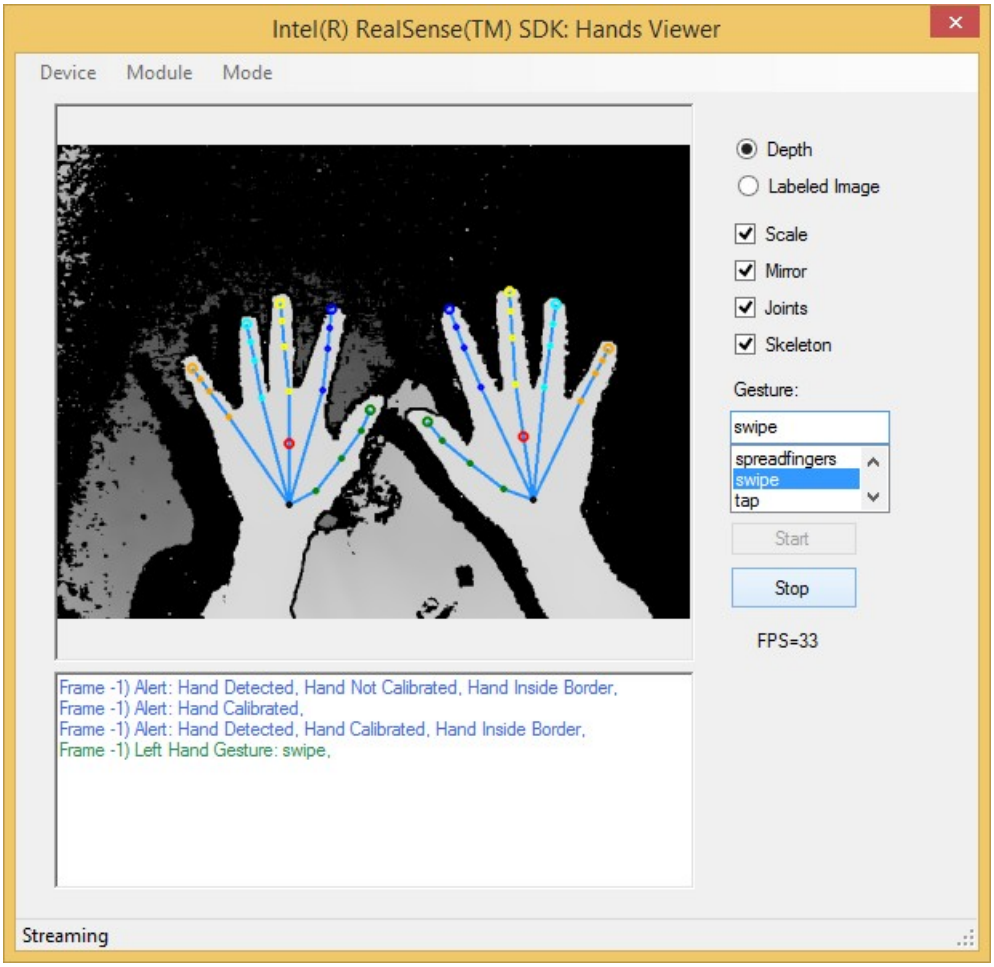


Figure 20: Sample hands_viewer

GUI Options

Click the `Start` button to load and activate the hand tracking module.

Check the option `Depth` to display the depth stream, as illustrated in Figure 20.

Check the option `Labeled Image` to display the labeled image.

Check the `Scale` option to scale the images to the size of the display panel, and the `Mirror` option to mirror the images (flip around Y axis).

Check the option `Joints` to visualize all 22 hand joints as colored circles.

Check the option `Skeleton` to visualize the hand skeleton.

Select an item in the `Gesture` list to recognize the specified gesture. If left empty, the sample does not recognize any gesture.

Whenever there is a fired gesture or alert, the sample logging window displays the gesture or alert details.

Finally, click the `Stop` button to terminate the tracking process.

3.3.5 Sample: FF_Hands3DViewer

Introduction

The `FF_Hands3DViewer` sample is an OpenGL* sample that visualizes hand tracking as an animated hand in the virtual world. As you move your hand and/or fingers, the virtual hand performs the same actions.

Launch

The samples can be launched directly from the `bin` folder of the SDK installation, or compiled and executed using Microsoft Visual Studio. The project and source files are located under the `sample/FF_Hands3DViewer` folder.

Sample Options

Present your hand in front of the camera. The virtual hand appears in the world view. You can move your hand and/or fingers. The virtual hand moves accordingly, as similar to Figure 21. You can use mouse wheels and buttons to rotate and zoom the virtual world.

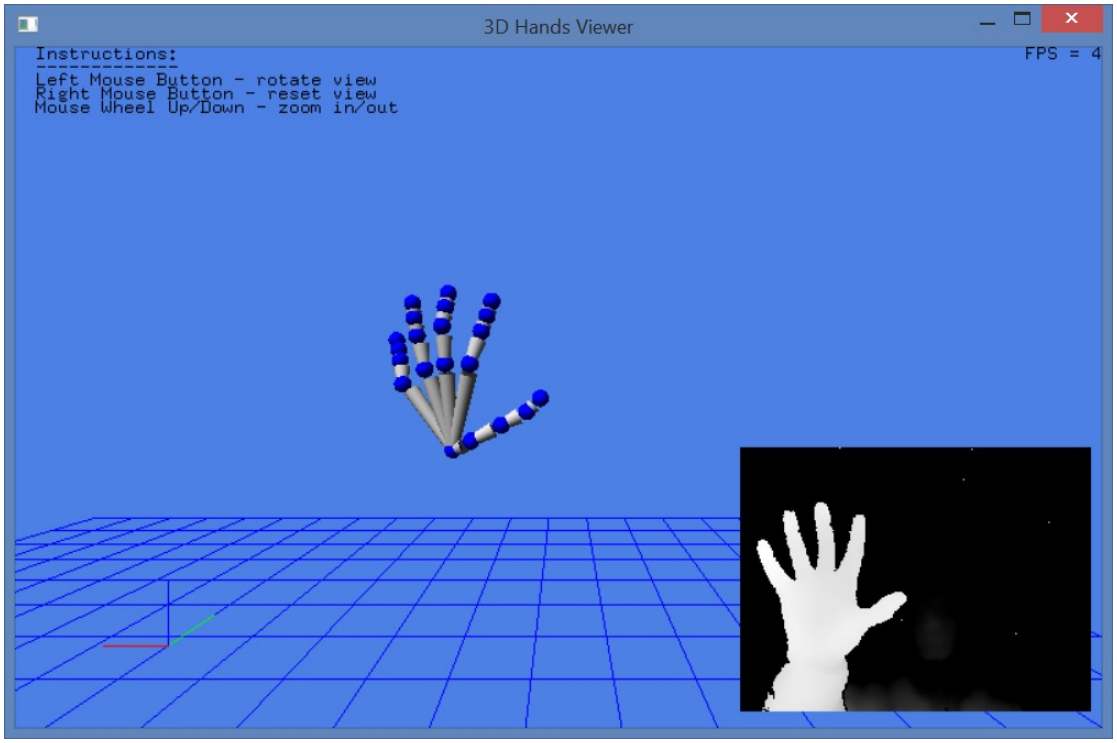


Figure 21: 3D Hands Viewer

3.3.6 Sample: FF_HandsConsole

Introduction

The `HandsConsole` sample is a C++ console application that demonstrates the SDK hand tracking feature. This application demonstrates, among other, the tracked hand skeleton (joints and bones), the segmentation mask (separating hand from background) and the gesture recognition mechanism.

Launch

The applications can be launched directly from the `bin` folder of the SDK installation, or compiled and executed using Microsoft Visual Studio. The project and source files are located inside the `sample/FF_HandsConsole` folder. Launch the sample executable under a command window. The sample output is similar to Figure 22.

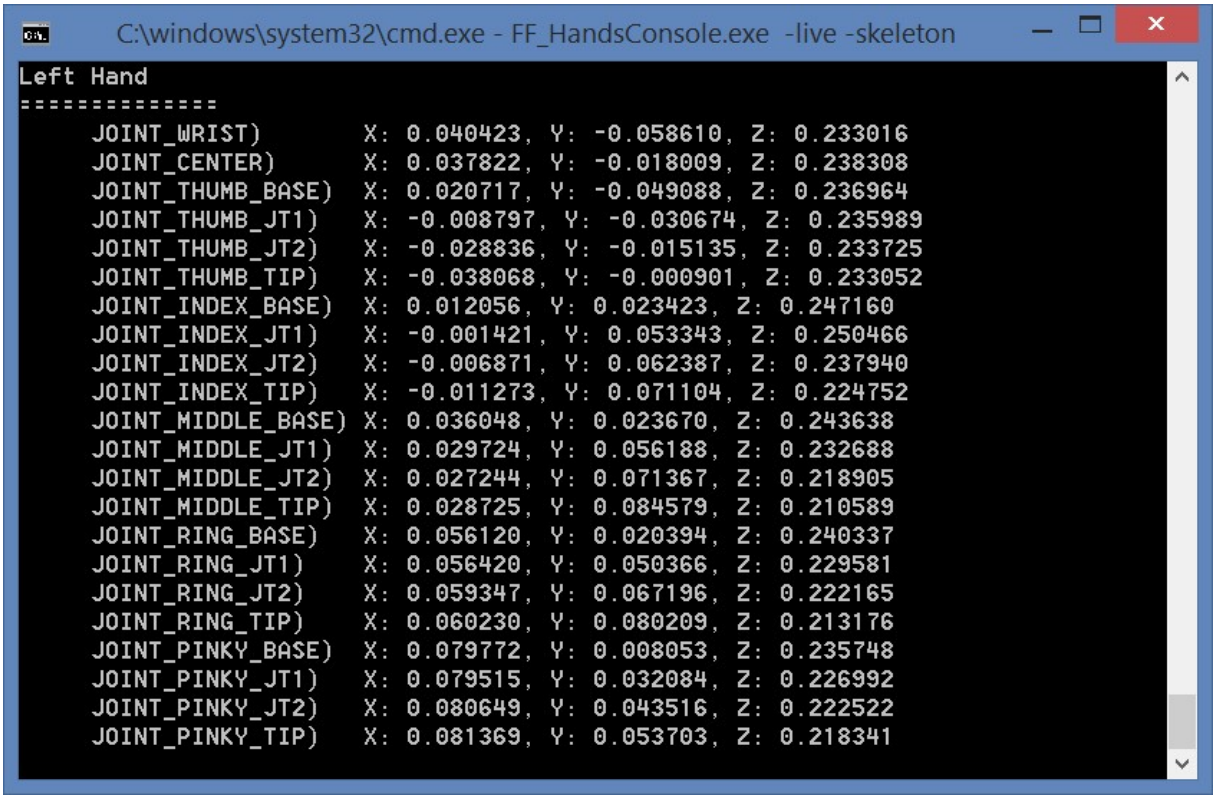


Figure 22: Sample Hands Console Window.

Command Options

Run the sample executable without any command line options shows the help message. The sample uses the following command line options:

Option	Description
`-live`	Stream data through a live camera. The option `-live` or `-seq` must

	present but not both.
`-seq <file>`	Stream data through a recorded file. The option `-live` or `-seq` must present but not both.
`-skeleton`	Print out joint world position information.
`-alerts`	Print out fired alerts.
`-gestures`	Print out fired gestures.

3.3.7 Sample: FF_IQSampleTool.cs

Introduction

The `FF_IQSampleTool.cs` application is based on the `FF_RawStreams.cs` sample. It showcases how to create tests for image quality that can be used with your camera. It features:

- Automatic Marker tracking
- XYZ on-hover output for depth streams
- Test for UV Map alignment error
- Test for depth uniformity
- Test for absolute depth values

This application requires the use of a Port3 Test Chart. If you do not have one, you can print your own (Included in the source is "Port 3 Chart.pdf" under the `Assets` directory), as illustrated in Figure 23, but your test results for the protruding square will not be accurate. (The protruding square measures 12cm x 12cm x 5cm – you can also create your own out of card stock paper and attach it to your printed chart).

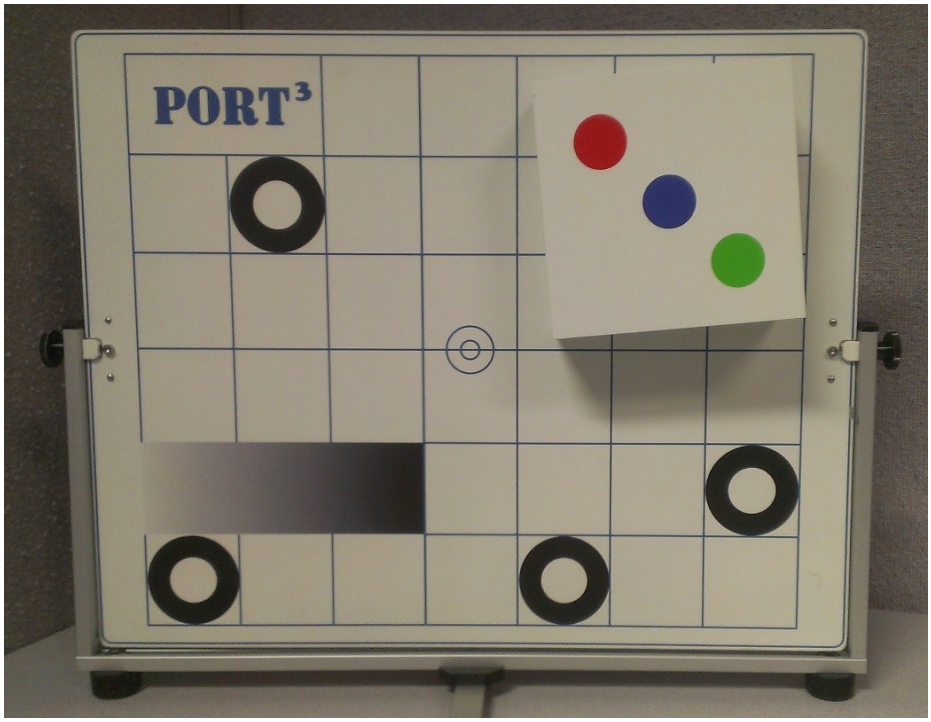


Figure 23: Test Chart for use with IQ Sample Tool

Getting Started

1. Obtain a chart. If you have a Port3 test chart, go to the next step. If not, you will have to

create one.

2. Setup your test area. Position your camera 50 cm away from the test chart. Ensure that the chart is parallel with the plane of the camera. If not parallel, test results can be skewed. (You can use a box, a metallic frame, the test chart with included spacer, etc. to do this).

3. Launch the application. If the camera is loaded successfully, the `Depth` radio button will be selectable. You will also see the 3D camera under the `Device` drop down menu.

4. Select the `Color` radio button and click on "`Start Streaming`" to run video.

5. For landmark tracking, the program must know the location of the test chart. To do this, you must outline the edges of the test chart. Drag your mouse cursor from one corner of the chart to the other. When done, click on "`Set RGB Border`". See Figure 24 for an illustration.

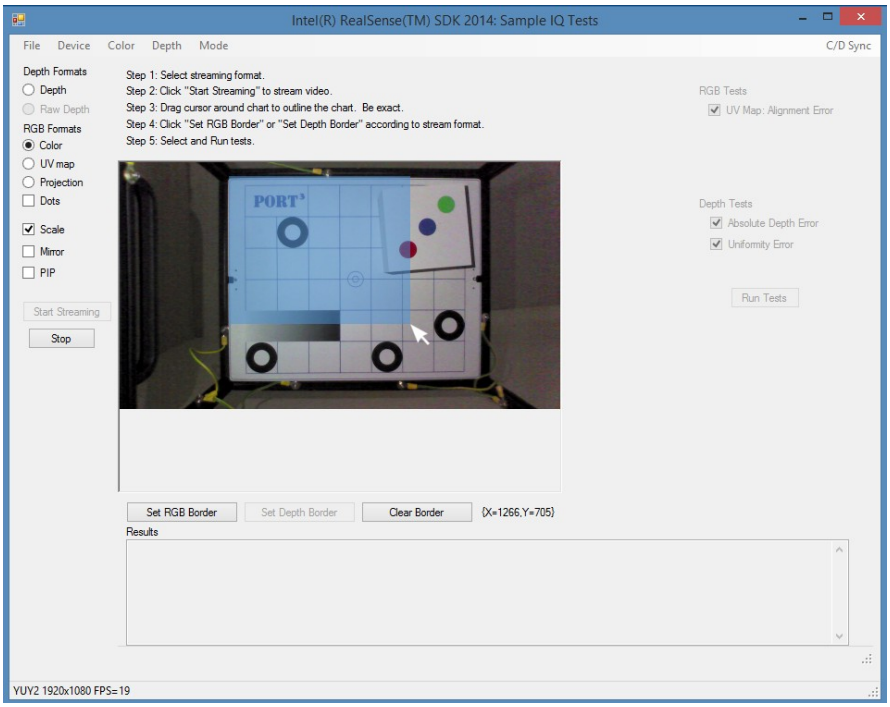


Figure 24: Specifying the chart edges for the Depth video stream.

6. Ensure the fiducial markers have been located properly. You should see the black circles outlined by red squares. You should also see the black crosshairs in the center of the black circle markers. Look at Figure 25 for the proper alignment (left) and the improper alignment (right). Improper alignment stems from poor lighting (for the RGB video) or from the rounding error introduced during the edge selection. If the markers are not detected properly, repeat step 5, but experiment with different selection sizes.

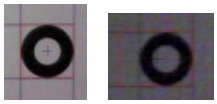


Figure 25: Left: Properly detected fiducial marker. Right: Detection Error

7. Your screen should look like Figure 26.

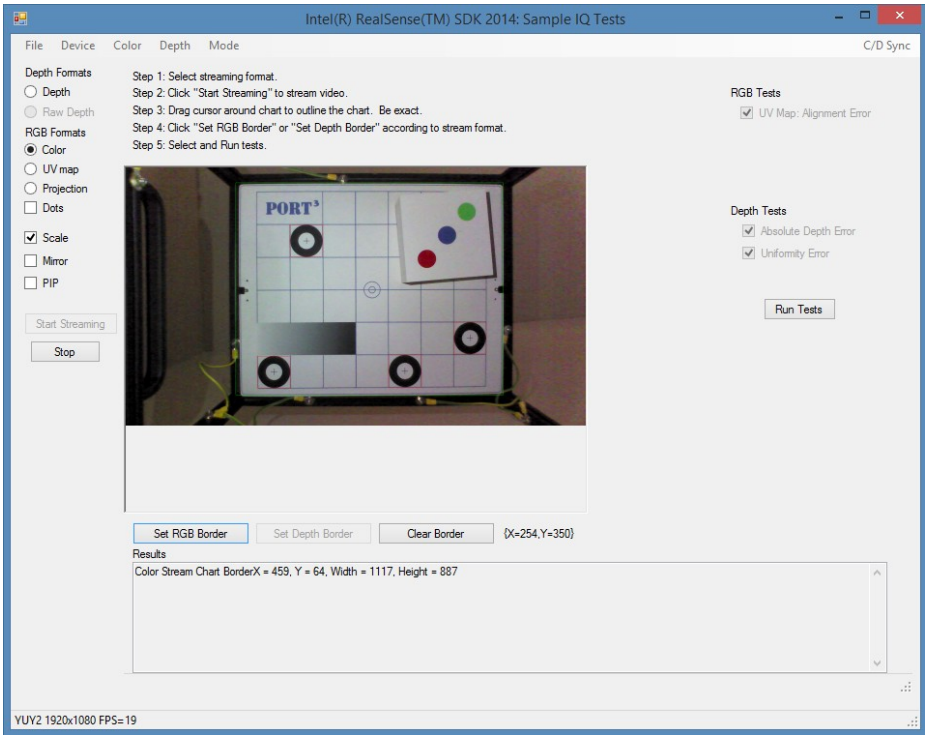


Figure 26: Proper Located Fiducial Markers.

8. Now select the `Depth` radio button. The video will change to a depth-only view.

9. Specify the chart edges by dragging your mouse from one corner to the opposite corner. Click on "`Set Depth Border`" (see Figure 1).

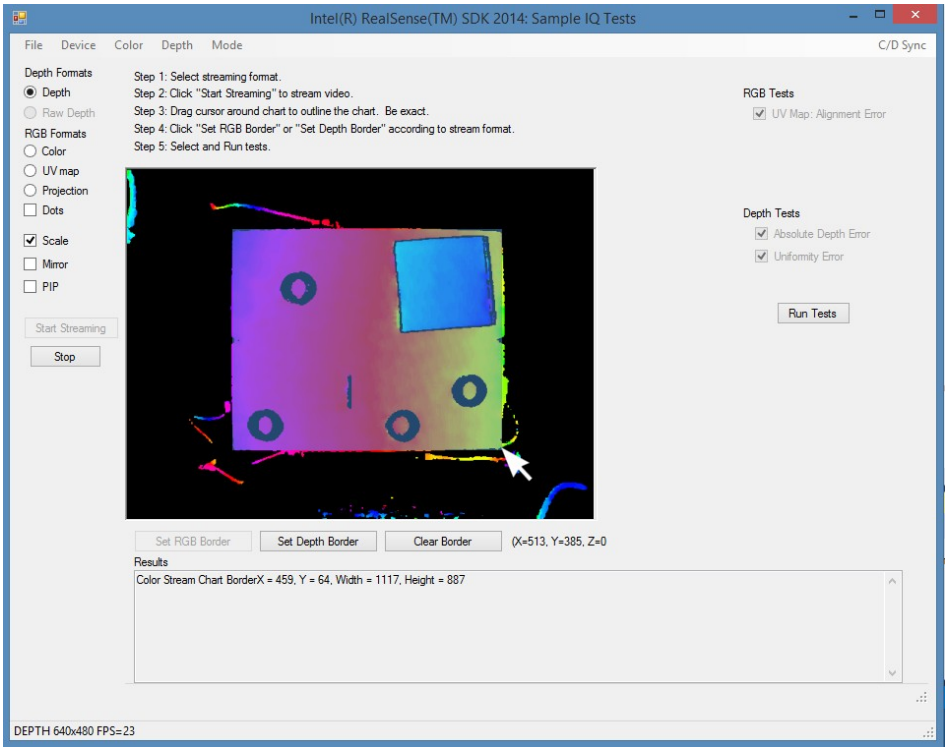


Figure 27: Specifying chart edges for Depth stream.

10. Your view should now look like Figure 28:

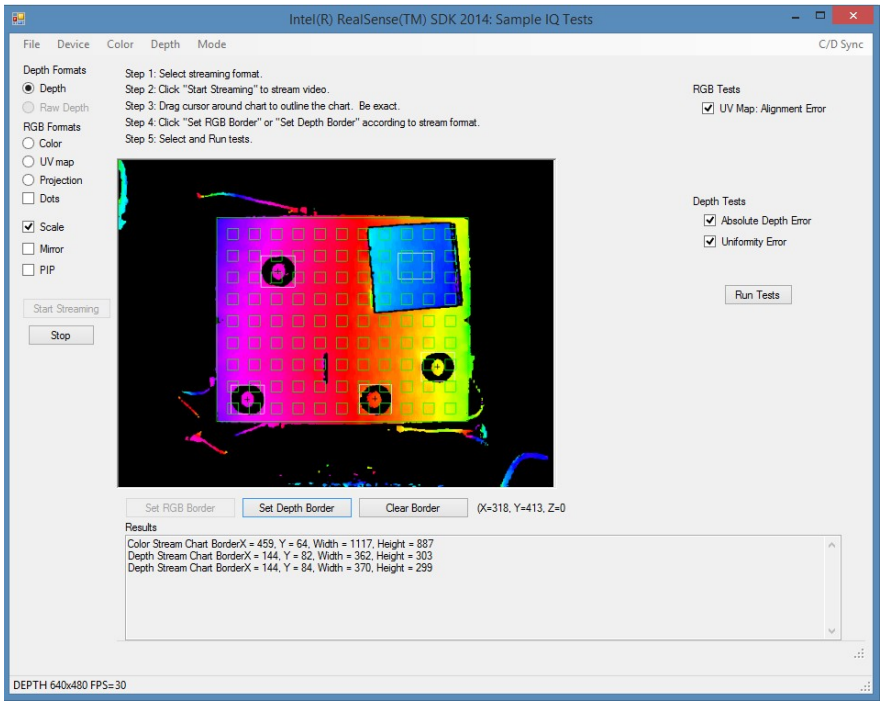


Figure 28: Selected Depth Edges.

11. Notice that the fiducial markers are outlined in the white squares. The green squares

are an array of ROIs (Regions of Interest) that are used for the Uniformity test. Check that all fiducial markers are properly aligned and that all green ROIs fall within the depth chart. If not, repeat Step 9.

12. Click `Run Tests` to analyze data.

Tests

- **UV Map: Alignment Error**

The UV Map aligns the RGB stream with the depth stream. This tests calculates the error in this alignment. An example of this error can be visibly seen when using the test chart. See Figure 29 for an example. Notice that the circular fiducial markers do not align. This is the UV Map alignment error.

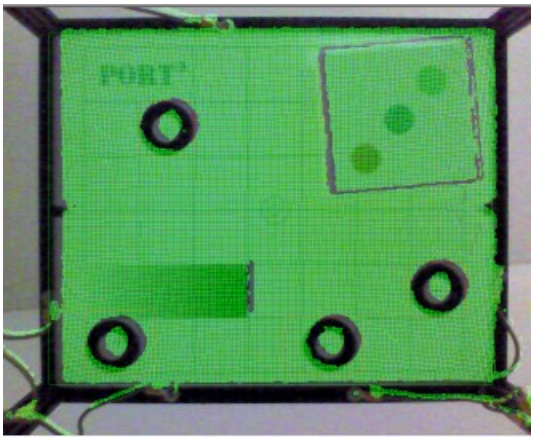


Figure 29 :Depth data (green dots) overlayed on color stream.

- **Absolute Depth Error**

This measures the depth from the surface of the protruding square and the camera plane. It is then compared to depth values recorded in the center of the black fiducial markers. Since the protruding square has a measurement of 12cm x 12cm x 5cm, the difference measured should equal 5 cm (50mm).

- **Uniformity Test**
  - This test creates an array of ROIs. The depth data of each ROI is then processed to get the mean and standard deviation. Uniformity can then be measured.
  - Any tilt error also becomes apparent with these results.

3.3.8 Sample: FF_ObjectTracking[.cs]

Introduction

The `FF_ObjectTracking[.cs]` are two samples that demonstrates the capabilities of the SDK object tracking module. The samples track 2D and 3D objects within a scene. The sample without suffix is a C++ sample and the one with the `.cs` suffix is a C# sample.

Launch

You can launch the prebuilt sample directly from the `$(RSSDK_DIR)/bin/$(Platform)` folder of the SDK installation, or compile and execute within the Microsoft Visual Studio. The project and source files are located under `$(RSSDK_DIR)/sample/FF_ObjectTracking` (C++) or `$(RSSDK_DIR)/framework/CSharp/FF_ObjectTracking.cs` (C#).

Menu Options

From the menu, you can select the input device, the Metaio* object tracking recognition module, the type of tracking the user wants to load (2D, 3D or Instant 3D) and the operating mode (live streaming mode, playback mode, and record mode). The GUI is shown in Figure 30.

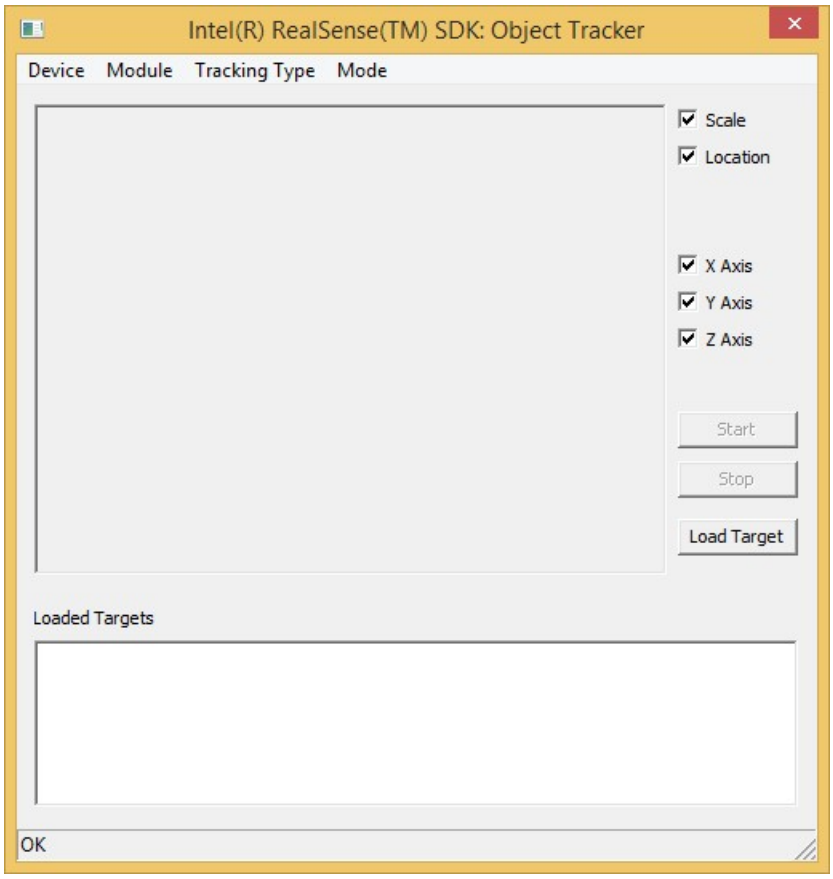


Figure 30: Sample object_tracker Menus

2D Tracking

From the `Tracking Type` menu, you can select the 2D Tracking mode and load in a `.png` or `.jpg` reference image for 2D tracking. The sample 2D tracking images are available under `Assets/2DTargets` shown in Figure 31.

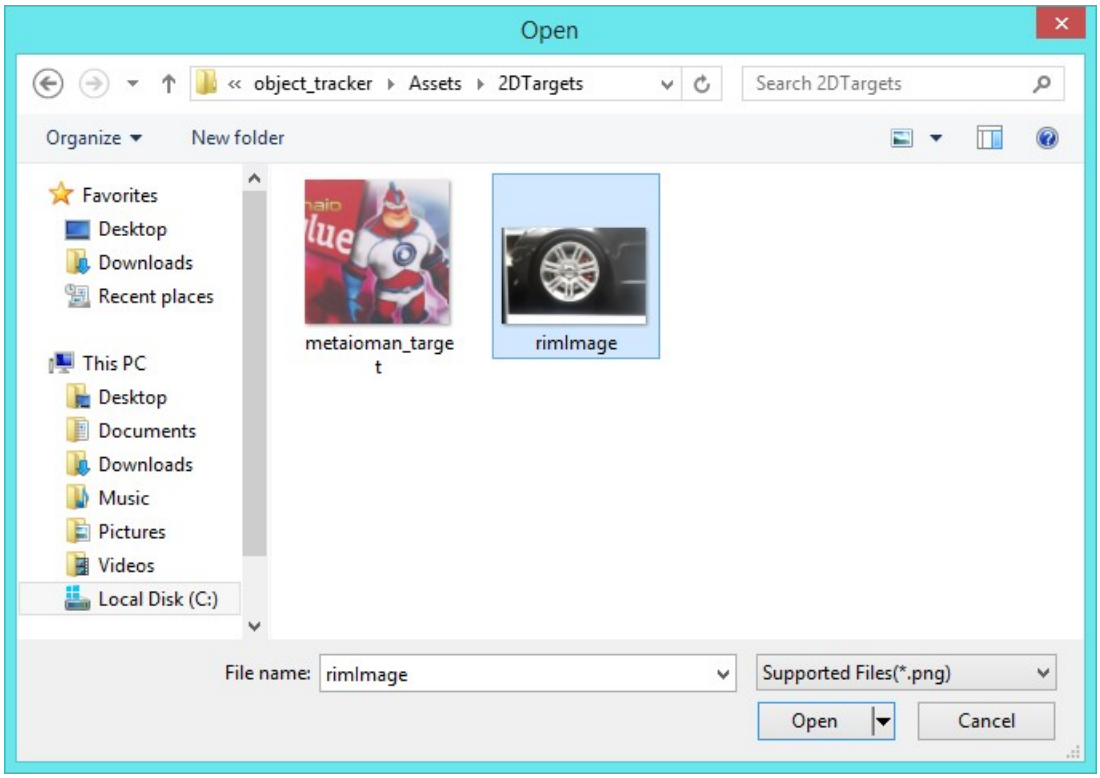


Figure 31: Sample object_tracker select 2D tracking image

After selecting the image, the user can press the `Start` button to begin to track the corresponding image. There will be projected axes on the tracked object as shown in Figure 32.

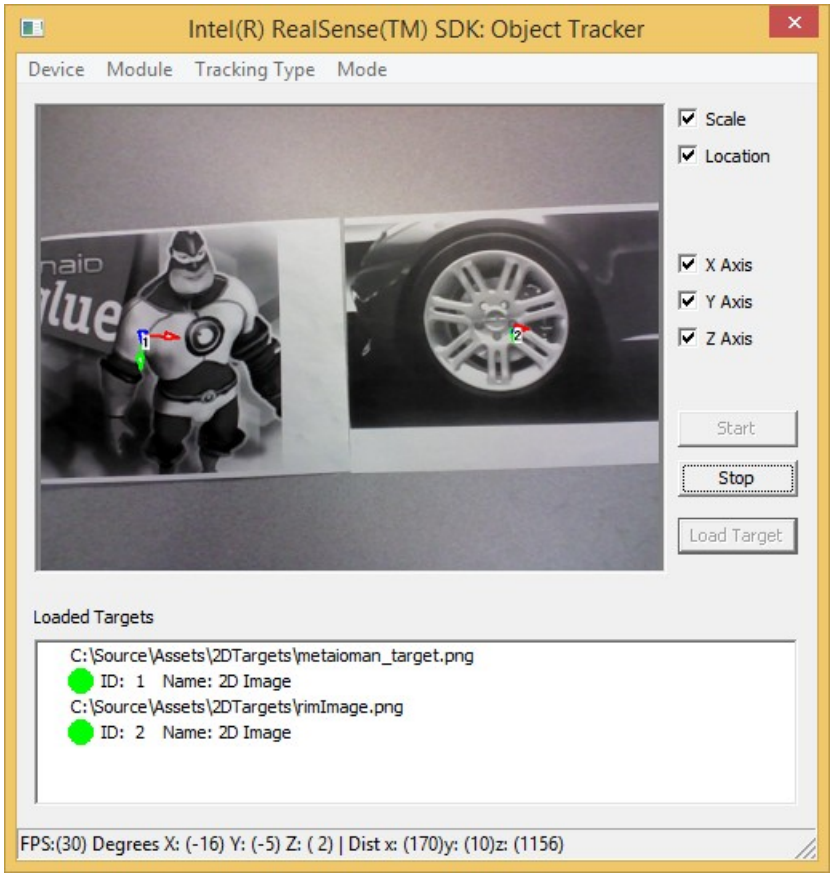


Figure 32: Sample object_tracker tracked 2D image

3D Tracking

From the `Tracking Type` menu, you can select the 3D tracking mode and load in a `.slam` or `.xml` file for 3D tracking, as illustrated in Figure 33. Instant 3D tracking is similar to regular 3D tracking, but the targets are automatically determined from the scene. Therefore, no targets need to be created and loaded ahead of time.

Figure 33: Sample object_tracker tracked 3D image

Other GUI Options

Click the `Start` button to activate the object tracker module. Check the `Scale` option to scale the images to the size of the display panel, and the `Location` option to toggle all axes on and off. Check the X axis, Y axis, and Z axis buttons to turn on and off the axes individually that projected on the tracked object. The `Load` Targets button will load different target files depending on the tracking type selected. Multiple targets can be loaded and tracked at the same time.

Map Creation, Extension and Alignment

The C++ sample also implements the 3D map creation functionality similar to the Metaio Toolbox application. The UI is similar to Figure 34. You can create a new map, extend an existing map with new features, and align a map to specify the coordinate system orientation and origin. The three buttons below the video panel allow switching between operations.

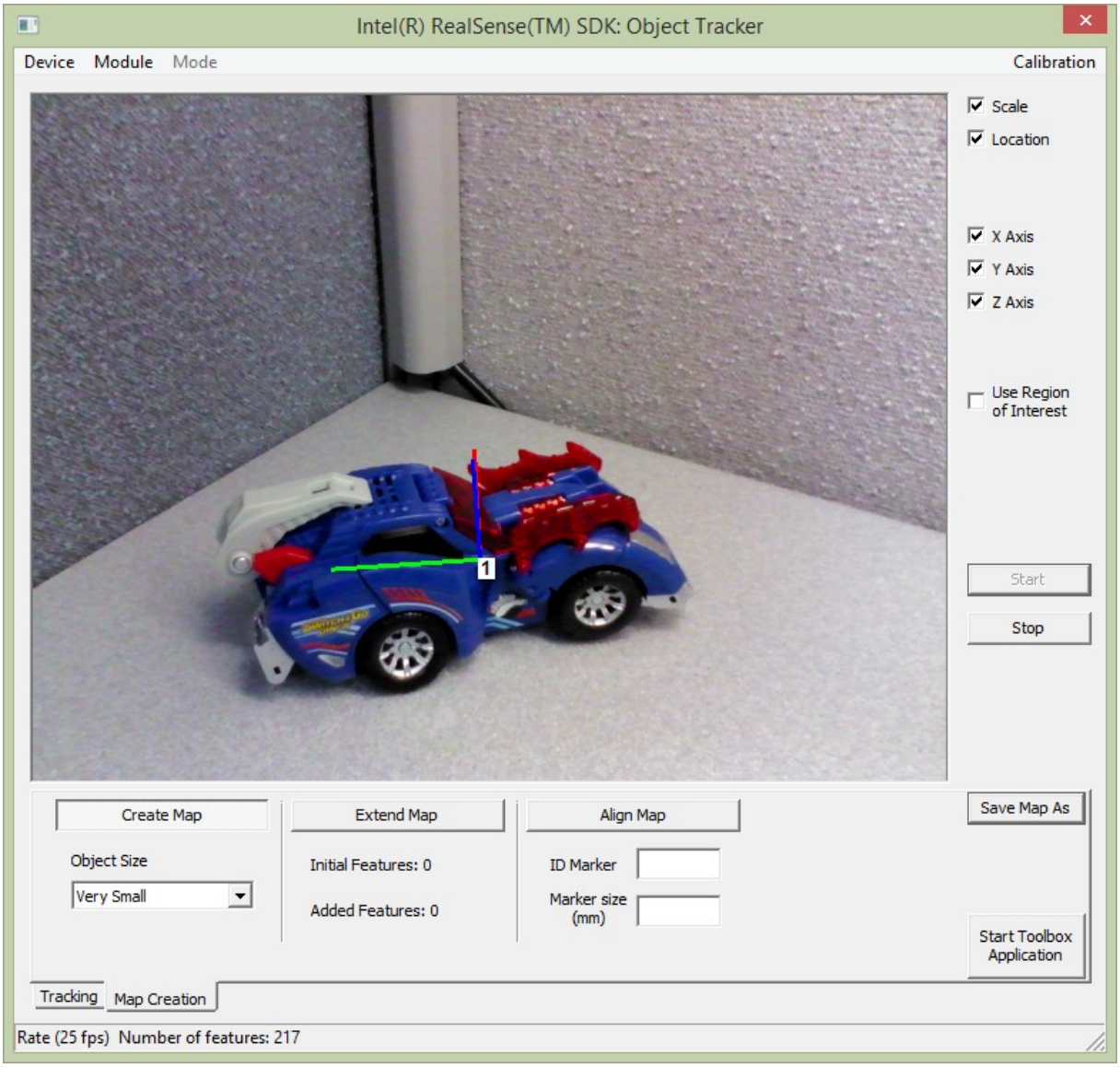


Figure 34: Map Creation, Extension and Alignment

On the map creation tab, the `Start/Stop` buttons control whether features are being detected for tracking purposes. If the `Start` button is not enabled, the condition is not met for the current operation (no map loaded, marker ID not specified, etc.). Once an operation is complete, the map file must be explicitly saved. It is not performed automatically.

While performing a map creation operation, the current status is provided in the status bar, including the number of features detected on the object. The sample application does not draw the feature points themselves, but the coordinate system pose is drawn as in tracking modes.

A region of interest can be specified for map creation and instant 3D tracking. The check box is enabled only when the region may be used. To specify a region, the video must be

streaming. Left click once on the video to specify the first corner. Left click again to specify the opposite corner. The active region is drawn in real time as the mouse is moved.

3.3.8.1 Using the Metaio* Tool Box

The `Metaio Toolbox` allows the developer to generate `.slam` and `.xml` files for use in the object tracker sample. The toolbox is located at `$(RSSDK_DIR)/contrib/Metaio/MetaioTrackerToolbox/MetaioTrackerToolbox.exe`.

The toolbox has 3 options: `3D Map Creation`, `Edge Tracking`, and `Camera Calibration`:

- Select the `3D Map Creation` mode to train a new `.slam` 3D tracking file, as shown in Figure 35.
  a. Select `640x480 – YUY2` as the stream.
  b. Set the sensitivity value based on the size of the object to be tracked.
  c. Check the `Set Region of Interest` option.
  d. Check the `Auto Adjust Region of Interest` option.
  e. Click the `Start` button.
  f. Move the camera (or the object) around until `Features Recorded` (top left) is at the desired amount.
  g. Save the recorded 3D map.

Give about 20 seconds for the initialization process. If features (red dots) do not appear, reset.

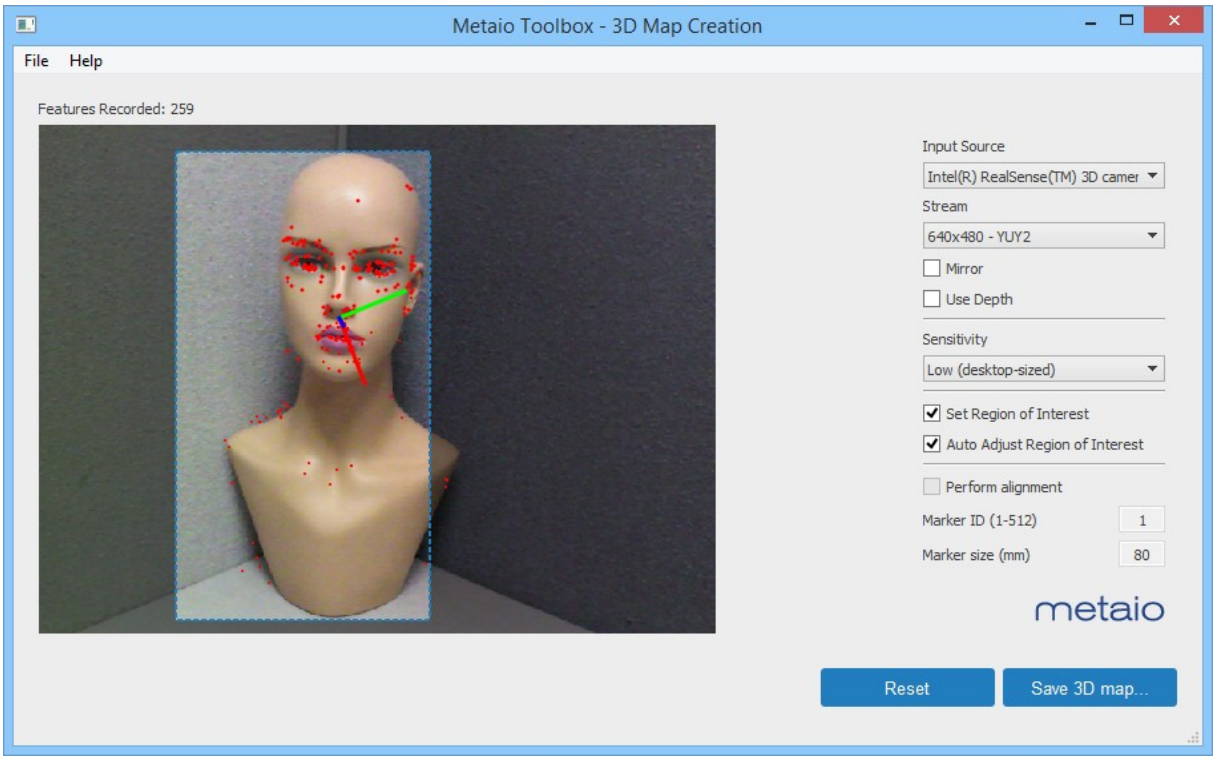


Figure 35: Metaio Toolbox 3D Map Creation Tool

- Select the `Edge Tracking` mode to load in a CAD `.obj` file for creating a new `.xml` 3D tracking file, as illustrated in Figure 36.

  a. Select an `.obj` file for tracking (sample obj files are under `Assets/3DTargets`)

  b. Choose the initial viewpoint sets that the object will track at.

  c. Click `Next`.

  d. Choose the `level of detail` value for the right level of values.

  e. Click `Accept. The .xml` 3D tracking file will be saved.

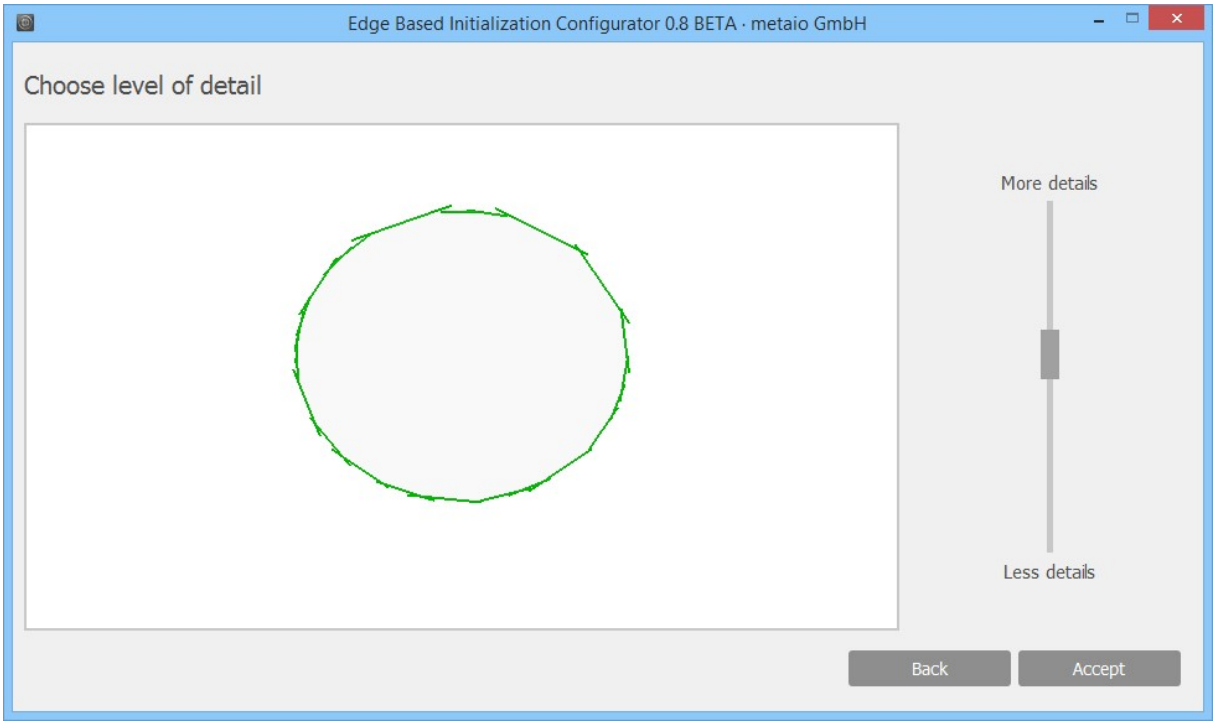


Figure 36: Metaio Toolbox .xml creation tool

Camera Calibration

1. Select the `Camera Calibration` mode.
2. Select `640x480-YUY2` as the stream.
3. Prepare the camera calibration markers (available under `$(RSSDK_DIR)/contrib/Metaio/MetaioTrackerToolbox/calibration`) and set the corresponding marker ID.
4. Click `Start` and wait until completion as shown in Figure 37.

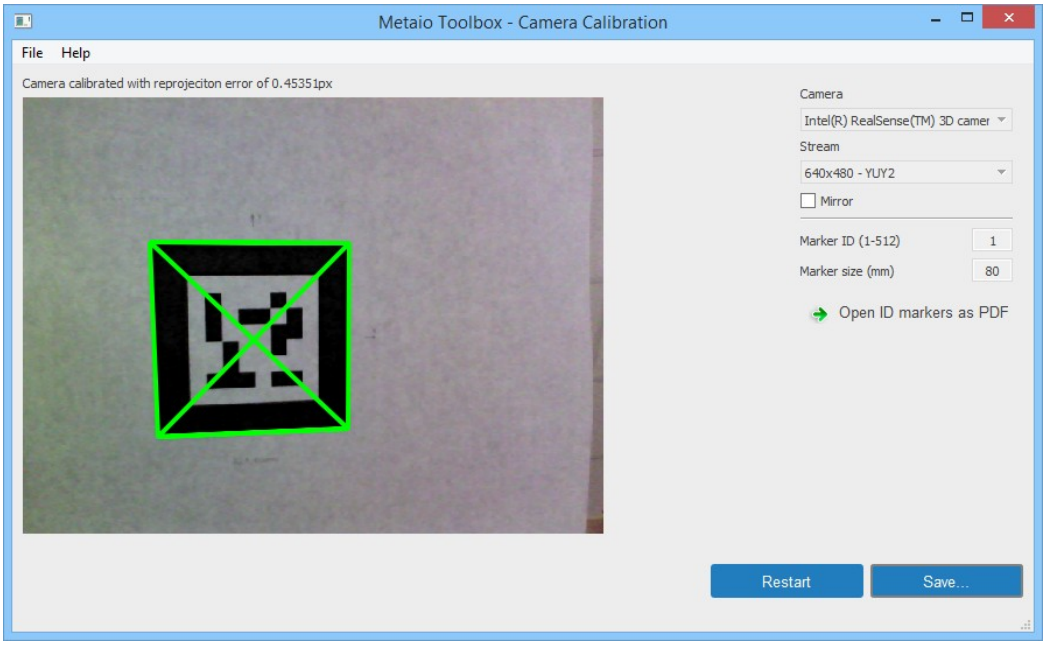
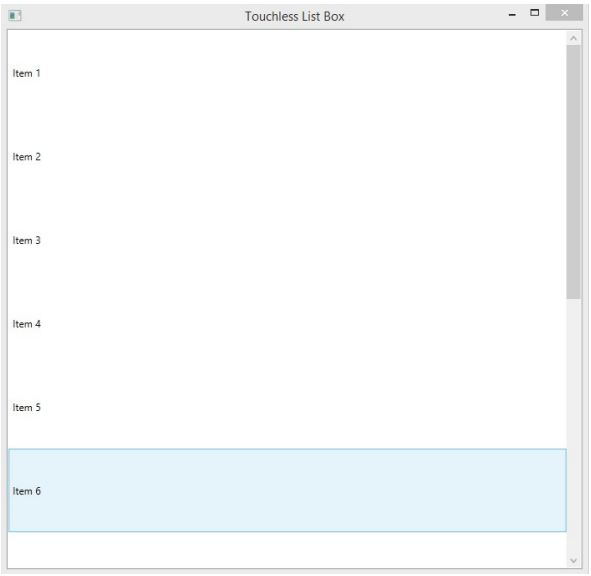


Figure 37: Metaio Toolbox camera calibration tool

3.3.9 Sample: FF_TouchlessListBox.cs



The `FF_TouchlessListBox.cs` sample shows how to listen to the touchless controller module events in order to implement a simple UI.

The sample presents a simple window with a basic table. You can use the following navigation gestures:

- Hold your hand about a foot (30cm) from the screen, and then move your hand up or down to highlight different table items.
- Perform a `Tap` gesture by pushing your hand forward in the air to select a specific table item.
- Pinch with your fingers and drag up or down to scroll the table.

The sample can playback a recorded sequence. To do so, run the sample with the file name as the command line argument. You can record the sequence by using any hand tracking sample(s).

The sample source is located under `$(RSSDK_DIR)/framework/CSharp/FF_TouchlessListBox.cs`.

3.3.10 Sample: FF_SpeechRecognition[.cs]

The `FF_SpeechRecognition` and `FF_SpeechRecognition.cs` samples demonstrate how to use the voice recognition module interface for voice command and control and dictation, in C++ and C# respectively.

The sample main window is shown in Figure 38. The menu can select the audio input source, the recognition module, the language (based on available settings of the recognition module) and the recognition mode (dictation or command and control). The status window shows the progressing status.

Click the `Start` button to start dictation or command and control, and the `Stop` button to stop the operation. During dictation, the dictation window shows the recognized sentence, as similarly shown in Figure 39. For command and control, click `[Enter New Command]` to enter a list of new commands. Then click the `Start` button to start command and control. The sample shows all hypotheses with confidence scores in the command and control window, as similarly shown in Figure 40.

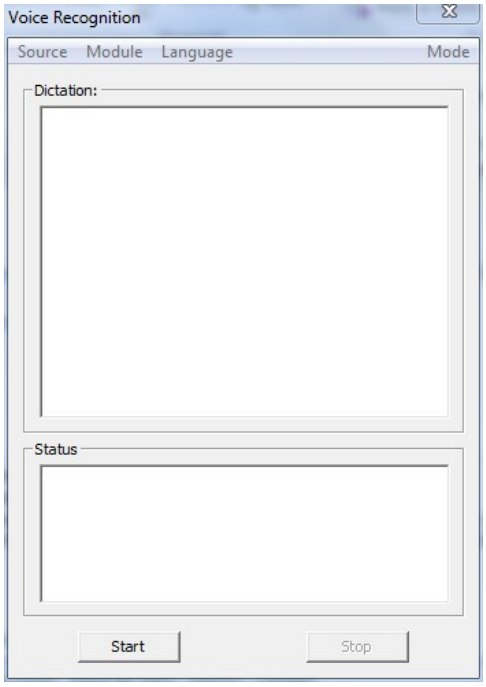


Figure 38: Voice Recognition Sample Main Window

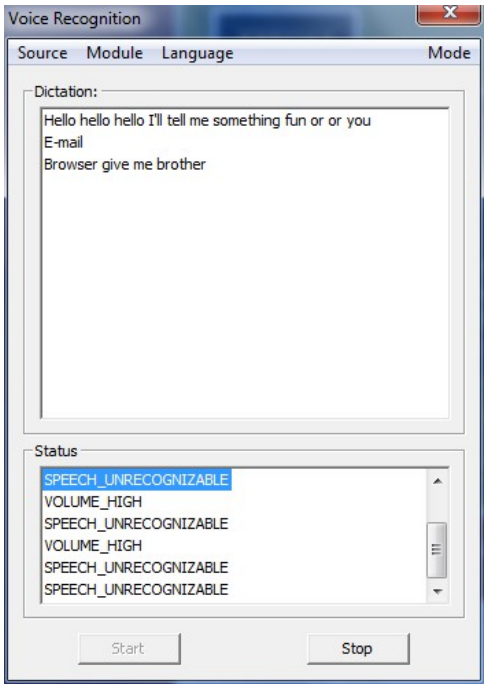


Figure 39: Voice Recognition Dictation Window

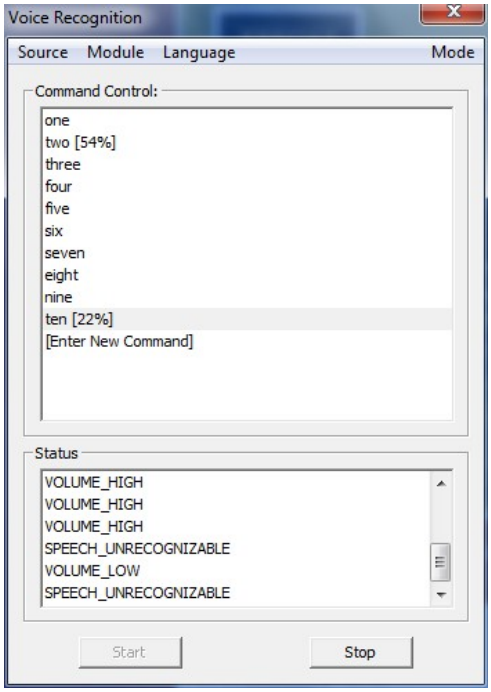


Figure 40: Voice Recognition Command Control Window

When in the dictation mode, from the `Mode` menu, you can add new vocabulary to the speech engine. A vocabulary file is just a text file that contains the list of words. An example is under the `examples` sub-directory of the sample folder.

When in the command and control mode, from the `Mode` menu, you can select to set the command and control grammar through the JSGF file (grammar specification.) See Java*

Speech Grammar Format for details. JSGF file examples are under the `examples` sub-directory of the sample folder.

3.3.11 Sample: FF_SpeechSynthesis[.cs]

The `FF_SpeechSynthesis` and `FF_SpeechSynthesis.cs` samples show how to use the voice synthesis interface for text to speech translation, in C++ and C# respectively.

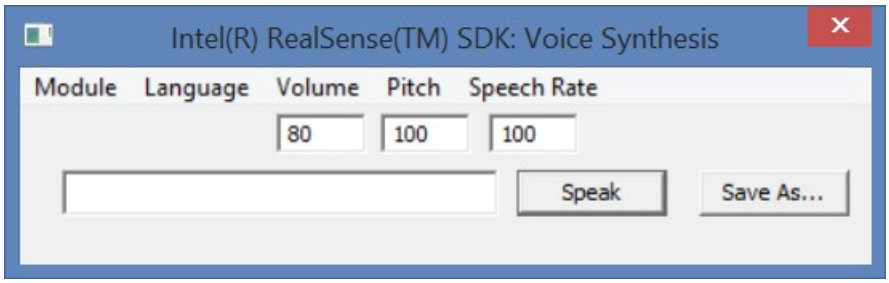


Figure 41: Voice Synthesis Sample Screen Shot

The screen shot is as shown in Figure 41.

- Use the `Module` menu to select a synthesis module.
- Use the `Language` menu to select a language setting.
- Change the volume, pitch, or speech rate using the corresponding Edit Box.
- Enter the sentence to be synthesized.
- Click the `Speak` button. The synthesized speech will be played back on the default audio device.
- Click the `Save As` button to save the synthesized speech to a wave file. (The C# sample does not have this button.)

3.4 R200 Samples

The samples work for the Intel® RealSense™ camera, model R200, a rear facing camera.

The sample names are prefixed with "`RF`", which stands for "`Rear Facing`".

3.4.1 Sample: RF_AugmentedRealitySP

Introduction

The `RF_AugmentedRealitySP` sample is a C++ game that illustrates how to use the SDK scene perception features for a simple game. The game scans your scenes and creates a 3D mesh model. You can shoot balls to the 3D mesh and see the balls stick to the mesh.

Launchy

The sample can be launched directly from the `bin` folder of the SDK installation, or compiled and executed using Microsoft Visual Studio. The project and source files are located inside the `sample/RF_AugmentedRealitySP` folder.

Operations

The sample window is similar to Figure 42:

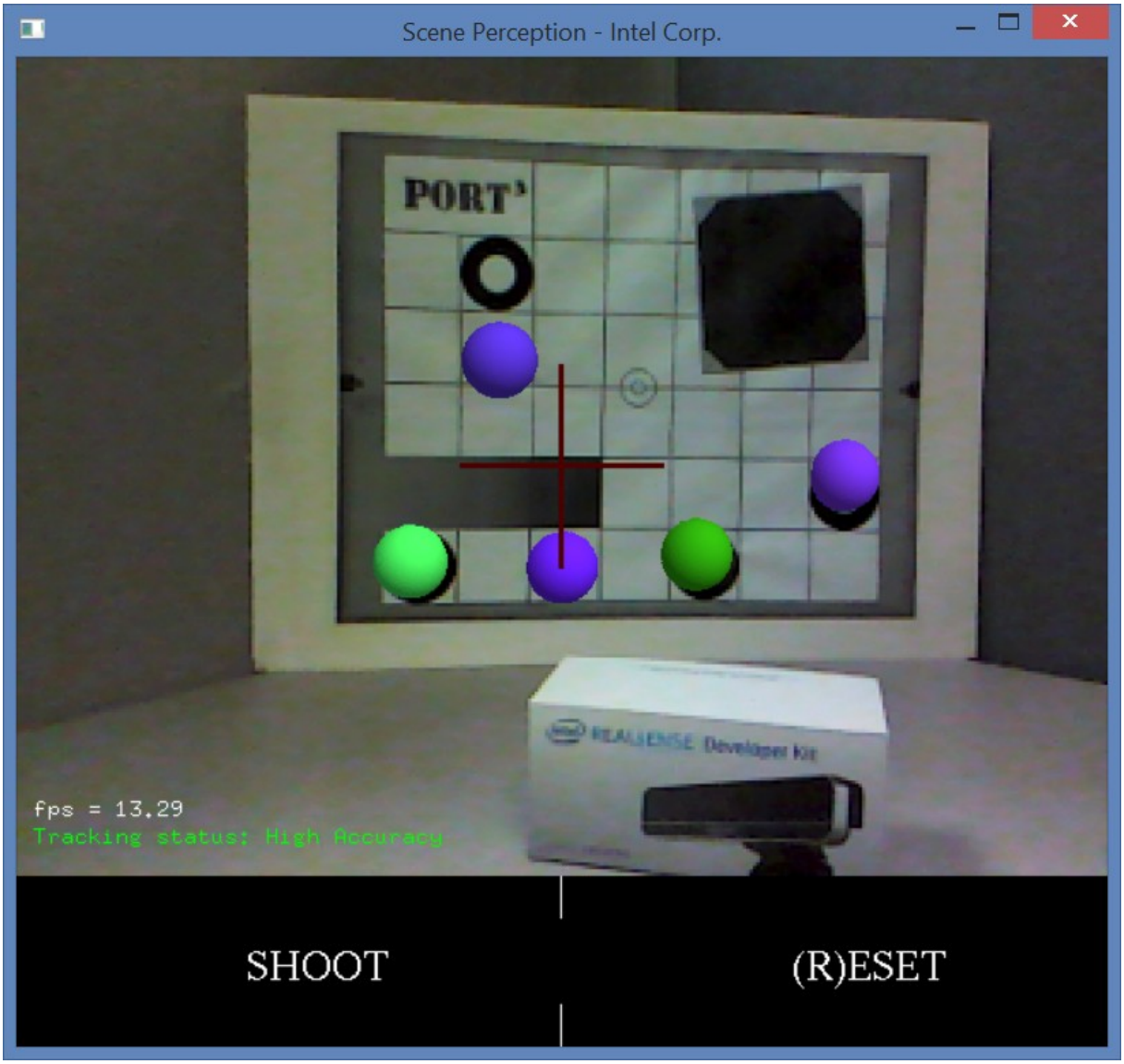


Figure 42: Sample DepthBlend

Click the SHOOT button to shoot the balls to your scenes and see the balls stick to the detected 3D mesh.

Click the RESET button to restart the scanning process.

3.4.2 Sample: RF_DepthBlendEP

Introduction

The `RF_DepthBlendEP` sample is a C++ application that demonstrates the depth blending feature of the SDK enhanced photography module. The sample blends a sticker image onto the logo and the user can adjust yaw/pitch/roll of the placement.

Launchy

The sample can be launched directly from the `bin` folder of the SDK installation, or compiled and executed using Microsoft Visual Studio. The project and source files are located inside the `sample/RF_DepthBlendEP` folder.

Menu Options

From the menu, the user can choose the input device as shown in Figure 43:

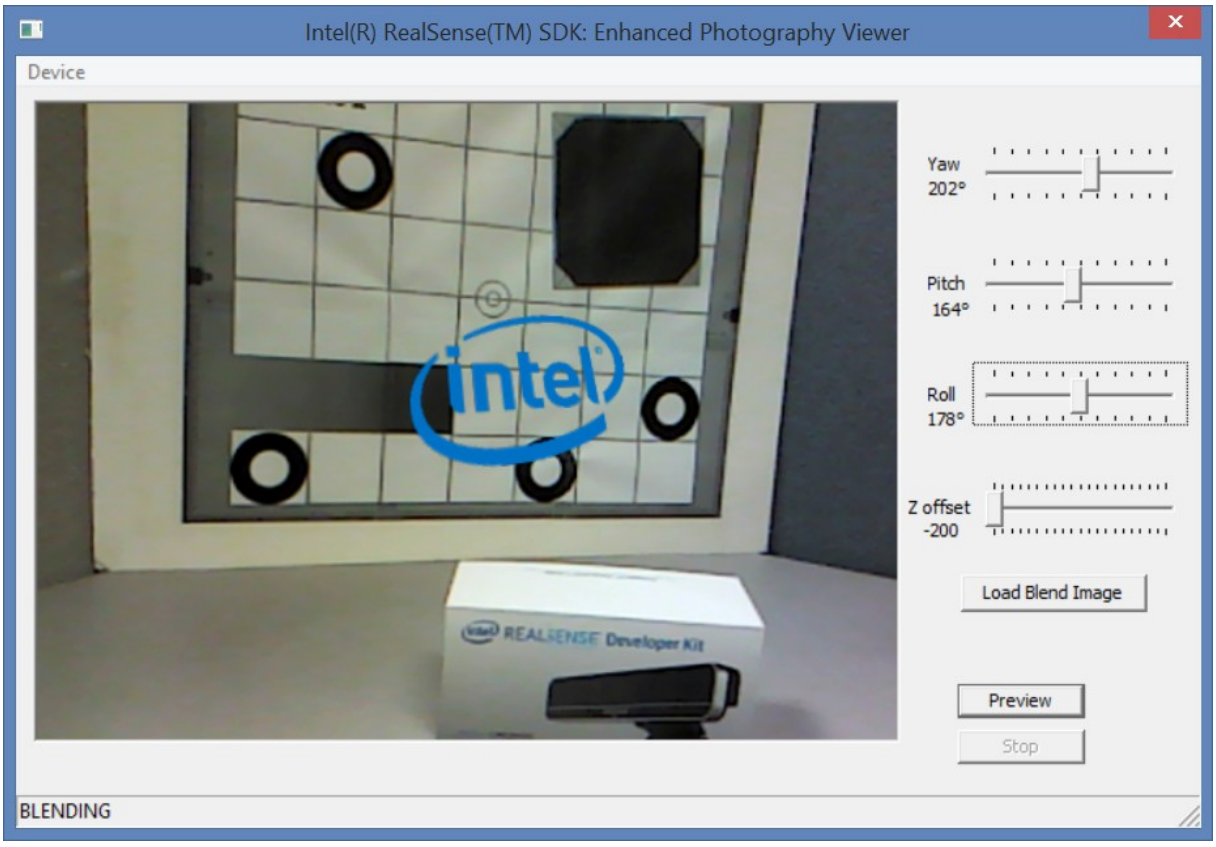


Figure 43: Sample DepthBlend

GUI Options

Click the `Preview` button to start living streaming. The same button switches between the `Preview` mode and the `Snapshot` mode. Choose a good picture and then click the `Snapshot` button. The image frame freezes to let you work on the snapshot image.

Click the `Load Blend Image` button to load the sticker image. The sample provides the Intel logo as a sample image. The user can choose any JPEG* or PNG* pictures.

Click on any position of the snapshot image to place the sticker image onto it. You can also adjust the raw/pitch/roll values of during placement.

Finally, click the `Stop` button to terminate the streaming process.

3.4.3 Sample: RF_EnhancedPhotography

Introduction

The `RF_DepthBlendEP` sample is a C++ application that demonstrates the SDK photo processing features. The sample shows how to take a snapshot out of a live streaming frames, and then perform refocusing and distance measurement operations.

Launch

The sample can be launched directly from the `bin` folder of the SDK installation, or compiled and executed using Microsoft Visual Studio. The project and source files are located inside the `sample/RF_EnhancedPhotography` folder.

Menu Options

From the menu, the user can choose the input device as shown in Figure 44:

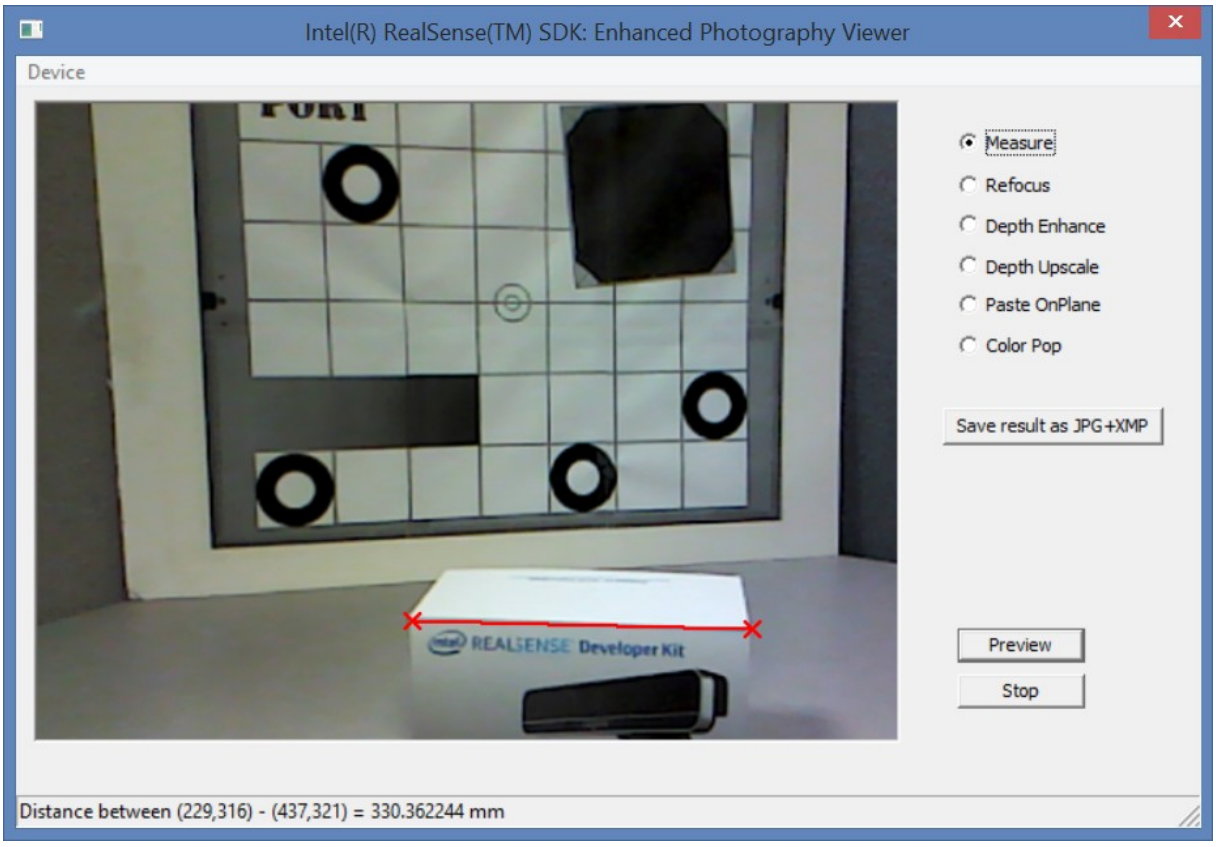


Figure 44: Sample Enhanced Photography

GUI Options

Click the `Preview` button to start living streaming. The same button switches between the `Preview` mode and the `Snapshot` mode. Choose a good picture and then click the `Snapshot` button. The image frame freezes to let you work on the snapshot image, with the

following available operations:

- **`Refocus`**: Click the `Refocus` radio button, and then any point on the snapshot image. The image is refocused to the mouse position.
- **`Measure`**: Click the `Measure` radio button and then any two points on the snapshot image. The sample calculates the distance between the two points, and shows the distance in the status bar.
- **`Depth Enhance`**: Click the `Depth Enhance` radio button. The sample shows the enhanced (hole-filled) depth image.
- **`Depth Upscale`**: Click the `Depth Upscale` radio button. The sample shows the depth image upscaled to the color image resolution.
- **`Paste OnPlane`**: Click the `Past OnPlane` radio button, and then two points on the snapshot image. The sample pastes the Intel logo onto the plane in the image. The two points specify the top left and bottom left points of the logo image.
- **`Color Pop`**: Click the `Color Pop` radio button and then a point on the snapshot image. The sample shows a color region around the specified point while suppressing color of the rest pixels of the image.

Click the `Save result as JPG+XMP` button to save the snapshot image. You can view the saved file from any browser.

Finally, click the `Stop` button to terminate the streaming process.

3.4.4 Sample: RF_MeasurementSP

Introduction

The `RF_MeasurementSP` sample is a C++ sample application that illustrates how to use the SDK scene perception features for simple measurement. The sample scans your scenes and creates a 3D mesh model. You can measure the distance between two points in the 3D mesh.

Launch

The sample can be launched directly from the `bin` folder of the SDK installation, or compiled and executed using Microsoft Visual Studio. The project and source files are located inside the `sample/RF_MeasurementSP` folder.

Operations

The sample window is similar to Figure 45:

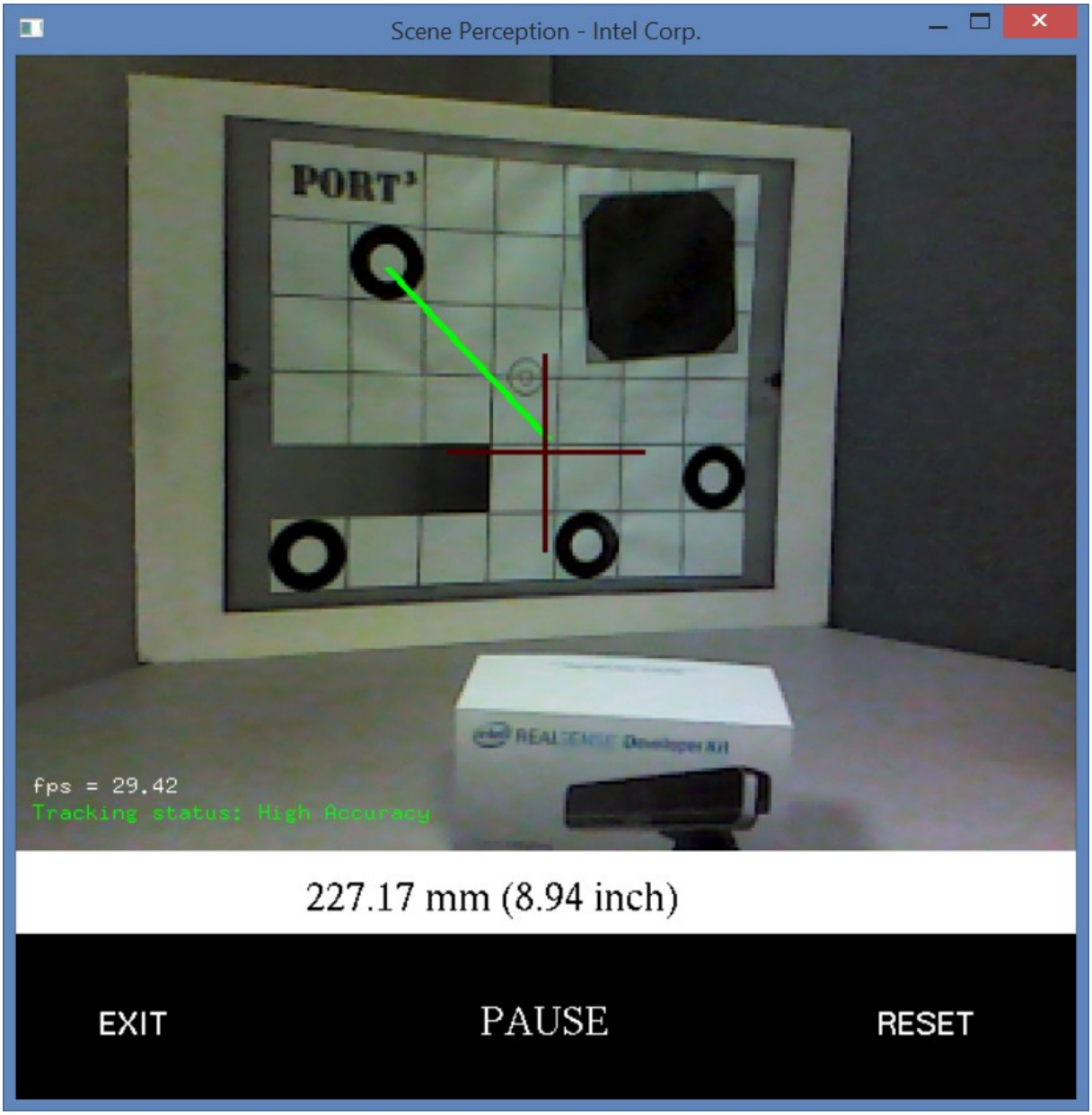


Figure 45: Sample DepthBlend

The sample works in the live streaming mode or in the paused mode. Use the `LIVE` button or the `Pause` button to switch between the two modes.

During live streaming, click the mouse button twice to locate two points (where the cross points at) on the scene. The sample shows the distance between the two points. In the pause mode, you can click directly on the image to locate the two points.

Click the `RESET` button to restart the scanning process and the `EXIT` button to quit.

3.4.5 Sample: RF_ScenePerception

The `RF_ScenePerception` sample illustrates the capability of the SDK Scene Perception module.

Build & Execute

Use the prebuilt binary under `$(RSSDK_DIR)/bin/$(Platform)` or re-build the sample using one of the solution files under `$(RSSDK_DIR)/sample/RF_ScenePerception` directory. The sample supports Microsoft* Visual Studio* 2012 or 2013.

Operations

When running the sample application, the camera needs to be oriented to some scene between `0.5-2.0` meters in front of the camera. Best scenes are indoor scenes such as a table top or a desk with enough structures/textures etc. Scenes with only flat unicolor walls or reflective materials (screens) or full of black objects do not reflect the emitted IR light, and therefore do not work well.

The sample application displays the input RGB or depth and shows the estimated camera pose and the reconstruction result of the scene as a mesh representation of the surface or a projected (ray-casted) view of the volume space. While moving the camera the relative camera/scene pose is estimated and displayed as a rectangle and a coordinate system representing the camera position and orientation.

The user can change the viewpoint to examine the reconstructed volume and enable/ disable some features like meshing the volume or extending the volume. The user can also save the reconstructed volume as mesh in an OBJ file.

The GUI of the sample application is similar to Figure 46. The live view pane is on the left. Right clicking the mouse button on it (or use the keyboard shortcut "`i`") changes the image view from color to depth. The 3D reconstructed scene pane is on the right. Use the mouse interaction (or the keyboard arrow keys) to change the viewpoint: left button for rotation, right button for translation, and middle button for zoom.

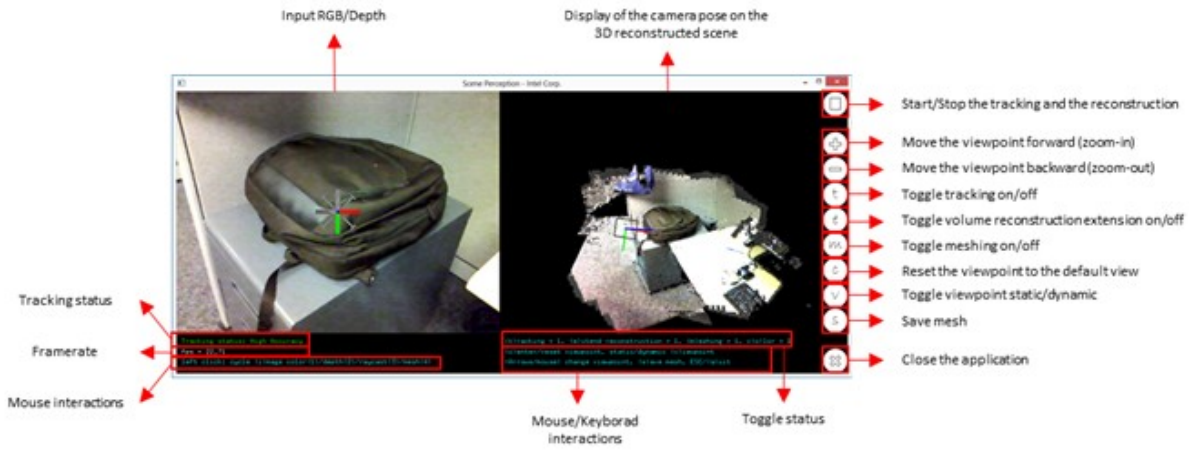


Figure 46: The Scene Perception Sample Window

Click on the right side GUI buttons for the following operations (from top to bottom):

- **Start/Stop** the tracking and the reconstruction.
- **Zoom In** the 3D reconstructed scene display. Keyboard shortcut: "+".
- **Zoom Out** the 3D reconstructed scene display. Keyboard shortcut: "–".
- **Toggle** the camera tracking state. Keyboard shortcut: "t".
- **Toogle** the volume reconstruction state. Keyboard shortcut: "e".
- **Toogle** the meshing state. Keyboard shortcut: "m".
- **Center/Reset** the camera view port. Keyboard shortcut: "c".
- **Freeze/Unfreeze** the camera view port: static/dynamic. Keyboard shortcut: "v".
- **Save** the mesh data to an OBJ file. Keyboard shortcut: "s".
- **Exit** the program. Keyboard shortcut: "q".

4 SDK Framework Samples

This section describes the SDK framework samples.

4.1 Java Sample: DF_FaceTracking

The `$(RSSDK_DIR)/framework/Java/DF_FaceTracking` sample shows how to run the SDK in Java for face tracking. The sample is tested under Java JDK 1.7.0_11.

Use the following steps to run the sample:

1. Set the `JAVA_HOME` environment variable to the Java JDK installation location.
2. Run the script `$(RSSDK_DIR)/framework/Java/DF_FaceTracking/run.bat`. You should see the console output similar to Figure 47.

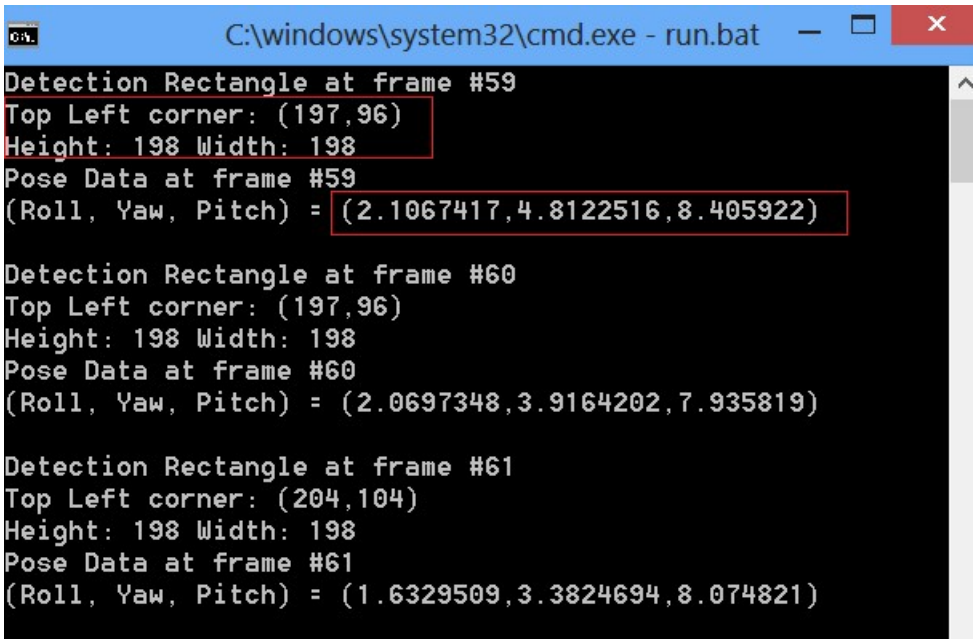


Figure 47: The face_tracking_java Sample Window

4.2 Java Sample: FF_HandsViewer

The `$(RSSDK_DIR)/framework/Java/FF_HandsViewer` sample shows how to run the SDK in Java for hand tracking. The sample is tested under Java JDK 1.7.0_11.

Use the following steps to run the sample:

1. Set the `JAVA_HOME` environment variable to the Java JDK installation location.
2. Run the script `$(RSSDK_DIR)/framework/Java/FF_HandsViewer/run.bat`. You should see the console output similar to Figure 48.

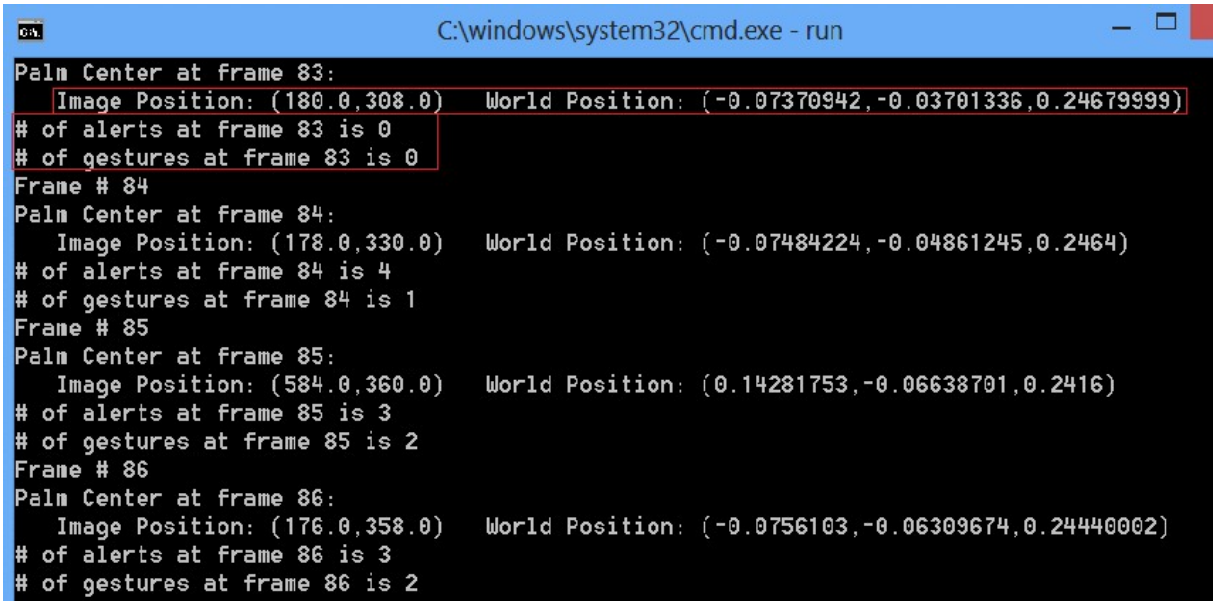


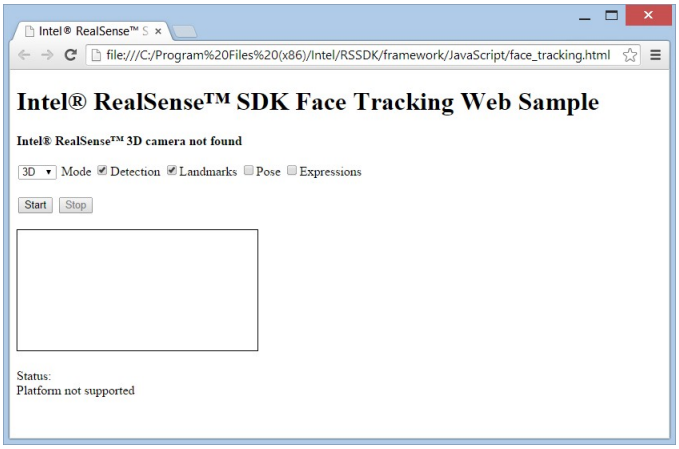
Figure 48: The hands_viewer_java Sample Window

4.3 JavaScript Samples

The SDK provides the following JavaScript samples:

Face Tracking

The face tracking sample illustrates how to use the SDK face tracking feature in a web browser. The source code is located at `$(RSSDK_DIR)/framework/JavaScript/DF_FaceTracking.html`. Click the HTML page to run the sample.



Click the `Start` button to start face tracking and the `Stop` button to stop the tracking. You can configure additional check-box options such as the tracking mode and the tracking features (detection, landmark, pose, and expression.)

The sample displays the augmented tracking image in the rectangle box.

Hands Viewer

The hands viewer sample illustrates how to use the SDK hand tracking feature in a web browser. The source code is located at `$(RSSDK_DIR)/framework/JavaScript/FF_HandsViewer.html`. Click the HTML page to run the sample.

Click the `Start` button to start hand tracking and the `Stop` button to stop the tracking. You can configure additional check-box options such as enabling gesture recognition and alerts.

The sample displays the augmented tracking image in the rectangle box.

Blobs Viewer

The blobs viewer sample illustrates how to use the SDK blob tracking feature in a web browser. The source code is located at `$(RSSDK_DIR)/framework/JavaScript/FF_BlobsViewer.html`. Click the HTML page to run the sample.

Click the `Start` button to start blob tracking and the `Stop` button to stop the tracking. The tracking data is printed at the line below `Frame Summary`.

The rectangular box is reserved for future extension to visualize the blob points. It is not used.

Voice Recognition

The voice recognition sample illustrates how to use the SDK speech recognition feature in a web browser. The source code is located at `$(RSSDK_DIR)/framework/JavaScript/FF_SpeechRecognition.html`. Click the HTML page to run the sample.

Configure the recognition mode (dictation or command & control) using the `Mode` drop down list. Then click the `Start` button to start speech recognition and the `Stop` button to stop it. The sample displays the recognized words or sentences in the `Recognition result` rectangle and any alert messages in the `Alerts` rectangle.

4.4 Processing Sample: FF_HandTracking

This sample `$(RSSDK_DIR)/framework/Processing/FF_HandTracking` shows how to use the SDK within the Processing framework. The sample is tested under Processing 2.1.2.

Use the following steps to run the sample:

1. Copy the **`libraries`** directory to your sketch location. You can find your sketch location from **`File->Preference`**. To verify, relaunch Processing and go to **`sketch/import`**, you should see **`libpxcclr_processing`** in the sub-menu.
2. Click on the **`HandTracking/HandTracking.pde`** to launch the sample.
3. Click the playback button to run the sample. You should see something similar to Figure 49.

Figure 49: The HandTracking Sample Window

4.5 Unity Sample: DF_FaceAnimation

The `DF_FaceAnimation` showcases a 3D avatar – a virtual character which mimics the user's facial expressions. The sample uses the Unity game engine for 3D visualization and animation.

Directory Structure

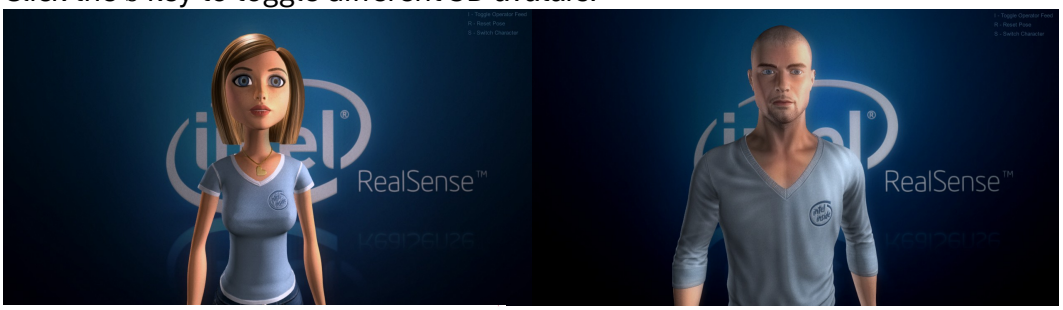
Directory/File	Description
Assets	The Unity assets and project setting directories that contain source code.
ProjectSettings	

Operation Instructions

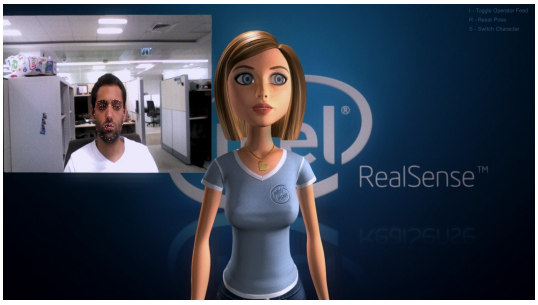
The sample is located under `$(RSSDK_DIR)/framework/Unity/DF_FaceAnimation`. Copy the sample source to a writable folder and click `Assets/Face Tracking Demo Scene.unity` to launch the sample project. You can build the sample or make any modification. The sample is tested under Unity Pro 4.3.4f1.

The sample displays a 3D cartoon model at the center of the screen.

Click the `S` key to toggle different 3D avatars.



You can reset the head orientation with the `R` key, and show/hide a camera viewer with the `I` key:



If you see a warning message "`NullReferenceException: Object reference not set to an instance of an object UnityEditor.DockArea.OnGUI()`" while running your application in Unity Editor, close the `Game` and `Scene` windows and redo the window layout. The warning message should go away after the layout change. See also http://neareal.net/index.php?ComputerGraphics%2FUnity%2FTips%2FError%20UnityEditor.DockArea.OnGui() (In Japanese.)

4.6 Unity Sample: FF_CubeSense

This sample `FF_CubeSense` shows how to run the SDK with the Unity game engine supporting the web player platform. The sample requires Unity v4.1.0 or later.

Copy the sample source to a writable folder and double click `Assets/CubeSense.unity` to launch the Unity editor, and then run the game. The screen shot is similar to Figure 50 (in Unity Editor) or Figure 51 (in an internet browser). By default, the sample initializes to run face tracking only. Click the buttons on the left upper corner to switch to the hand tracking mode, or adding the image display mode.


The image mode is a stretch for the network transmission bandwidth. The application will slow down significantly with constant image streaming.

The script `SenseInput.cs` initializes the camera streaming, face and hand tracking. When there are available data (samples from the camera, or data from face tracking or hand tracking), the script sends events to the `CubeMotion.cs` script.

The script `CubeMotion.cs` visualizes face and hand tracking data. The script visualizes the face tracking data by pointing the cube to the normalized view port position of the detected nose position coordinates. The script visualizes the hand tracking data by rotating the cube left if the detected hand position is at the left side of the screen, and rotating the cube right if the hand position is at the right side.

The script `TextWindow.cs` prints out the detected face and hand tracking data using a grayed color in the background.

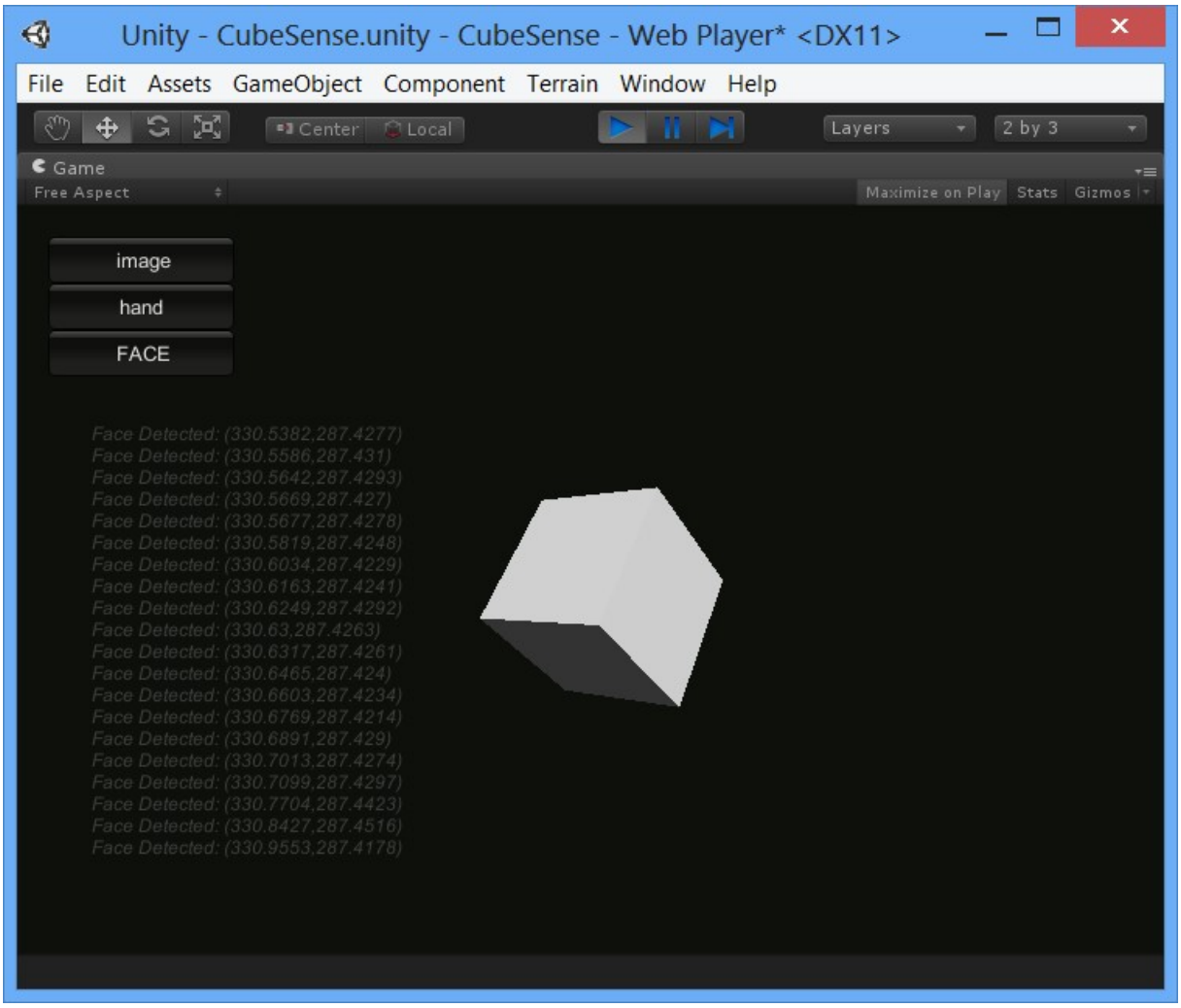


Figure 50: The CubeSense Sample Window in Unity Editor

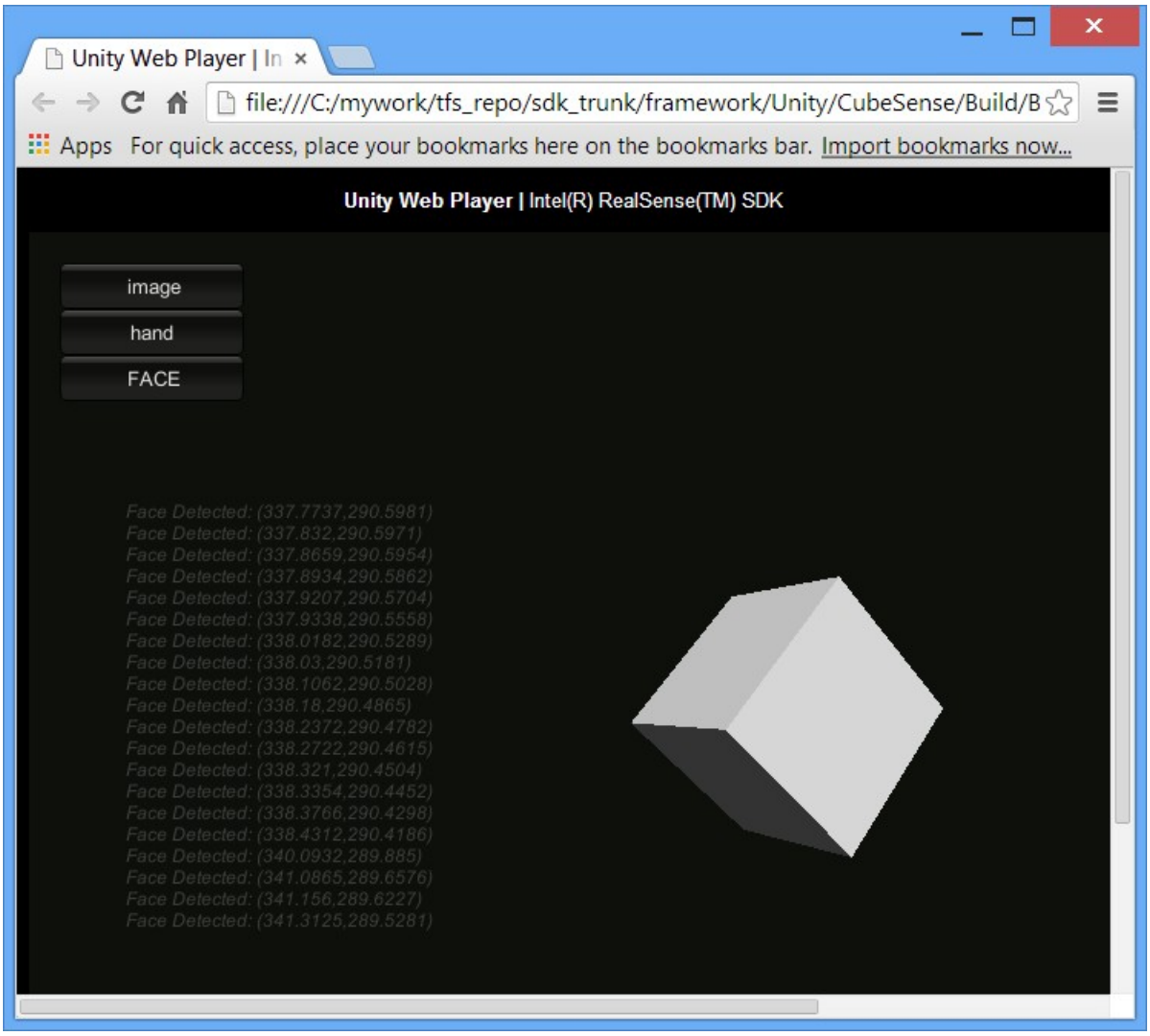


Figure 51: The CubeSense Sample Window in the Google Chrome Browser

If you see a warning message "`NullReferenceException: Object reference not set to an instance of an object UnityEditor.DockArea.OnGUI()`" while running your application in Unity Editor, close the `Game` and `Scene` windows and redo the window layout. The warning message should go away after the layout change. See also http://neareal.net/index.php?ComputerGraphics%2FUnity%2FTips%2FError%20UnityEditor.DockArea.OnGui() (In Japanese.)

4.7 Unity Sample: FF_HandsAnimation

The `FF_HandsAnimation` sample visualizes a 3D hand skeleton (or two hands) in the scene when the sample detects that there is a hand in front of the camera. The sample demonstrates the hand tracking module features. If there are hand gestures, the sample displays the recognized gesture text on top of the screen. This sample requires Unity v4.1.0 PRO.

Directory Structure

Directory/File	Description
Assets	The Unity assets and project setting directories that contain source code.
ProjectSettings	

Operation Instructions

The sample is located under `$(RSSDK_DIR)/framework/Unity/FF_HandsAnimation`. Copy the sample source to a writable location and then click on `Assets/Scenes/FF_HandsAnimation.unity` to execute the sample.

Raise your hand(s) in front of the camera. The sample should show the hand skeleton(s) in the cloud, similar to Figure 52.

You can additionally change the view perspective (rotate the main camera) by using the left and right arrows. Use the `R` key to reset/restart, and the `Q` key to quit.

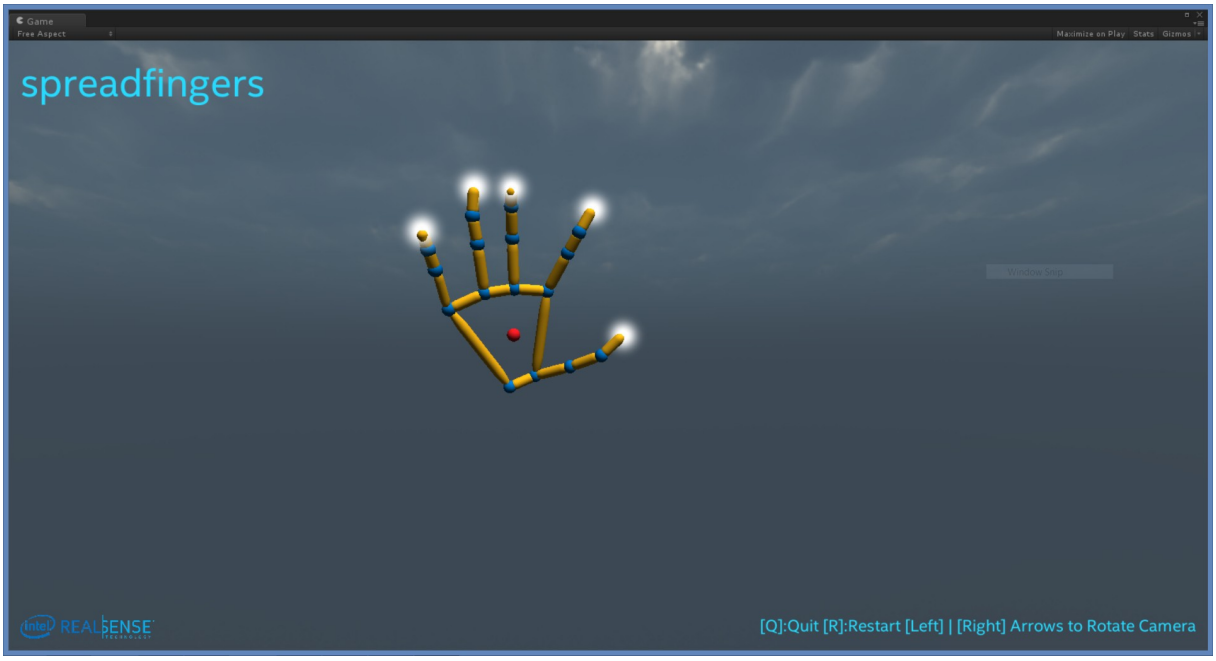


Figure 52: The Hands Animation Main Window.

If you see a warning message "`NullReferenceException: Object reference not set to an instance of an object UnityEditor.DockArea.OnGUI()`" while running your application in Unity Editor, close the `Game` and `Scene` windows and redo the window layout. The warning message should go away after the layout change. See also http://neareal.net/index.php?ComputerGraphics%2FUnity%2FTips%2FError%20UnityEditor.DockArea.OnGui() (In Japanese.)

Design Points

The script `Scripts/HandsViewer.cs` implements the `SenseManager` logic that initiates the hand tracking module, retrieves the hand location data when ready, and then maps to the hand skeleton, as a prefeb under `Prefebs`.

4.8 Unity Sample: FF_NineCubes

This sample `FF_NineCubes` shows how to run the SDK with the Unity game engine supporting the PC Standalone platform. The sample requires Unity v4.1.0 PRO (for the plugin capability) or later.

Copy the sample source to a writable folder and double click `Assets/NineCubes.unity` to launch the Unity editor, and then run the game. The screen shot is similar to Figure 53. By default, the sample initializes to run face tracking only. Click the buttons on the left upper corner to switch to the hand tracking mode, or adding the image display mode.

The script `SenseInput.cs` initializes the camera streaming, face and hand tracking. When there are available data (samples from the camera, or data from face tracking or hand tracking), the script sends events to the `CubeMagic.cs` script.

The script `CubeMagic.cs` visualizes face and hand tracking data. For either tracking data (nose position or hand position), the tracking coordinates are normalized to the view port and then evenly divided into nine pieces, each controlling the spinning of a cube. If the image display mode is enabled, the current depth frame is painted onto the currently spinning cube.

The script `TextWindow.cs` prints out the detected face and hand tracking data using a grayed color in the background.

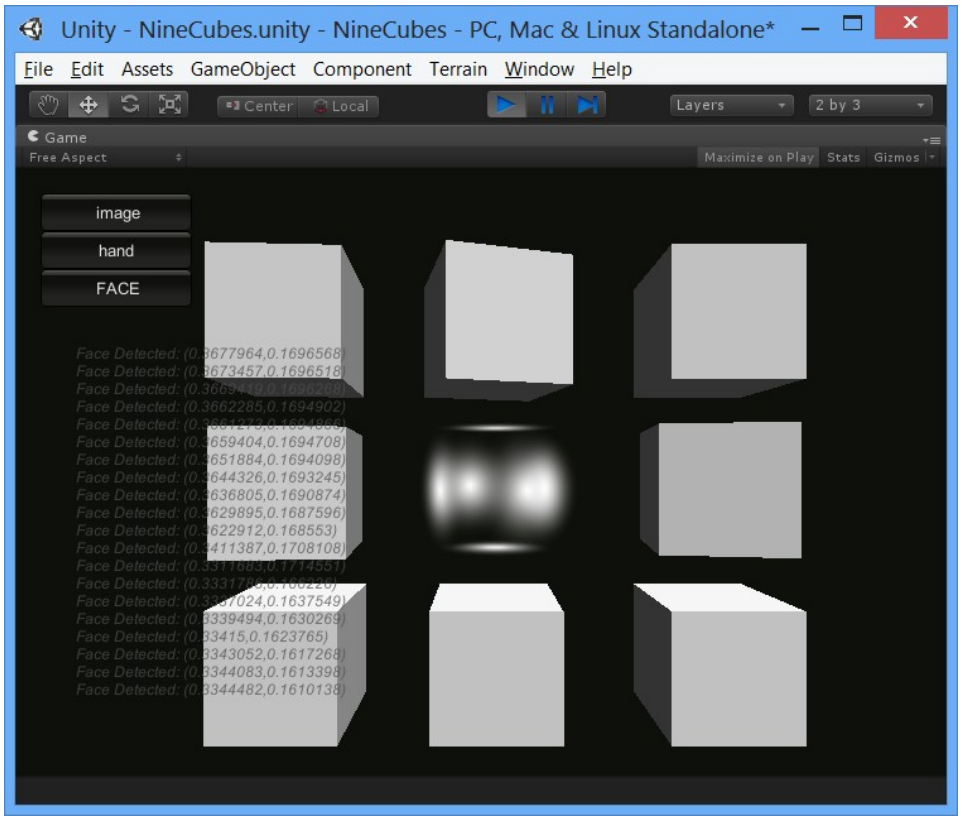


Figure 53: The NIneCubes Sample Window

If you see a warning message "`NullReferenceException: Object reference not set to an instance of an object UnityEditor.DockArea.OnGUI()`" while running your application in Unity Editor, close the `Game` and `Scene` windows and redo the window layout. The warning message should go away after the layout change. See also http://neareal.net/index.php?ComputerGraphics%2FUnity%2FTips%2FError%20UnityEditor.DockArea.OnGui() (In Japanese.)

4.9 Unity Sample: RF_ScenePerception

This sample `RF_ScenePerception` shows how to run the SDK scene perception module with the Unity game engine supporting the PC Standalone platform. The sample requires Unity v4.1.0 PRO (for the plugin capability) or later.

Copy the sample source to a writable folder and double click `Assets/Scenes/RF_ScenePerception.unity` to launch the Unity editor, and then run the game. The screen shot is similar to Figure 54.

The sample starts by displaying the color images with the frame rate at the top left and the scene quality at the bottom left. When the scene quality reaches certain threshold, the `Start` button appears on the control panel at the right. You can click the `Start` button to start scene perception or the `Quit` button to exit the sample.

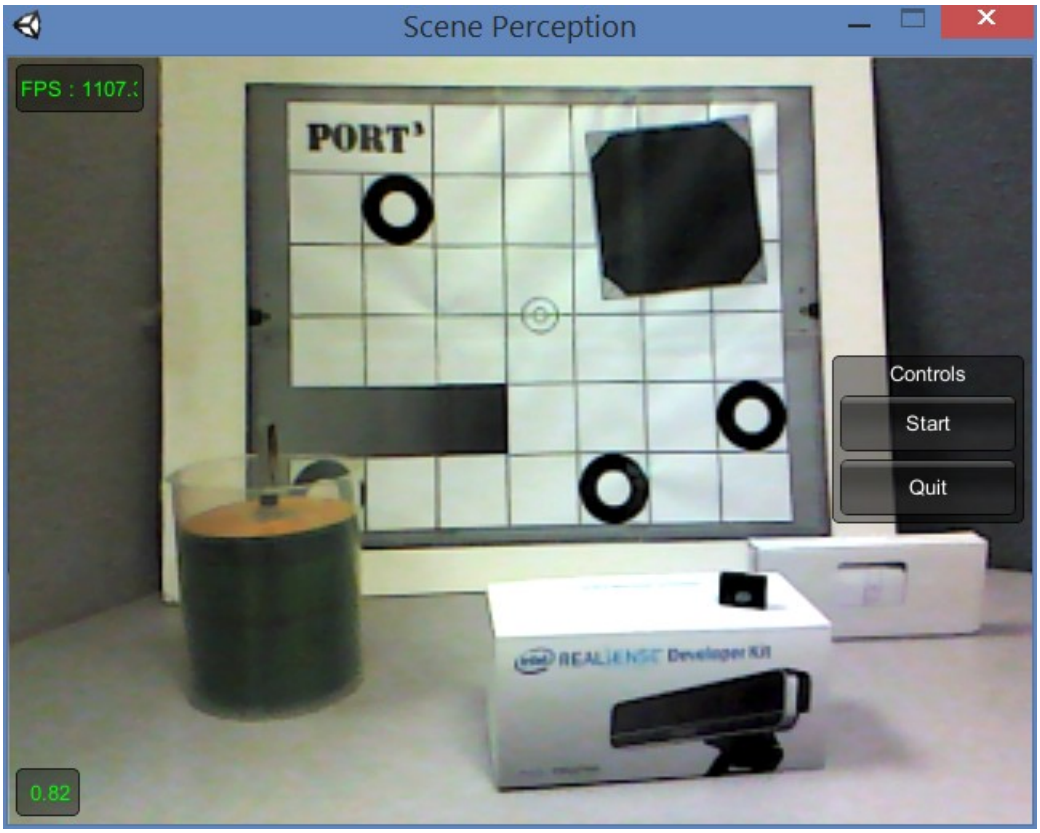


Figure 54: The Scene Perception Sample Window

The scene perception window is similar to Figure 55. The top left shows the rendering frame rate; the bottom left shows the tracking accuracy; and the right panels show available controls. You can enable/disable tracking, reconstruction and meshing by clicking the corresponding buttons. When meshing is enabled, you see the meshes in green on top of the color image display.

Click the `Save` Mesh button to save the meshes, and the `Reset` button to start over.

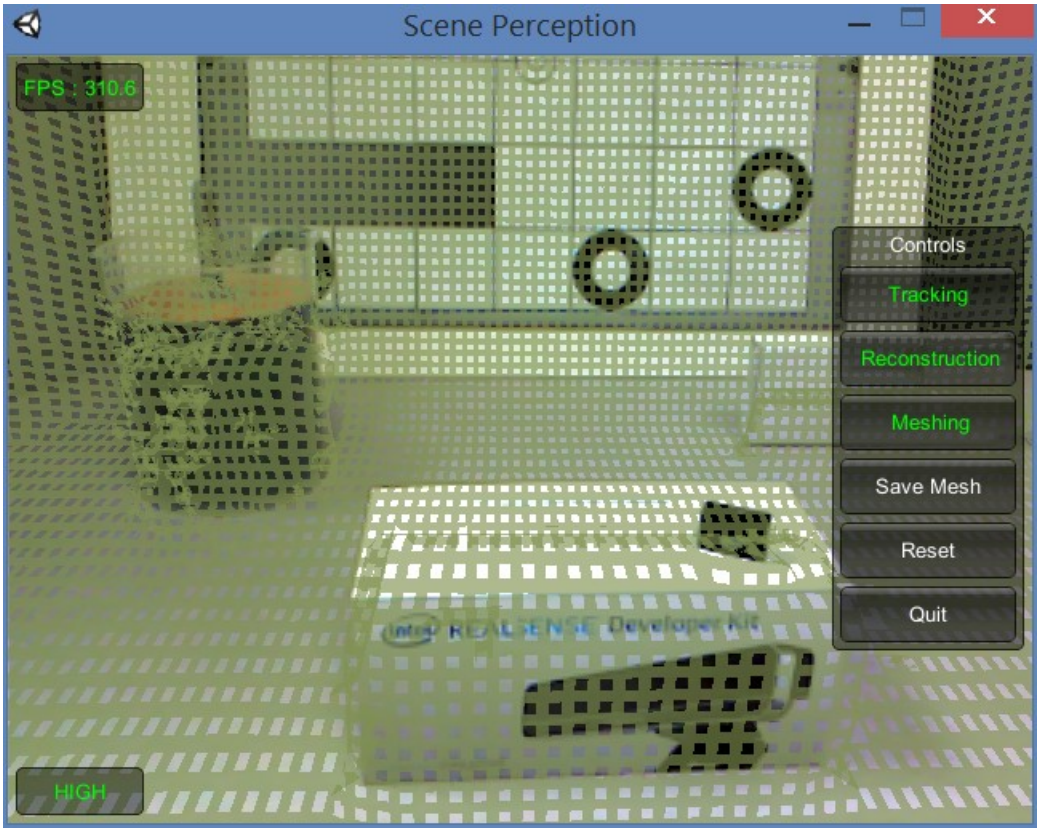


Figure 55: The Meshed Scene

The sample consists of a set of scripts, a meshing prefab and a set of rendering shaders as follows:

- The script `SenseInput.cs` uses the SDK SenseManager to initialize the camera, start streaming, and some setup/shutdown steps.
- The script `Options.cs` implements the menu selection as well as the initial scene quality checking screen.
- The script `DepthFusion.cs` implements the scene perception functions, calling member functions of the `SDK PXCMScenePerception` interface. When there are meshes to be displayed, the script pushes the meshes into a rendering queue and uses the `meshPrefab` prefab to visualize the meshes.

The meshing process takes a long time thus it must be done on a thread, while the mesh game object must be created in the Unity main thread. To avoid dropping frames, the Unity main thread updates about 200 mesh game objects every frame.

- The script `WorldFacingCamera.cs` updates the camera pose information that the scene perception module tracks.

If you see a warning message "`NullReferenceException: Object reference not`

`set to an instance of an object UnityEditor.DockArea.OnGUI()`" while running your application in Unity Editor, close the `Game` and `Scene` windows and redo the window layout. The warning message should go away after the layout change. See also http://neareal.net/index.php?ComputerGraphics%2FUnity%2FTips%2FError%20UnityEditor.DockArea.OnGui() (In Japanese.)

4.10 Unity Toolkit Samples

The SDK Unity Toolkit exposes the following samples:

Copy the sample source to a writable folder before running the sample in Unity Editor.

Sample1 – Translation

Instructions:

Hold your hand up and close it (make a fist) in order to control the yellow sphere.

Try to trip all the blue boxes.

Description:

This sample shows how to use the `Translate Action` to move around a `Game Object` (the sphere) and how to use the `Enable Behavior Action` and the `Disable Behavior Action` in order to light the sphere whenever an action is detected.

Sample2 - Rotation & Scale

Instructions:

Hold 2 hands up and close them (make a fist).

Move your hands apart and toward to each other to change the scale of the red cube.

Rotate your hands in a wheel motion to rotate the red cube around the Z axis.

Description:

This sample shows how to use the `Rotate Action` and `Scale Action` to rotate and scale a `Game Object` (the cube).

Sample3 - AR Mirror

Instructions:

Move your face around to see the mask moving with your face.

Hold up your hand to see the hand and finger tracking.

Description:

This sample shows how to create and Augmented Reality scene by using the prefabs of the Sense AR.

Sample4 - Falling Balls

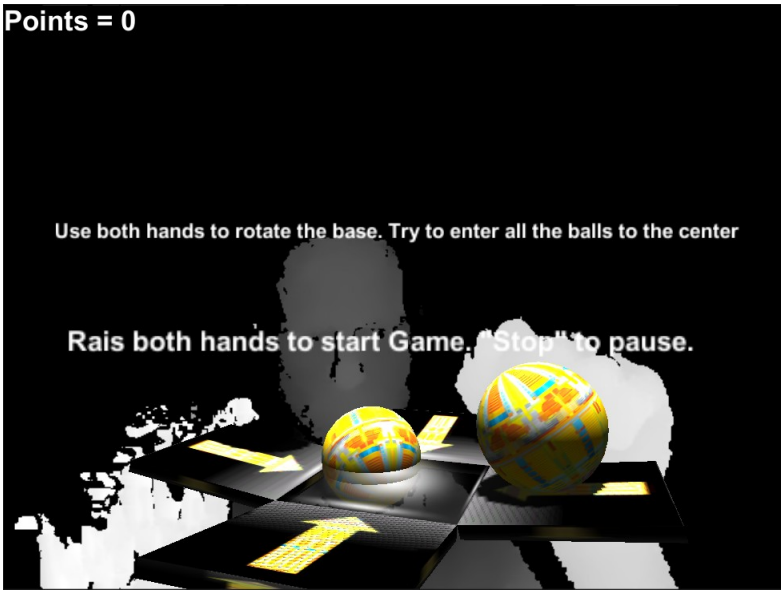
Instructions:

Use both hands to rotate the base. Try to enter all the balls to the center

Raise both hands to start Game. "`Stop`" to pause.

Description:

This sample shows how to create a mini-game using the `Rotate Action` together with freezing some axis



Sample5 - Send Message Mechanism

Instructions:

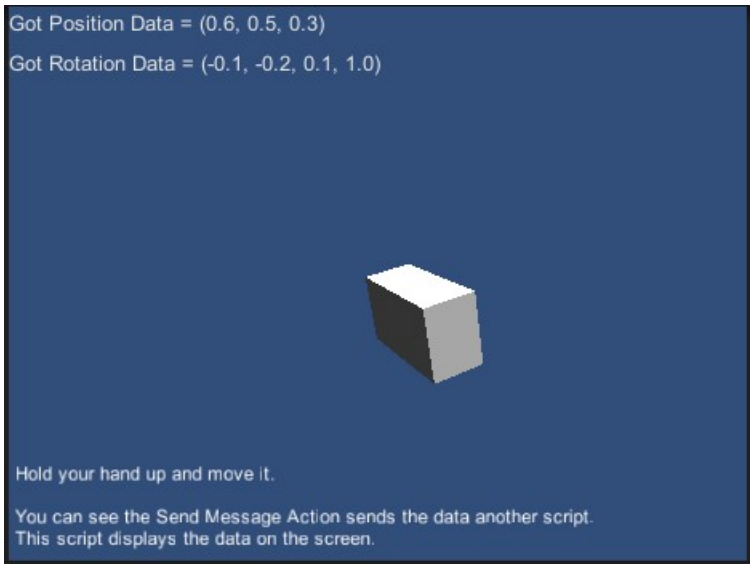
Hold your hand up and move it.

You can see the `Send Message Action` sends the data another script. This script displays the data on the screen.

Description:

This sample shows how to use the `Send Message Action` to send tracking data to another Mono-Behavior script.

By specifying the function name on the `Send Message Action`, this function will be called in the `Sample Trigger Registration` script.



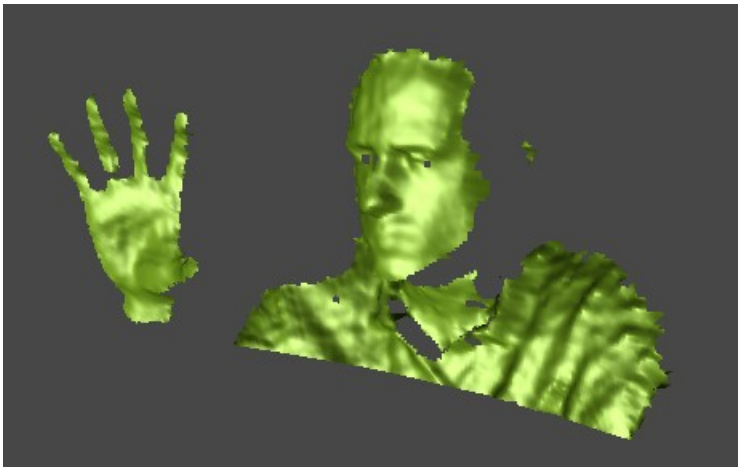
Sample6 - Point Cloud

Instructions:

Just run it and see the 3D data captured by the camera displayed on the scene.

Description:

This sample shows how to use the `PointCloudMesh` prefab to create a mesh dynamically from the depth data.



Sample7 - Object Tracking

Instructions:

Print out the image: `targetEarth.jpg` and present it to the camera. See how the objects follows the image.

Description:

This sample shows how to use the `Object Tracking` module.

You can specify different trackers (images) by changing the `Tracker 2DPath` variable.