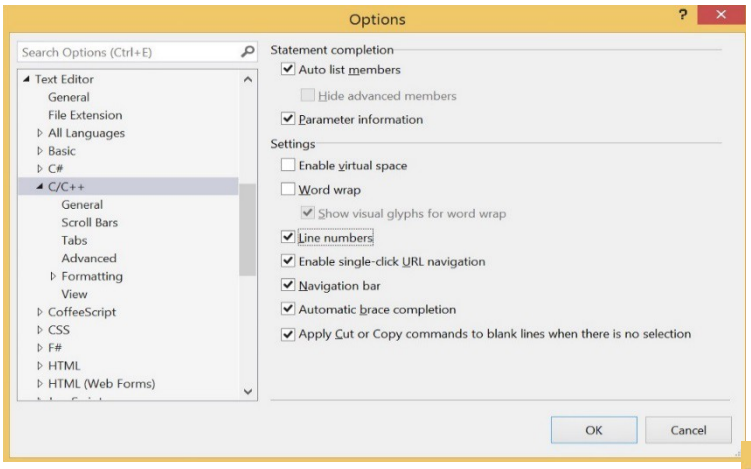
SFTL006: Creating Best-in-Class Intel® RealSense™ Applications

General Microsoft* Visual Studio* Tips

Permissions: Be sure to run as admin or restart Microsoft* Visual Studio* with elevated permissions.

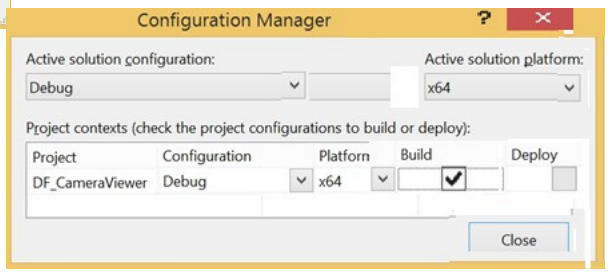


Line Numbers

If you don't see the line numbers:
From the Solution Explorer panel menu, select "Tools" -> "Options".
From the left list box,
choose "Text Editor",
then select "C/C++",
and make sure "Line numbers" is checked. Click OK.

To Configure the Build Target Platform, and Build the Solution

On the Menu Bar, select the "Build" menu -> "Configuration Manager". Make sure the fields are set as shown.



To Configure the Development Environment

C++ : To import property sheets, choose View → Other Windows → Property Manager. Right click on the project and choose Add Existing Property Sheet. They're in $(RSSDK_DIR)/props.
Use VS2010-13.Integration.MD.props for dynamic runtime compilation or
use VS2010-13.Integration.MT.props for static runtime.

C#. The SDK exposes 2 DLLs supporting .Net 4.0 : libpxcclr.cs.dll & libpxccpp2.dll
In the solution explorer, right click the project name and select Add Reference.
For C#: Add $(RSSDK_DIR)/bin/[win32 or x64]/libpxcclr.cs.dll
For C++ P/Invoke DLL: The app must manually copy libpxccpp2.dll to the local directory (if using Copy Local=True).

Lab 1: Intel® RealSense™ R200/F200 Raw Streams

This lab shows how to get raw streams from an Intel® RealSense™ Camera using the Camera Viewer module with either the user facing or world facing camera. You will use the Lab sample app (DF_ CameraViewer_vs2013.sln) and add these functions in the **camera_viewer.cpp** file (Use same file for F200 and R200 but add lines as specified per camera. For the solution, R200 and F200 Folders are separate and the folder you used should be renamed to src).

- Create SenseManager instance
- Configure the 3 streams (color, depth, IR). Later separate into 3 windows
- Initialize the pipeline
- Acquire and render each frame and clear for next (loops)

L1 Step-1: Open source camera_viewer.cpp

On the desktop, open IDF_RealSenseLab> Lab1_FR_RawStreams > DF_ CameraViewer_vs2013.sln

From the "Solution Explorer" panel, open src > camera_viewer.cpp.

L1 Step-2: Create Instance, Configure Streams, and Init the Pipeline

After Line 18, add code below as TODO 1

```
// create an instance of SenseManager
      PXCSenseManager *pp = PXCSenseManager::CreateInstance();
```

After Line 26, add code below as TODO 2

```
// Create stream renders
```

For F200 use:

```
     UtilRender renderc(L"Color"), renderd(L"Depth"), renderi(L"IR");
```

For R200 use:

```
      UtilRender renderc(L"Color"), renderd(L"Depth"), renderl(L"IR-Left"),
renderr(L"IR-Right");
```

After Line 30, add code below as TODO 3

```
// configure the components
```

for F200 use:

```
       pp->EnableStream(PXCCapture::STREAM_TYPE_COLOR |
       PXCCapture::STREAM_TYPE_DEPTH |      PXCCapture::STREAM_TYPE_IR, 0, 0);
```

For R200 use:

```
    pp->EnableStream(PXCCapture::STREAM_TYPE_COLOR | PXCCapture::STREAM_TYPE_DEPTH |

    PXCCapture::STREAM_TYPE_RIGHT | PXCCapture::STREAM_TYPE_LEFT, 0, 0);
```

L1 Step-3: Handle each frame

Starting at Line 45, aquire and render each frame (until loop is stopped) ***(TODO 4)***

```
/* Waits until new frame is available and locks it for application processing */
      sts=pp->AcquireFrame(false);
```

After Line 48 to set the color, depth and IR images in 3 separate windows, (***(TODO 5)***

```
/* Render the frame*/
      const PXCCapture::Sample *sample = pp->QuerySample();
      if (sample) {
         if (sample->color && !renderc.RenderFrame(sample->color)) break;
         if (sample->depth && !renderd.RenderFrame(sample->depth)) break;
```

Then add in TODO 6

```
For F200 add
         if (sample->ir    && !renderi.RenderFrame(sample->ir))    break;

        }
```

For R200 use:

```
      if (sample->left    && !renderr.RenderFrame(sample->left))    break;
      if (sample->right   && !renderl.RenderFrame(sample->right))    break;
      }
```

L1 Step-4: Graceful close

After Line 55, add in line as ***TODO 7***

```
// Release the frame so the pipeline can process next frame
      pp->ReleaseFrame();
```

After Line 66, BEFORE exiting, add in line as TODO 8

```
   // Release the (created SenseManager) Instance
      pp->Release();
```

Step-5: Press "Save" to save your code in CameraViewer.cpp.

Step-6: Build the solution.

Step-7: Run the App by pressing "Local Windows Debugger".

After the app starts, you should see 4 windows show up, respectively, for Windows command, Color stream, Depth Stream, and IR stream.

To exit, press Esc or q in the windows command window.

Lab 2: Intel® RealSense™ F200 Background Replacement Usage

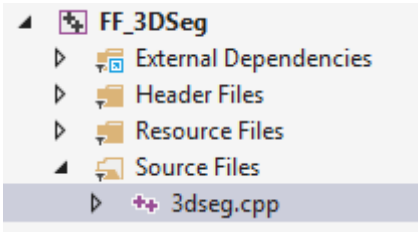
Use the background segmentation algorithm module to replace the video background with grayscale or a different picture. You will use Lab2a_F_3DSeg to do grayscale and then Lab2b for video background. Please use the separate 3dseg.cpp files in 2a or 2b to:

- Setup camera pipeline
- Initialize 3dSegmentation module
- Blend segmented image with greyscale background
- Replace grayscale with another background video (Lab 2b)

Lab 2a: Create Sample application with background replacement capability

L2 Step-1: Open 3dSegmentation source

On the Desktop, open IDF_RealSenseLab
→ Lab2a_F_3DSeg> FF_3DSeg_vs2013.sln
From the "Solution Explorer" panel, open src > 3dseg.cpp



L2 Step-2: Setup Camera pipeline

After line 26 (32) replace ToDo 1
```
// create a pipeline construct
PXCSenseManager* pSenseManager = PXCSenseManager::CreateInstance();
```

At Line 35 (41) replace ToDo 2
```
// Enable the User Segmentation video module
pxcStatus result = pSenseManager->Enable3DSeg();
```

At Line 42 (48) replace ToDo 3
```
//Query 3dSegmentation module
PXC3DSeg* pSeg = pSenseManager->Query3DSeg();
```

At Line 55 (62) replace ToDo 4
```
// Initialize and report the resulting stream source and profile to stdout
result = pSenseManager->Init();
```

Lab 2 Step-3: Acquire the segmented frame and blend with background

Line 105, ToDo 5
```
//Get the segmented image from PXC3dSeg Video Module
result =                              pSenseManager->AcquireFrame(true);
```

Line 167 (179) ToDo6
```
//Get the address of the row of pixels
pxcBYTE* p = segmented_image_data.planes[0]
      + y * segmented_image_data.pitches[0];
```

Line 182 (194) ToDo7 `//Blend the image with grey pixels`

```
Else for (int ch = 0; ch < 3; ch++) p[ch] = (p[ch] >> 4) + GREY;
```

Now Build and Run!

Lab 2B. Use 2B files, make same changes as above and then change background from greyscale to picture. USE 2B files

L2 Step-4: Open Lab 2b 3dseg.cpp

On the Desktop, open IDF_RealSenseLab → Lab2**b**_F_3DSeg -> FF_3DSeg_video.sln. You will modify the source file **3dseg.cpp from 2b**

L2 Step-5: Setup pipeline. Repeat Step 2 above, line numbers in (). Do NOT repeat Step 3

L2 Step-6:

Line 115, replace ToDo 8 //Open CV to read file

```
string filename = "test.mp4";
      VideoCapture cvCaptureFromFile(filename);
      if (!cvCaptureFromFile.isOpened())
            throw "Error when reading steam_avi";
      Mat framefile;
```

Line 139 replace ToDo 9 `//Acquire the image`

```
PXCImage* segmented_image = pSeg->AcquireSegmentedImage();
```

Line 209, replace ToDo 10 `//Read each RGB pixel from compressed MP4 frame`

```
PixelColor.val[0] = fPtr[y*frame.cols*channel + x*channel + 0]; // B
PixelColor.val[1] = fPtr[y*frame.cols*channel + x*channel + 1]; // G
PixelColor.val[2] = fPtr[y*frame.cols*channel + x*channel + 2]; // R
```

Line 223, replace ToDo 11 `//Blend RGB from video frame and segmented image`

```
      p[0] = PixelColor.val[2];
      p[1] = PixelColor.val[1];
      p[2] = PixelColor.val[0];
```

Lab 3: Intel® RealSense™ F200 Hand Tracking and Gesture

Use the hand module to recognize gestures and determine left or right hand. You will use Lab3a_F_Hands and Lab3b_F_Hands to complete the following functions in the Handviewer.cpp file

- Setup camera pipeline
- Initialize the hand module
- Detect and alert on recognized gesture (3a)
- Detect hand and alert the recognized gesture with hand information (3b)

Lab 3a: Create Sample application to recognize at least 3 different gestures

L3 Step-1: Open Hand Viewer source

On the Desktop, open IDF_RealSenseLab
→ Lab3a_F_Hands> FF_HandViewer_vs2013.sln
From the "Solution Explorer" panel, open src > Handviewer.cpp

L3 Step-2: Setup the Camera Pipeline

The function `void SimplePipeline(HWND hwndDlg)` starts at line 283 (299).
Fill in the following key lines of code:

Line 289: //Create Sense Manager

```
PXCSenseManager *pp = g_session->CreateSenseManager();
if(!pp)
{
    SetStatus(hwndDlg,L"Failed to create SenseManager");
    return;
}
```

Line 303: //Enable Hand Module

```
    pxcStatus status = pp->EnableHand(0);
```

Line 306: After //Query Hand Module

```
    PXCHandModule *handAnalyzer=pp->QueryHand();
    if(handAnalyzer == NULL || status != pxcStatus::PXC_STATUS_NO_ERROR)
    {
        SetStatus(hwndDlg,L"Failed to pair the gesture module with I/O");
        Return
    }
```

(Lab 3 continued)

Line 317: `//Initialized the Pipeline`

```
pxcStatus result = pp->Init();
```

L3 Step-3: Enable Gesture

After Line 346: `//Disable all gesture for better performance`

```
config->DisableAllGestures();
```

After Line 350: //Enable specific gesture. Add gesture name from manual

```
config->EnableGesture(L"spreadfingers", true);
```

Lab 3b: Modify Gesture Recognition to detect which hand and specific gesture

Follow Steps 1 and 2 of Lab 3a EXCEPT use lab 3b files!!!!!!!

L3 Step-4:

Around Line 223 in Lab 3b file, in the Function display Gesture,
AFTER //Get hand data related to fired gesture , add

```
PXCHandData::IHand* handData;
```

Line 227, AFTER //Check body side

```
if(handData->QueryBodySide() == PXCHandData::BodySideType::BODY_SIDE_LEFT)
{
        gestureStatus += ",Left Hand Gesture: ";
```

AFTER `//Else if body side`

```
else if(handData->QueryBodySide() == PXCHandData::BodySideType::BODY_SIDE_RIGHT)
{
        gestureStatus += "  Right Hand Gesture: ";
```

L3 Step-5: Save file and build code

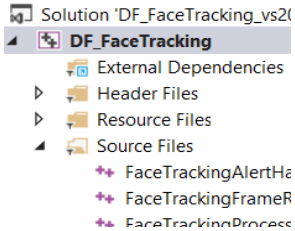
Lab 4: **Intel® RealSense™ F200 camera Face Tracking**

Use various functions of the face tracking algorithm module to detect and track the face and landmarks of the face and detect poses. You will use the Lab sample app (DF_FaceTracking_vs2013.sln) and fill in the following functions In **FaceTrackingProcessor.cpp**

- Setup the camera pipeline
- Configure Face Tracking Utilities (Face, Landmark and Pose Detection)
- Expression Recognition

Lab 4a:

L4 Step-1: Open FaceTracking source



On the Desktop, open IDF_RealSenseLab Lab4_F_FaceTracking\DF_FaceTracking_vs2013.sln
From the "Solution Explorer" panel, open src > **FaceTrackingProcessor.cpp**

L4 Step-2: Setup Camera pipeline

Read thru the function (CreateSenseManager) from line 66, start at `void FaceTrackingProcessor::Process(HWNDdialogWindow)`

Around Line 100, replace ToDo1 with

```
/* Set Module */
      senseManager->EnableFace();

/* Initialize */
      FaceTrackingUtilities::SetStatus(dialogWindow, L"Init Started",
statusPart);
```

Then add a QueryFace()

```
      PXCFaceModule* faceModule = senseManager->QueryFace();
      if (faceModule == NULL)
      {
            assert(faceModule);
            return;
      }
```

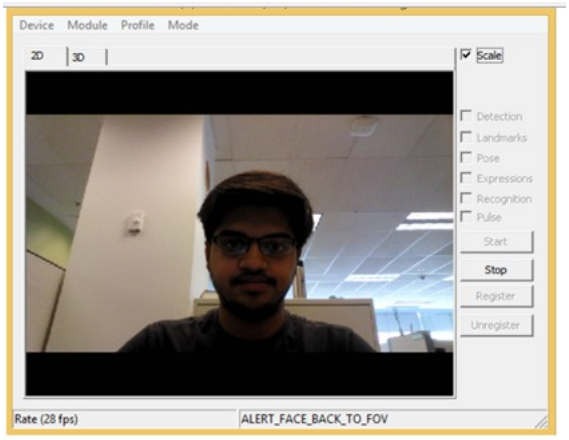
(Lab 4 continued)

Then Create the Active Configuration

```
PXCFaceConfiguration* config = faceModule->CreateActiveConfiguration();
if (config == NULL)
{
	assert(config);
	return;
}
```

Then Set the Tracking Mode

```
config->SetTrackingMode(FaceTrackingUtilities::GetCheckedProfile(dialogWindow));
config->ApplyChanges();
```



L4 Step-3: Run the code

Press ctrl + F5 to see the camera stream.

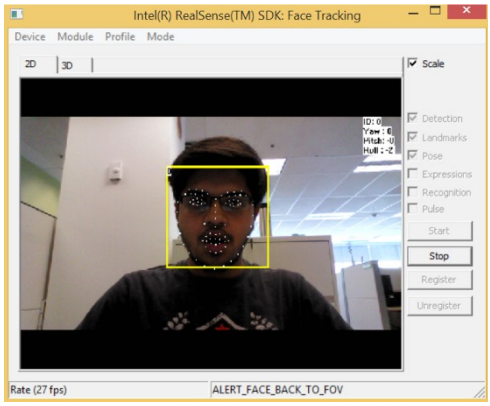
Lab 4, Task 2: Setup Face, Landmark, and Pose detection

```
147     CheckForDepthStream(senseManager, d
148     FaceTrackingAlertHandler alertHandl
149     if (FaceTrackingUtilities::GetCheck
150     {
151         /*
152
153         ToDo Task 2 : Enable Face , Lan
154
155         */
156         config->EnableAllAlerts();
157         config->SubscribeAlert(&alertHa
```

L4 Step-4: Around Line 171 (look for FaceTrackingAlertHandler) , replace ToDo 2 (not Todo 3) with the following code for location, landmark, and pose to enable all expressions.

```
      config->detection.isEnabled = FaceTrackingUtilities:
:IsModuleSelected(dialogWindow, IDC_LOCATION);
        config->landmarks.isEnabled = FaceTrackingUtilities:
:IsModuleSelected(dialogWindow, IDC_LANDMARK);
        config->pose.isEnabled = FaceTrackingUtilities:
:IsModuleSelected(dialogWindow, IDC_POSE);

(should resume existing code at ToDo 3 and then config->EnableAllAlerts();)
```

L4 Step-5: Hit ctrl+ F5 to run the code.
On the popup, check the Detection/Landmarks/Pose box on the right

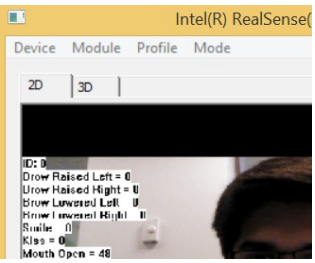


You can see that the face is detected in a yellow bounding box, various landmarks on the face are detected and painted in white and the pose information is displayed on the top right (Pitch, Yaw and roll information).

Lab 4, Task 3 : Expression Recognition

L4 Step-6: At the bottom of the step 5 section (but above `config->EnableAllAlerts()`, replace ToDo3 with

```
if (FaceTrackingUtilities::IsModuleSelected(dialogWindow, IDC_EXPRESSIONS))
{
    config->QueryExpressions()->Enable();
    config->QueryExpressions()->EnableAllExpressions();
}
```



L4 Step-7: Hit ctrl+ F5 to run the code.
On the popup, check Expressions and click Start:

In the example picture, the mouth is open and eyes are looking up. Note the left side of the picture box shows the values of the selected expressions.

Try putting your tongue out, closing an eye or even kissing to see the expressions being recognized!

Lab 5: Intel® RealSense™ R200 Enhanced Photography Module

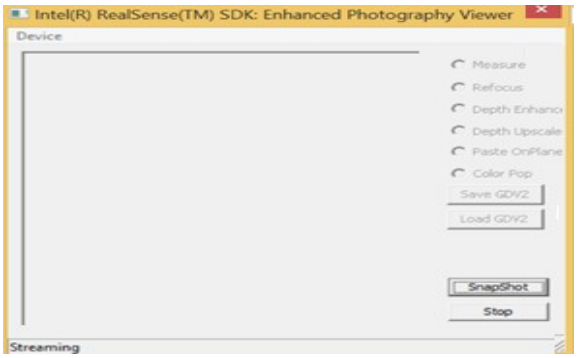
The R200 provides multiple enhanced photography algorithms. Use Lab5_R_EnhancedPhotography to perform the following functions in the RF_EnhancedPhotography.cpp file

- Measure the distance between two points
- Refocus based on Depth
- Paste a sticker on plane
- Pop a Color

Lab 5: Take a snapshot and measure distance

L5 Step-1: Open the EP Viewer

From the desktop, click on
IDF_RealSenseLab-> Lab5_R_EnhancedPhotography-> RF_EnhancedPhotography_vs2013.sln



raphy.cpp

Press Ctrl + F5 to run the code.

A blank EP Viewer will appear.
The bottom left status indicator will show ‘Streaming’ but the video window will be blank.

You need to write the code to display the camera stream.

L5 Step-2: Open the EP Viewer and read through pipeline creation

Open Source Files\RF_EnhancedPhotography.cpp

Scroll down to line 417,

This sets up the camera pipe line, enables the enhanced photo module and initializes the camera streams.

Lab 5 Step-3: Setup *Display*

At line 464,
the while loop queries a frame from the camera and needs to display it on the screen

```
464  while (!g_stop) { //streaming
465      SetStatus(hwndDlg,L"Strea
466      if (senseMgr->AcquireFram
           <PXC_STATUS_NO_ERROR) b
467
468      /* Check if a sample is a
469      // ToDo 1 :
470      if (sample) {
471          /* Display the color
472          if(sample->color) {
473      // ToDo 2 :
474              UpdatePanel(hwndD
```

On line 469, replace ToDo 1 with

```
sample = senseMgr->QuerySample();
```

On line 473, replace ToDo 2 with:

```
DrawBitmap(hwndDlg, sample->color);
```

L5 Step-4: Compile and check results

Press Ctrl + F5 to compile. Check for color streaming in the EP Viewer.

L5 Step-5: Measure the distance between two points.

Starting at Line #180, read through the function ProcessPhoto

```
187
188  /* If capture mode, process the photo *
189  while (!g_stop && IsSnapShot(hwndDlg))
```

The measurement feature is implemented in lines 202 to 212

```
201  // Measurement feature
202  if (IsMeasure(hwndDlg)) {
203      DrawBitmap(hwndDlg, photo->Qu
204      UpdatePanel(hwndDlg);
205      if (g_start.x>=0 && g_start.y
           g_end.y>=0 && !hold.Compare
206          // measurement calls
207          SetStatus(hwndDlg, L"Proc
208          pxcF32 distance = 0; //To
209          swprintf_s<sizeof(line)/s
             (line,L"Distance between
             mm",
210              g_start.x, g_start.y,
```

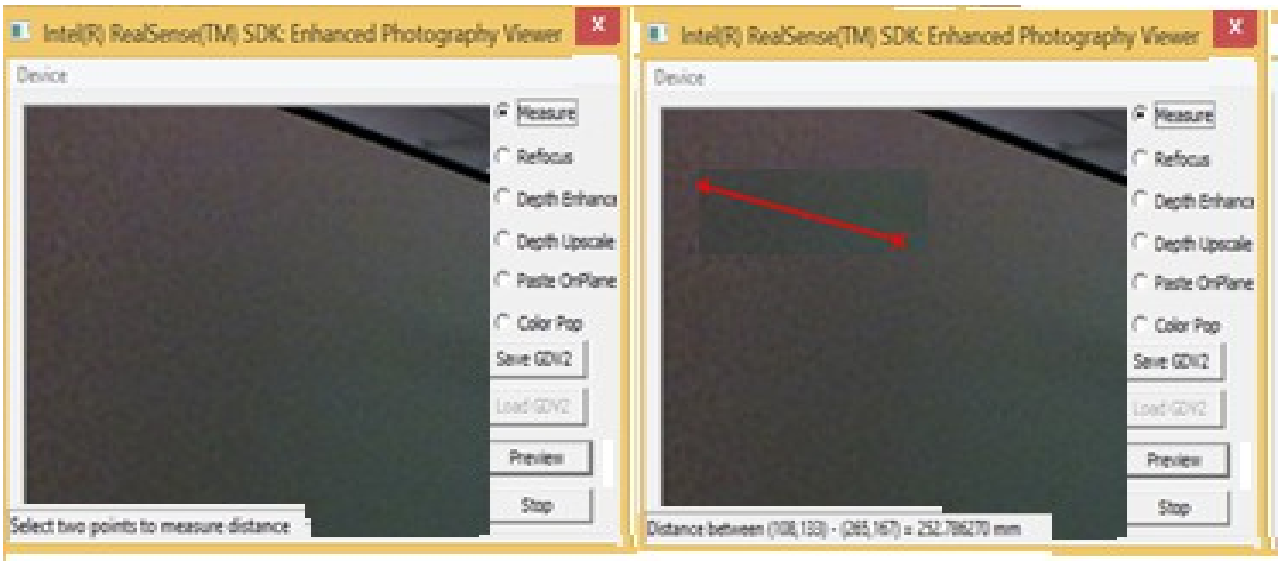
On Line 208,

```
pxcF32 distance = 0;
```

replace ToDo 3 (not whole line) with

```
distance = ep->MeasureDistance(photo,
g_start, g_end);
```

Run the code and take a picture of a flat surface about 1' away from the camera.
Choose the Measure radio button and select two points to measure the distance between them.


Intel(R) RealSense(TM) SDK: Enhanced Photography Viewer
Device
Measure
Refocus
Depth Enhance
Depth Upscale
Paste OnPlane
Color Pop
Save GDV2
Load GDV2
Preview
Stop
Select two points to measure distance
Intel(R) RealSense(TM) SDK: Enhanced Photography Viewer
Device
Measure
Refocus
Depth Enhance
Depth Upscale
Paste OnPlane
Color Pop
Save GDV2
Load GDV2
Preview
Stop
Distance between (108,133) - (265,167) = 252.786270 mm

Lab5 Step-6: Refocus based on depth

Depth refocus is implemented in
lines 217- 240

On line 224, replace ToDo 4 with:

```
photoOut = ep->DepthRefocus
(photo, g_start, 30);
```

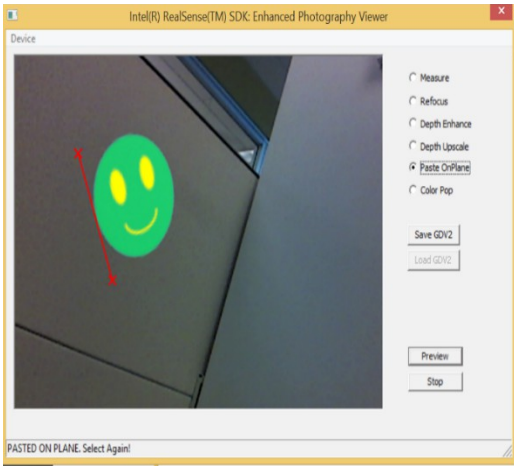
Run the code again.
Point it at an object and take a snapshot. Then choose the refocus radio button and select a point on the object to refocus it . Notice the change of focus.

L5 Step-7: Paste sticker on Plane

The Paste on plane feature is implemented in lines 243 - 266 .

~ line 252, replace ToDo 5 with

```
photoOut = ep->PasteOnPlane
(photo, sticker, g_start, g_end);
```



Run the code and take a snapshot of a flat surface.

Choose the paste on plane radio button

Lab5 Step-8: Color Pop

The Color Pop function is coded on lines 268-295

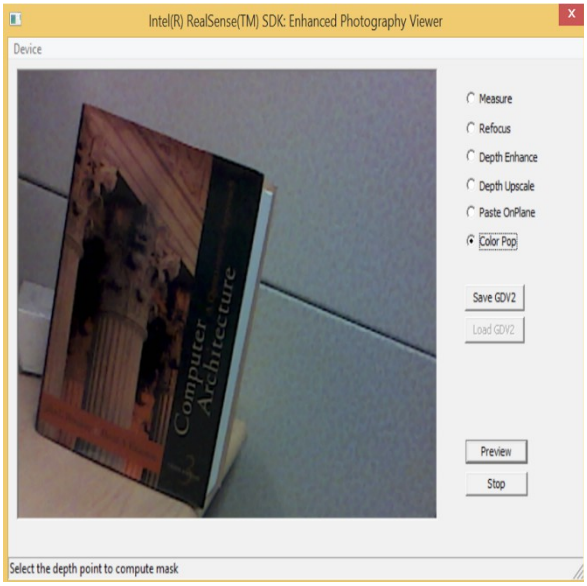
On line 273, replace the entire line 276

```
268    // 3 layer segmentation feature
269    if (IsColorPop(hwndDlg)) {
270
PXCImage *mask = NULL; //ToDo 6 :
271            // Application should release image wh
               no t needed
272            PXCImage *mask = NULL; //ToDo 6 :
PXCImage *mask = ep->ComputeMaskFromCoordinate(photo, g_start);
```

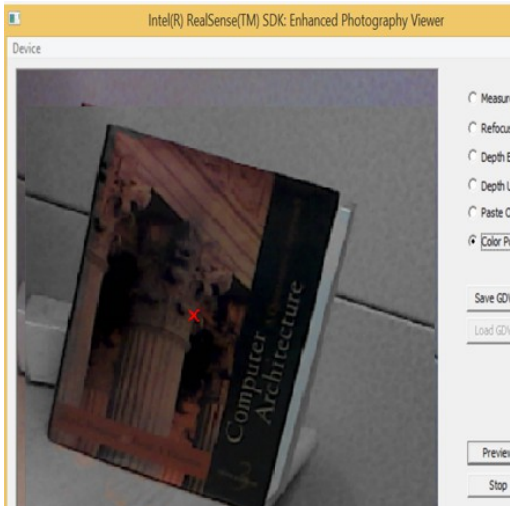
On line 282,
replace ToDo 7 with:

```
BlendColorPop(photoOut, mask);
```

I5 Step-9: Run the code and take a snapshot.



Select the color pop radio button and choose a pop point.



Lab 6: Intel® RealSense™ R200 Augmented Reality SP

Use the Scene Perception algorithm module to create an Augmented Reality sample app (RF_AugmentedRealitySP_vs2013.sln) with the following functions:

In sp_controller.h

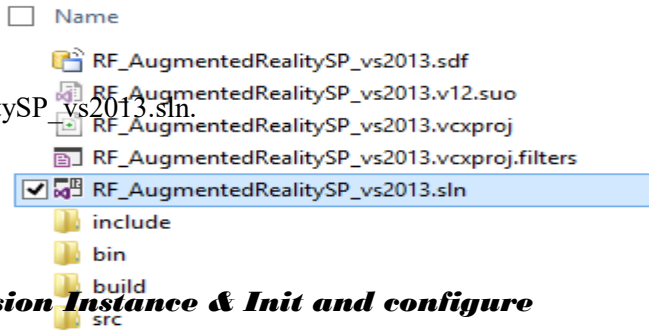
- Initialize and Configure Scene Perception module
- Query for Scene Perception Sample
- Retrieve color/depth images, camera pose, and tracking accuracy from sample

In AugmentedReality.cpp

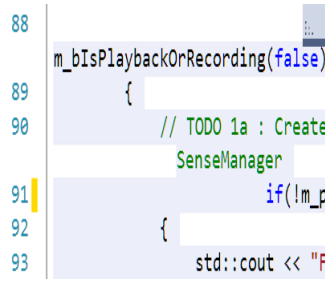
- Overlay objects onto the scene when localization Tracking Accuracy is acceptable
- Clean up resources by releasing the frame
- Experiment with the R200's real-time Localization/Tracking capabilities

L6 Step-1: Open the AR/SP Module

On the Desktop, open IDF_RealSenseLab
→ Lab5_R_AugmentedRealitySP → RF_AugmentedRealitySP_vs2013.sln.



Open \include\ **sp_controller.h**,



L6 Step-2: Create Session Instance & Init and configure the SP module

At Line 90, replace TODO 1a with

```
m_pSession = PXCSession::CreateInstance();
```

At Line 108, replace TODO 1b with

pxcStatus sts = `m_pSenseMgr->EnableScenePerception();`

L6 Step-3: Query for ScenePerception Sample

At Line 247, in the *ProcessNextFrame* function, replace TODO 2 with

```
auto pxcStatus = m_pSenseMgr->AcquireFrame(true);
```

Lab 6 Step-4 Retrieve color/depth images, camera pose, and tracking accuracy

At Line 277, (still in ProcessNextFrame function), replace TODO 3 with

```
if (color) *color = sample->color;
if (depth) *depth = sample->depth;
```

L6 Step-5 Overlay objects onto the Scene

Open the \src\AugmentedReality.cpp file

On line 505, in clbDisplay function, replace TODO 4 by calling the function

```
drawObjects();
```

L6 Step-6 Release the Frame and Clean Up Resources

On line 556, (still in the clbDisplay function), replace TODO 5 by calling the function

```
pScenePerceptionController->CleanupFrame();
```

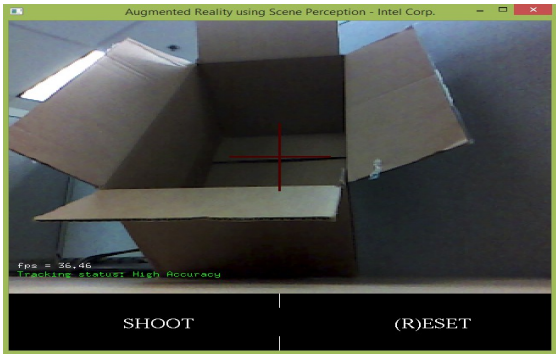
```
Shoot Button is only ava
 15 - BTN_HEIGHT * 2, di

releasing the frame
```

L6 Step-7 Experiment

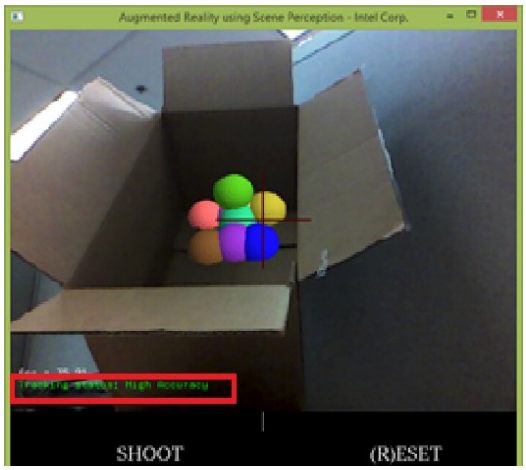
Press F5 to build the sample app and run it.

Point at something (that has texture) which is 1-2m (3-6') away from the camera. (sample uses cardboard box). When the "Tracking status" shows "High Accuracy", the SHOOT button will appear. (Requires successful track for buttons to appear.)

Click the shoot button to overlay the ball objects on the object on the screen.



(Lab 7 Task 3 Conclusion)

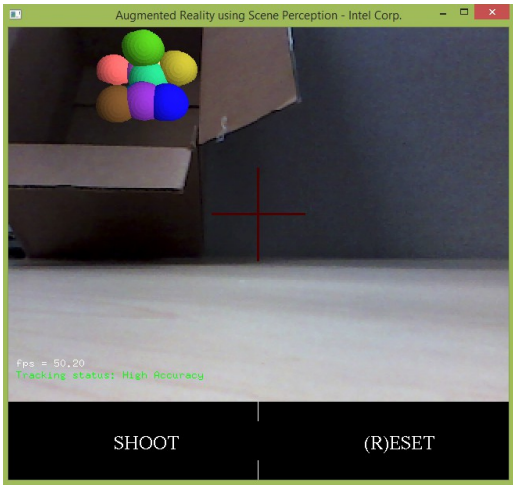


The Scene Perception module keeps track, in realtime, of the camera pose, and the localization data.

To verify that the R200 Scene Perception Module is accurately tracking the camera pose, try moving the camera closer, further or to the side.

Observe the tracking accuracy as the balls stay in the overlay location on the object.

You can hit the RESET button to start-over as needed.



Legal Notices and Disclaimers