Intel® RealSense™ 3D Camera

Outline

- Introduction to Depth Measurement Technology
- Introduction to Intel® RealSense™ Cameras
- Development of Intel® RealSense™ F200 Program

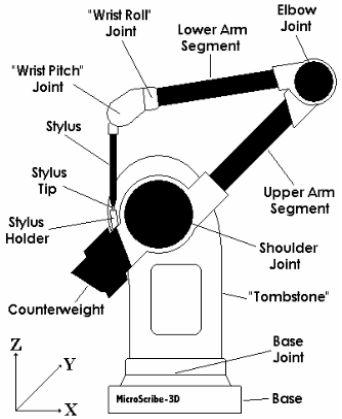
Introduction to Depth Measurement Technology

- Contact
- Non-Contact
  - Active
  - Passive

Contact Depth Measurement Technology

- 定義
  - 透過實際觸碰物體表面的方式計算深度
- Advantage
  - 相當精確, 常被用於工程製造產業
- Disadvantage
  - 在掃描過程中必須接觸物體, 待測物有遭到探針破壞損毀之可能, 因此不適用於高價值物件如古文物、遺跡等的重建作業
  - 相較於其他方法需要較長的時間
    - 現今最快的 CMM 每秒能完成數百次測量
    - 光學技術如雷射掃描儀運作頻率則高達 1萬次/秒 ~500萬次/秒

Coordinate Measuring Machine (CMM)



MicroScribe

http://www.mcneelmiami.com/home
http://www.remtek.com/remtek/ProEngineer/pro_scribe_win.htm
http://www.3d-microscribe.com/MX%20and%20G2%20Intro%20Page.htm

MicroScribe

https://www.youtube.com/watch?v=lvTU0ykBKfc&list=PLHxbo0NT14nAylXwzbRfD_1CqMr7lO8Iv
https://www.youtube.com/watch?v=_vddldNtu_8

Non-Contact Depth Measurement Technology

- Active
  - 將額外的能量投射至物體，藉由能量的反射來計算三維空間資訊
  - 常見的投射能量有一般的可見光、高能光束、超音波與 X 射線

- Passive
  - 不發射任何輻射線(ex: 雷射), 而是以測量由待測物表面反射周遭輻射線的方法，達到預期的效果
  - 由於環境中的可見光輻射，是相當容易取得並利用的, 大部分這類型的掃描儀以偵測環境的可見光為主
  - 相對於可見光的其他輻射線，如紅外線，也是能被應用於這項用途的
  - 因為大部分情況下，並不需要規格太特殊的硬體支援, 這類產品往往相當便宜

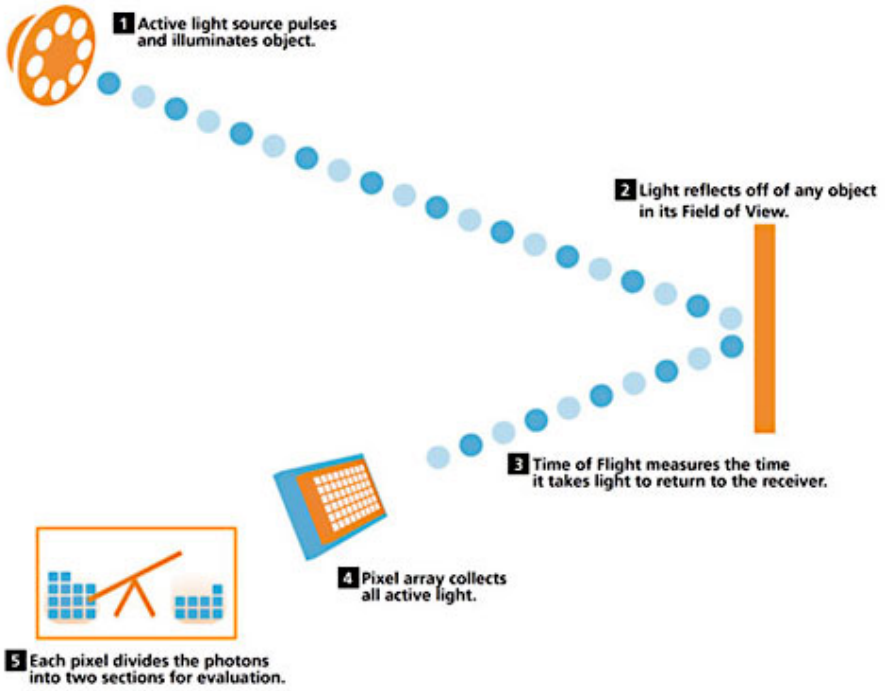
LIght Detection And Ranging, Lidar

Non-Contact Depth Measurement Technology

- Active
  - Time-of-Flight
  - Triangulation
- Passive
  - Stereoscopic
  - Photometric
  - Silhouette

Time-of-Flight

- 定義
  - 儀器發射一個雷射光脈衝，雷射光打到物體表面後反射，再由儀器內的探測器接收訊號，並記錄時間
- 原理
  - 由於光速 c 為一已知條件，光訊號往返一趟的時間即可換算為訊號所行走的距離，此距離又為儀器到物體表面距離的兩倍，故若令 t 為光訊號往返一趟的時間，則光訊號行走的距離等於 (c*t)/2
- 其量測精度受到能多準確地量測時間 t，因為大約 3.3 picosecond 的時間，光訊號就走了1公釐



Creative Senz3D
(Intel)

Kinect 2
(Microsoft)

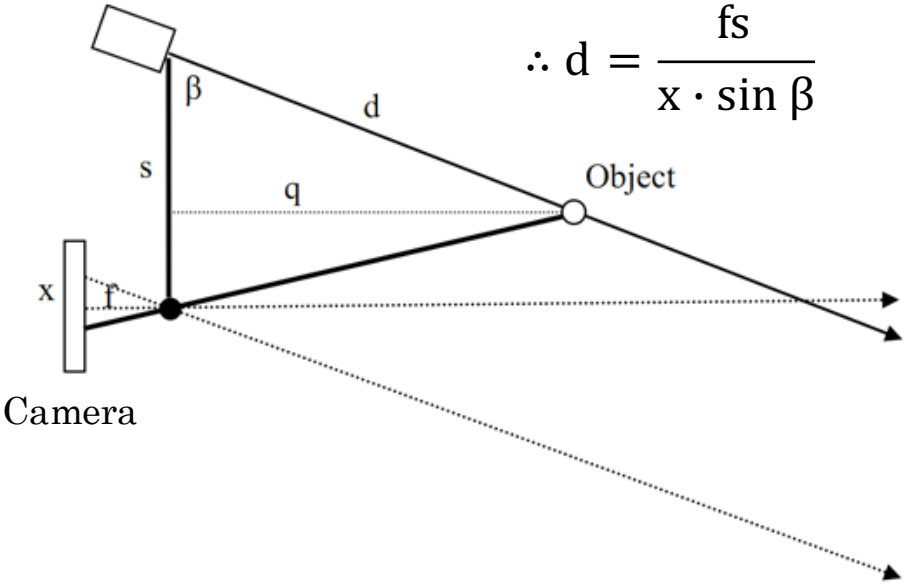
Triangulation

- 定義
  - 發射一道雷射到待測物上，並利用 camera 尋找待測物上的 laser dot
- 隨著待測物距離的不同，laser dot 在 camera 畫面中的位置亦有所不同
- 被稱為 Triangulation，是因為 laser projector, laser dot 與 camera 構成一個三角形
- 原理
  - 已知條件
    - camera 的焦距 f
    - laser projector 與 camera 的距離 s
    - laser projector 在三角形中的角度 β
  - 透過測量 x, 即可得到 laser projector 到待測物的距離 d
  - x 是 laser dot 在 camera 感光元件(如CMOS)上的成像到一側邊緣的距離

$$\frac{f}{x} = \frac{q}{s} \Rightarrow q = \frac{fs}{x}$$

$$\sin\beta = \frac{q}{d} \Rightarrow d = \frac{q}{\sin\beta}$$

$$\therefore d = \frac{fs}{x \cdot \sin\beta}$$



Kinect 1 (Microsoft)

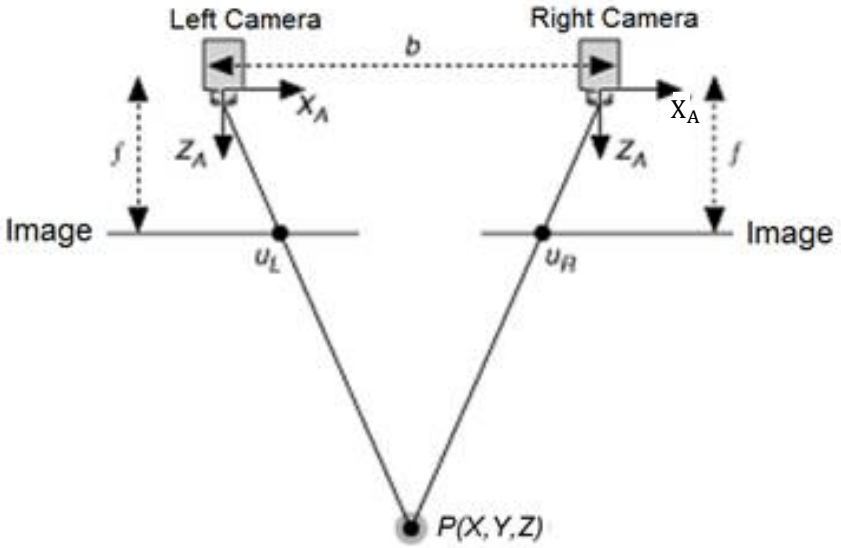
Xtion Pro (ASUS)

Xtion Pro Live (ASUS)

http://www.csksoft.net/blog/post/lowcost_3d_laser_ranger_1.html

Stereoscopic

- 定義
  - 使用兩個放在一起的攝影機，平行注視待測物
- 此方法在概念上，類似人類藉由雙眼感知的影像相疊推算深度
  - 當然實際上人腦對深度資訊的感知歷程複雜許多
- 若已知兩個攝影機的彼此間距與焦距長度，而擷取的左右兩張圖片又能成功疊合，則深度資訊可迅速推得
- 原理
  - 已知條件
    - 左, 右 camera 的距離 b
  - 透過測量 $U_L - U_R$, 即可得到 camera 到待測物的距離 Z
    - $U_L - U_R$ 是 P 點在左, 右 camera 上的成像距離差 (disparity)



$$\begin{cases} \dfrac{f}{U_L} = \dfrac{Z}{X} \Rightarrow U_L = f\dfrac{X}{Z} \\ \dfrac{f}{U_R} = \dfrac{Z}{-(b-X)} \Rightarrow U_R = f\dfrac{X-b}{Z} \end{cases}$$

$$\Rightarrow U_L - U_R = \frac{b}{Z} \Rightarrow Z = \frac{b}{U_L - U_R}$$

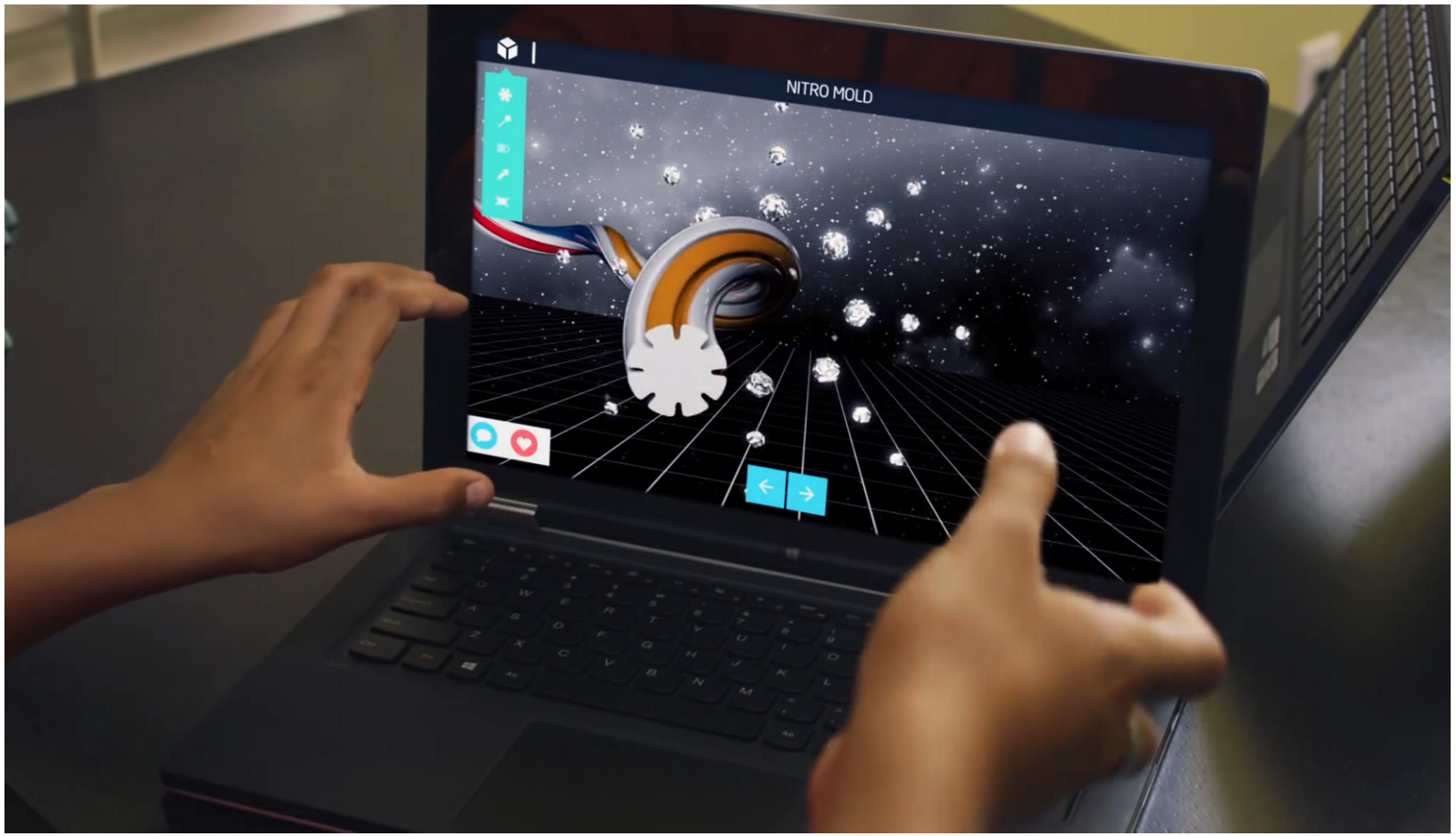
Bumblebee2 (Point Grey)

http://www.ni.com/white-paper/14103/zht/
http://www.ptgrey.com/bumblebee2-firewire-stereo-vision-camera-systems

Depth Measurement Cameras

Depth Measurement Technology			Camera
Contact			Coordinate Measuring Machine (CMM)
			MicroScribe
Non-Contact	Active	Time-of-Flight	Creative Senz3D
			Kinect 2
		Triangulation	Kinect 1
			Xtion Pro, Xtion Pro Live
	Passive	Stereoscopic	Bumblebee2

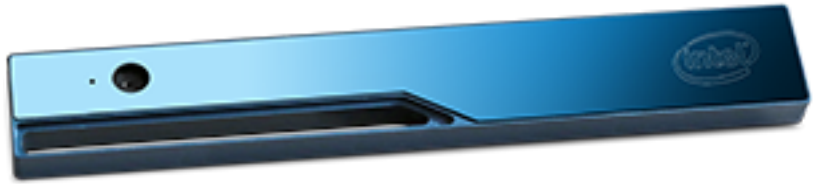
Introduction to Intel® RealSense™ Cameras



https://www.youtube.com/watch?v=Vuyd8AQQ9Mw

Intel® RealSense™ 3D Camera

RealSense (F200)

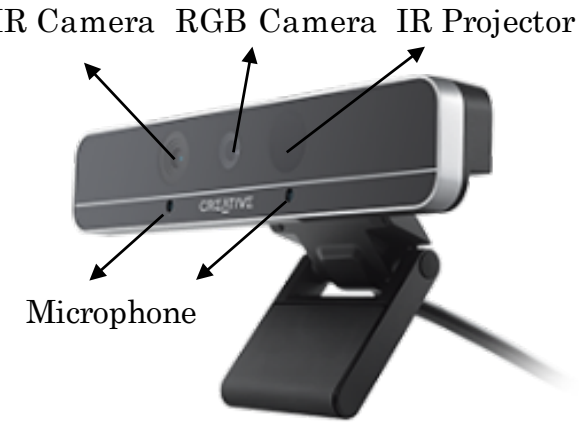


RealSense (R200)

Intel® RealSense™ 3D Camera (F200)

- 由 Intel 開發
- 主要是要內嵌在筆電, 2-in-1 或 All-in-One 電腦上
  - ASUS ROG G771JM → Laptop
  - Lenovo B50 Touch → All-in-One
  - … (11 more) acer hp FUJITSU DELL
    https://www-ssl.intel.com/content/www/tw/zh/architecture-and-technology/realsense-devices.html?_ga=1.65011631.1072206394.1430312390
- 軟體驅動:
  Intel® RealSense™ SDK (shared with SR300, R200)

Creative Senz3D
(1st generation)



F200 Developer Kit
(2nd generation)

(目前尚在預購階段)

SR300 Developer Kit
(2nd generation of F200)

What's New in SR300 ?

- Increased range and lateral speed.
- Improved color quality under low-light performance conditions.
- Improved color and depth stream synchronization.
- Decreased power consumption.
- …

https://software.intel.com/sites/landingpage/realsense/camera-sdk/v1.1/documentation/html/index.html?doc_mgsr300_working_with_sr300.html

Intel® RealSense™ 3D Camera (F200) Specification

Module Dimensions	150 mm x 30 mm x 58 mm
Interface	**USB 3.0 (5 V*)**
Sensor	Color, Depth, IR Camera Microphone*2
Sensor Resolution	Color: Up to 1080p at 30 fps (FHD) Depth: Up to 640x480 at 60 fps (VGA), HVGA at 110 fps IR: Up to 640x480 at 300 fps
Depth Effective Range	**0.2 m ~ 1.2 m***
Depth Effective Environment	Indoor and Indirect Sunlight (Up to 1000 Lux)
Platform	**4th generation (or later) Intel® Core™ processor**
OS Support	**Microsoft Windows 8.1* (or later) (64-bit)**
Software	Intel® RealSense™ SDK (shared with SR300, R200)

※ The Voltage of USB 3.0 should be high enough
※ Range and accuracy may vary based on targeted software usage.

Intel® RealSense™ 3D Camera (SR300) Specification

Module Dimensions	110 mm x 12.6 mm x 3.8 mm
Interface	USB 3.0
Sensor	Color, Depth, IR Camera Microphone*2
Sensor Resolution	Color: Up to 1080p at 30 fps (FHD), 720p at 60 fps Depth: Up to 640x480 at 60 fps (VGA), HVGA at 110 fps IR: Up to 640x480 at 200 fps
Depth Effective Range	**0.2 m ~ 2.0 m***
Depth Effective Environment	Indoor and Indirect Sunlight
Platform	**6th generation (or later) Intel® Core™ processor (recommended)**
OS Support	**Microsoft Windows 10 (or later) (64-bit)**
Software	Intel® RealSense™ SDK (shared with F200, R200)

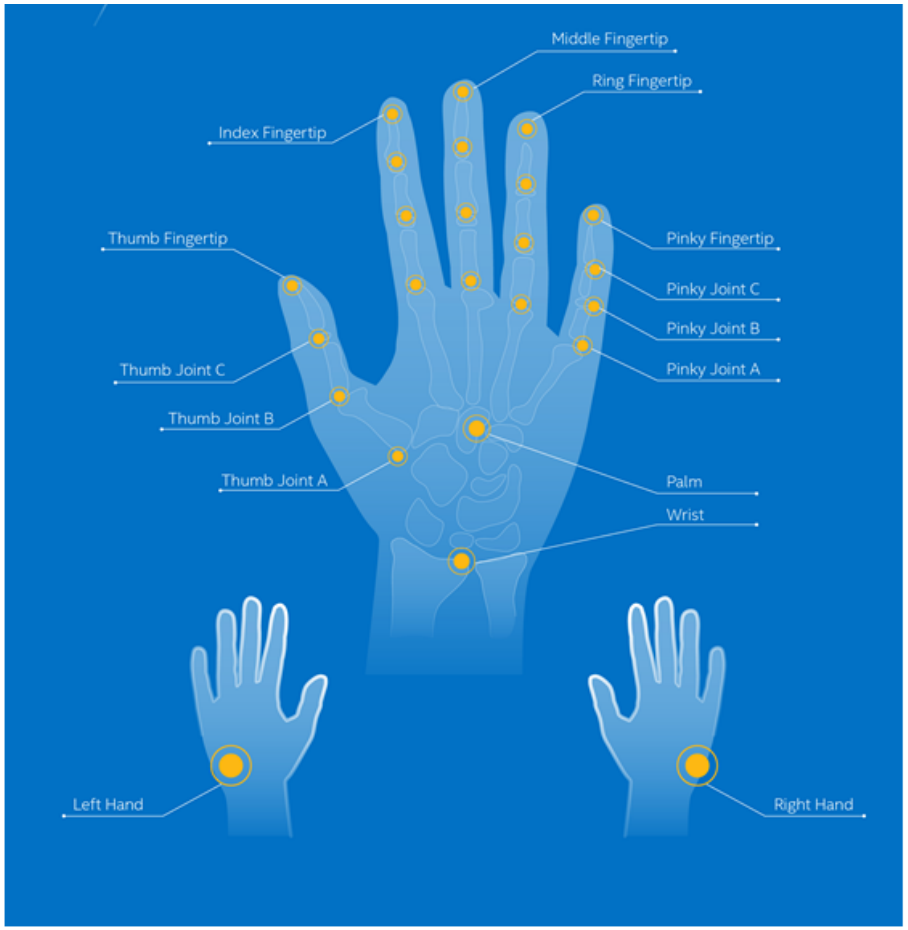
※ Range and accuracy may vary based on targeted software usage.

Intel® RealSense™ 3D Camera (F200) Features

- Hand
- Face
- Environment
- Speech

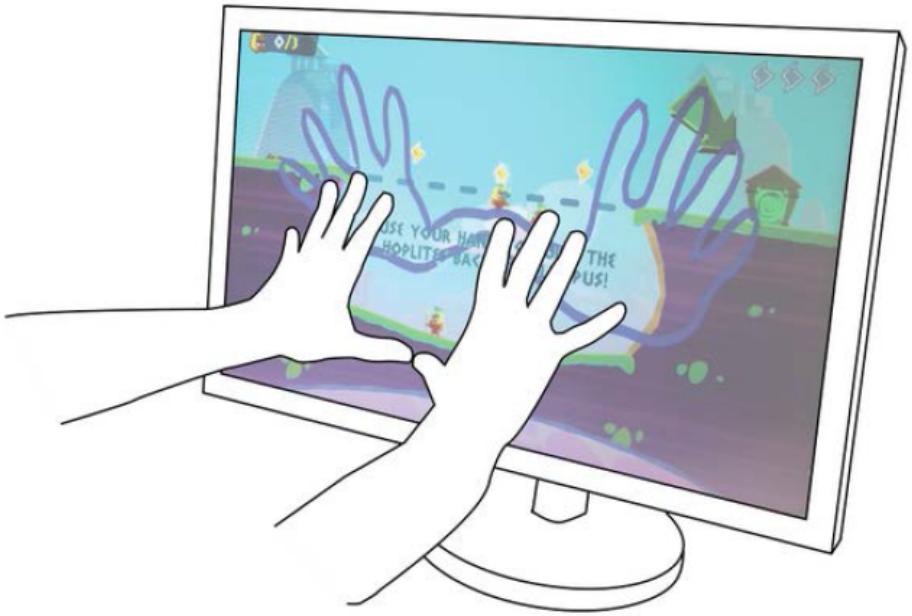
Intel® RealSense™ 3D Camera (F200) Features - Hand

- Hand and Finger Tracking
- 22 Joints (Skeleton)
- Left, Right Side Detection
- Static Poses and Dynamic Gestures Recognition



Intel® RealSense™ 3D Camera (F200) Features - Blob Detection

- Detect up to 4 blobs
- Points (Closest, Center, Top…)
- Convenient alternative to Hand Tracking
- Smaller impact in system resources

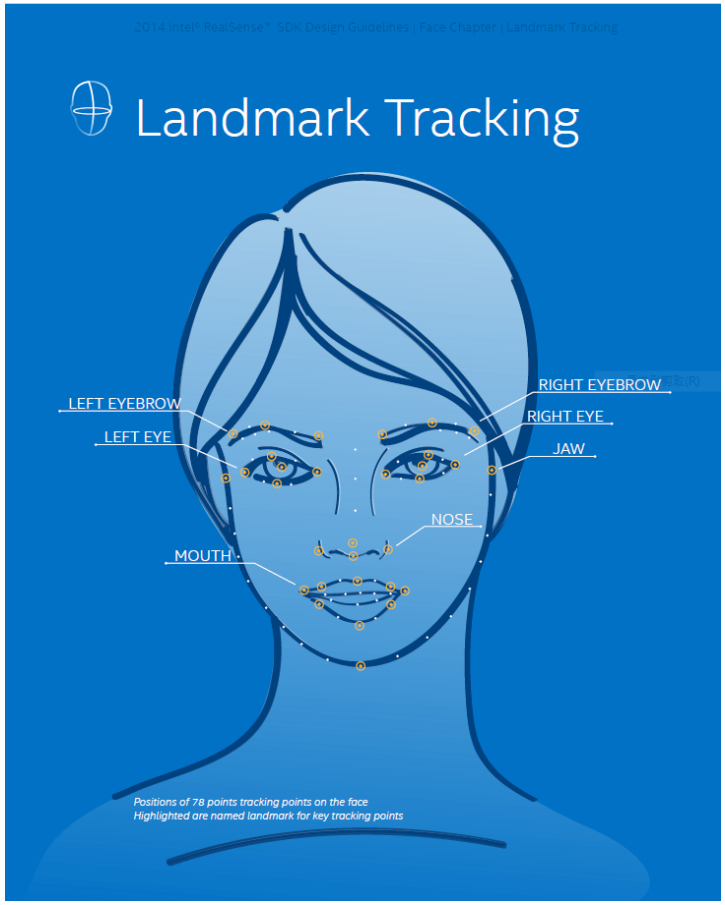


Intel® RealSense™ 3D Camera (F200) Features - Touchless Controller

- Allows the usage of hand gestures to control basic user interface (UI) functions
- Based on the hand tracking module
- Maps gestures to UI controls and generates system events

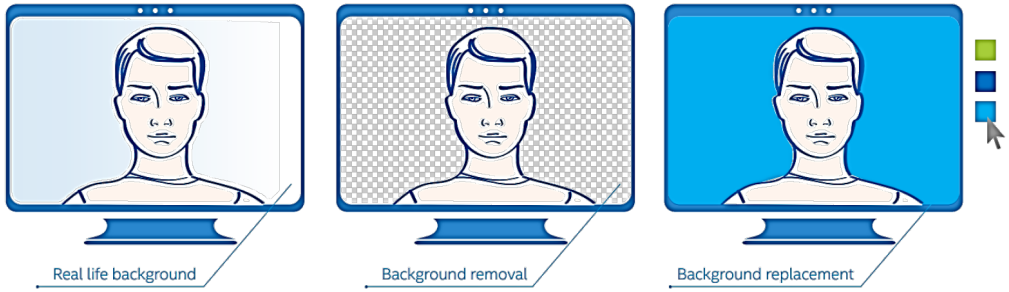
Intel® RealSense™ 3D Camera (F200) Features - Face

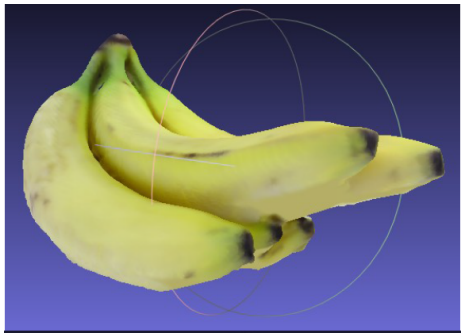
- Multiple Face Detection and tracking
- 78-point Landmark Detection (facial features)
- Face Recognition and Facial Expressions
- Emotion Detection
- Pulse Estimator



Intel® RealSense™ 3D Camera (F200) Features - Environment

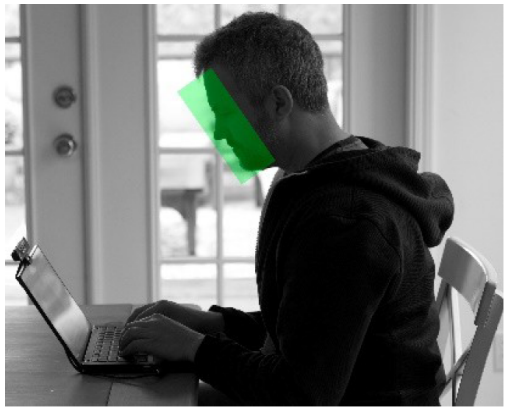
- 3D Background Segmentation
- 3D Object / Face / Room Scanning
- 2D/3D Object Tracking



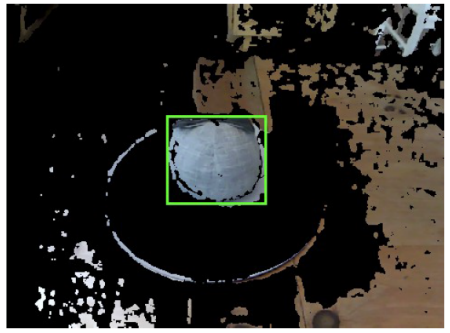


Intel® RealSense™ 3D Camera (F200) Features - 3D Scan

- Face
- Object



Face



Object

Intel® RealSense™ 3D Camera (F200) Features - Speech

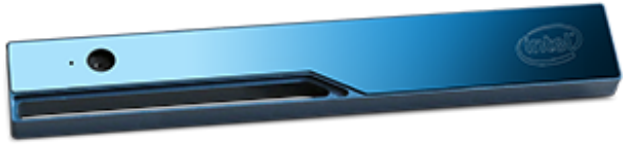
- Speech Recognition and Synthesis
  - Command and Control
  - Dictation
  - Text to Speech

F200 Application – 3DMe (by 3DSYSTEMS)

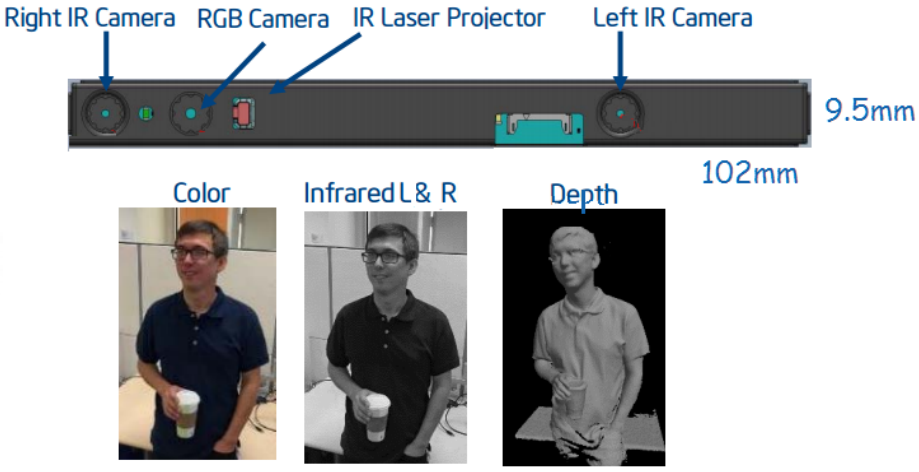
http://cubify.com/store/_3dme

Intel® RealSense™ 3D Camera (R200)

- 由 Intel 開發
- 主要是要內嵌在平板電腦，2-in-1 或 All-in-One 電腦上
- 軟體驅動:
  Intel® RealSense™ SDK (shared with F200, SR300)



R200 Developer Kit



https://software.intel.com/en-us/articles/realsense-r200-camera

Intel® RealSense™ 3D Camera (R200) Specification

Module Dimensions	130 mm x 20 mm x 7 mm
Interface	USB 3.0
Sensor	Color, Depth, IR Camera Microphone*2
Sensor Resolution	Color: 1080p at 30 fps (FHD) Depth/IR: Up to 640x480 at 60 fps (VGA)
Depth Effective Range	**0.51 ~ 4.0 m (indoor) ; longer range (outdoor)**
Depth Effective Environment	**Indoor / Outdoor**
Platform	4th generation (or later) Intel® Core™ processor
OS Support	**Microsoft Windows 8.1 (or later) (64-bit)** Android* (support coming soon)
Software	Intel® RealSense™ SDK (shared with F200, SR300)

R200 Application – Augmented Reality

- Augmented World Expo 2015

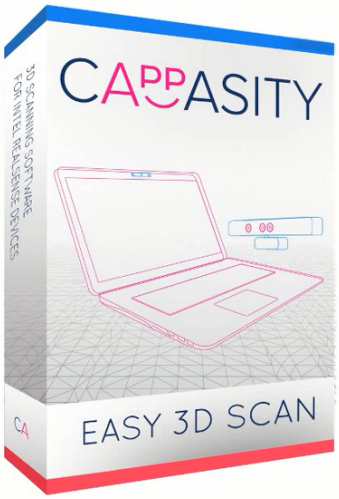
augmented world expo
awe2015

- This game makes of the two main features of the R200 camera:
  - Tracking/Localization:
    Real-time estimation of the Camera's position and orientation (pose) using depth, RGB and IMU data.
  - 3D Volume/Surface Reconstruction:
    Building real-time digital representation of the 3D Scene observed by the camera

https://www.youtube.com/watch?v=2BN_-2gCrqU

R200 Application – 3D Scan

- EASY 3D SCAN by Cappasity



Intel RealSense R200 3D Camera
Alpha Test
Mobile 3D scanning

https://www.youtube.com/watch?v=MwIuTmfyXCQ

Development of Intel® RealSense™ F200 Program

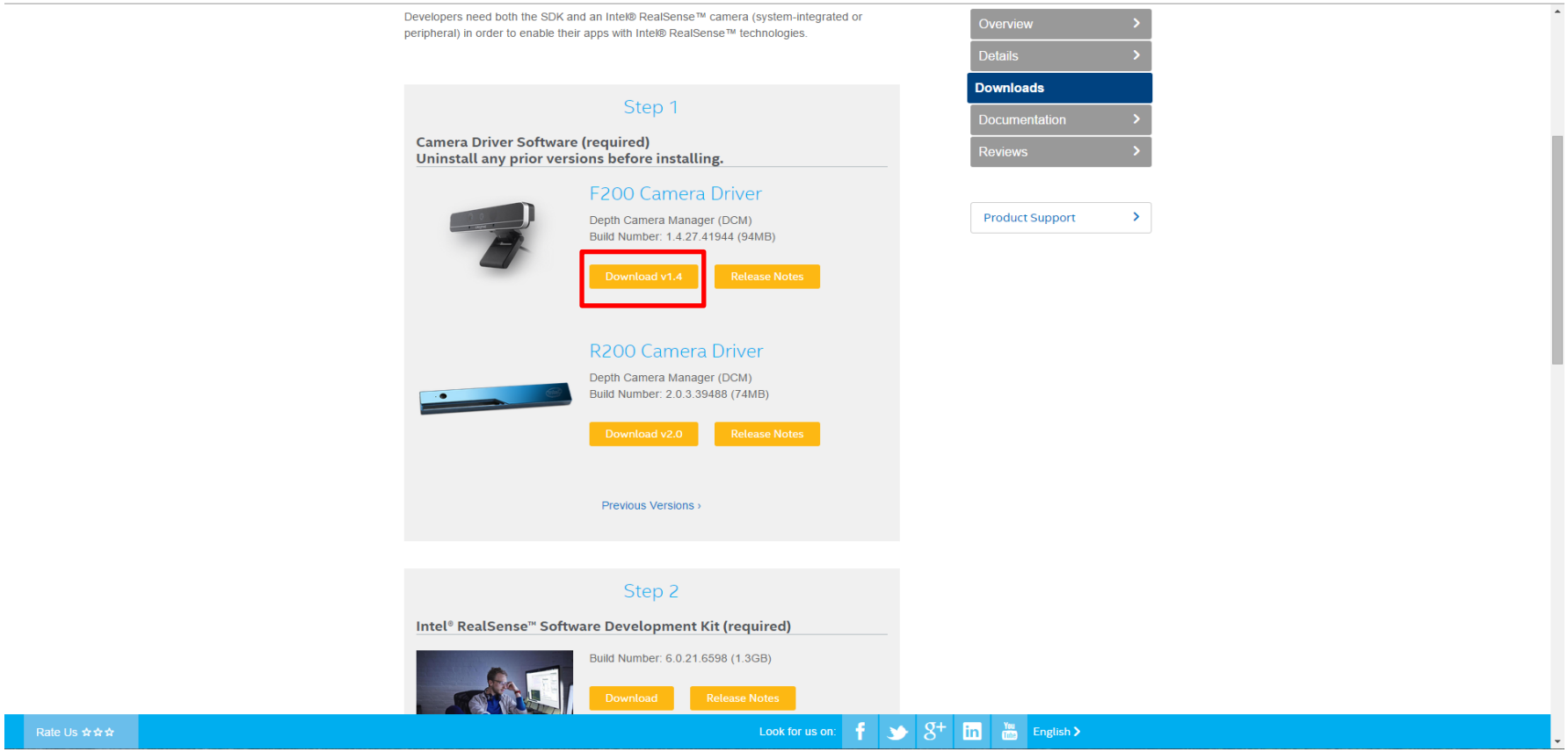
- Software Installation
- SDK Overview
- Two Examples

Software Installation

- Depth Camera Manager (DCM)
- Software Development Kit (SDK)

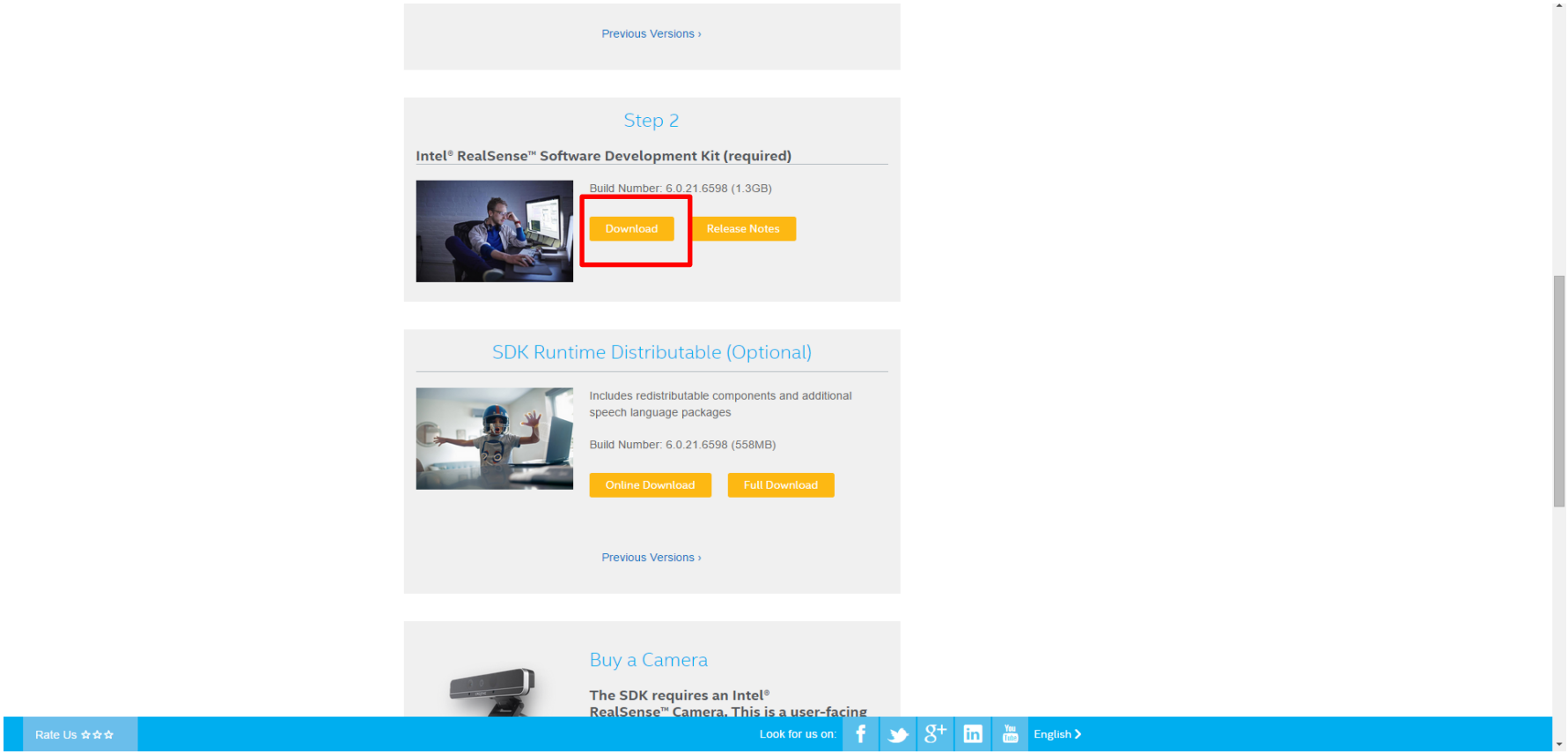
Depth Camera Manager (DCM) Downloading

- https://software.intel.com/en-us/intel-realsense-sdk/download



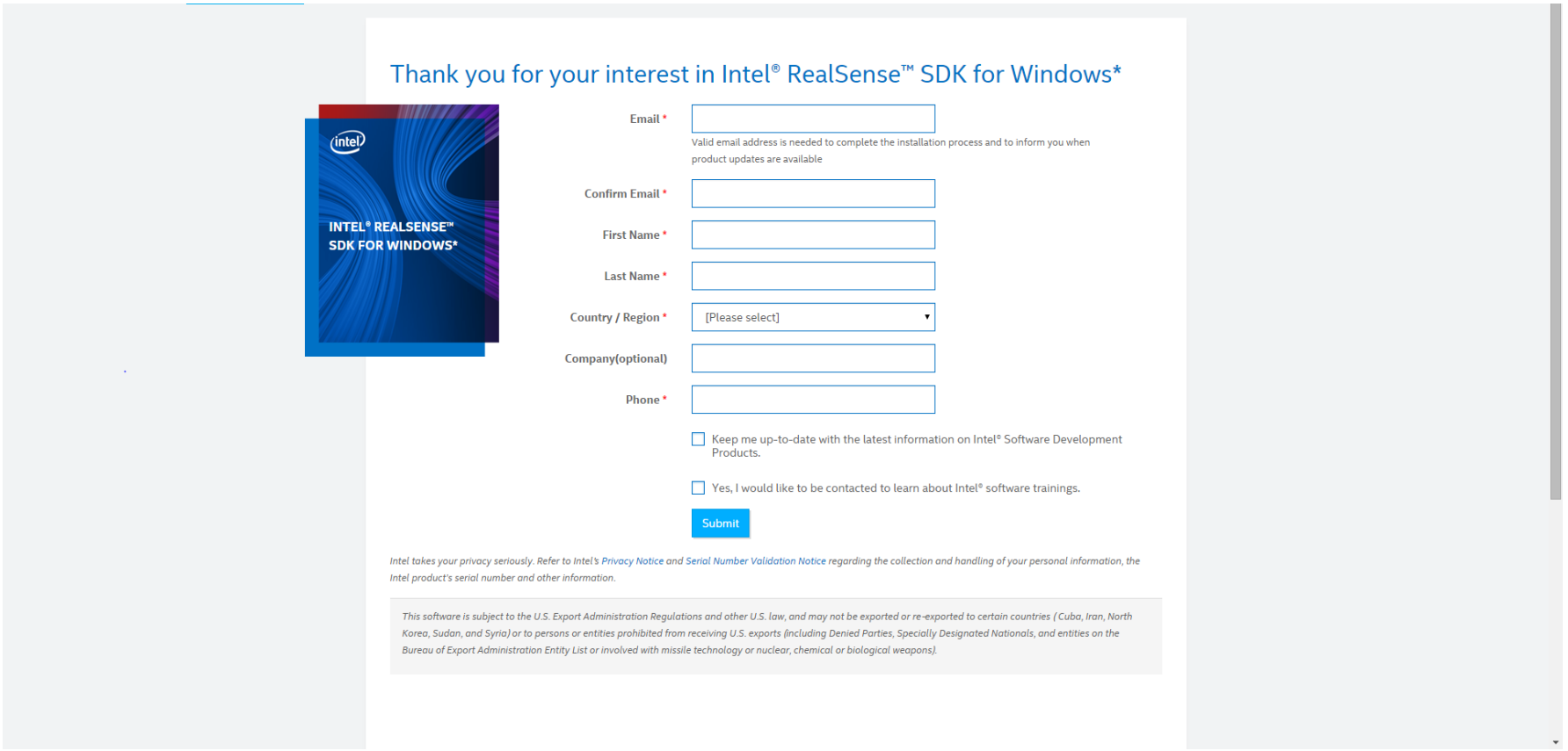
Software Development Kit (SDK) Downloading

- https://software.intel.com/en-us/intel-realsense-sdk/download



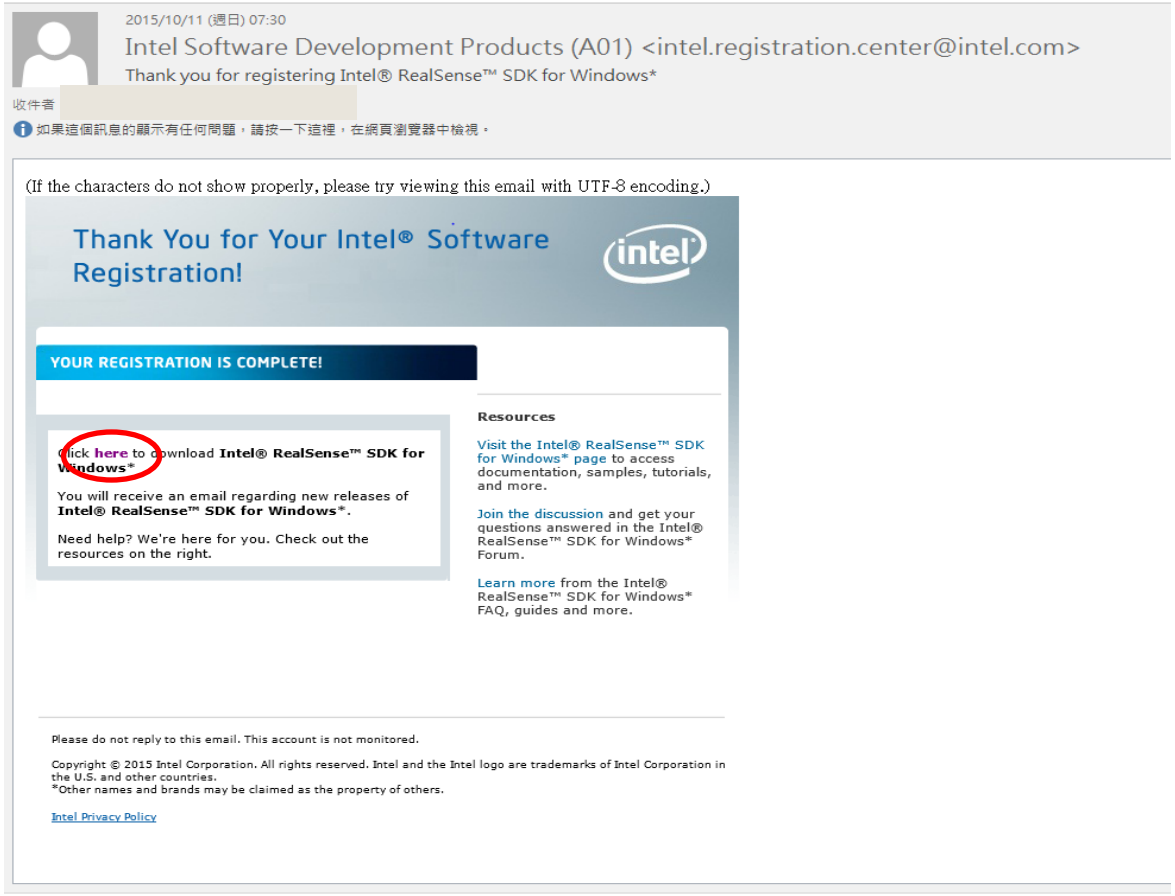
Software Development Kit (SDK) Downloading

- https://software.intel.com/en-us/intel-realsense-sdk/download



Software Development Kit (SDK) Downloading

- https://software.intel.com/en-us/intel-realsense-sdk/download



2015/10/11 (週日) 07:30

Intel Software Development Products (A01) <intel.registration.center@intel.com>

Thank you for registering Intel® RealSense™ SDK for Windows*

收件者

如果這個訊息的顯示有任何問題，請按一下這裡，在網頁瀏覽器中檢視。

(If the characters do not show properly, please try viewing this email with UTF-8 encoding.)

Thank You for Your Intel® Software Registration!

YOUR REGISTRATION IS COMPLETE!

Click here to download **Intel® RealSense™ SDK for Windows***.

You will receive an email regarding new releases of **Intel® RealSense™ SDK for Windows***.

Need help? We're here for you. Check out the resources on the right.

Resources

Visit the Intel® RealSense™ SDK for Windows* page to access documentation, samples, tutorials, and more.

Join the discussion and get your questions answered in the Intel® RealSense™ SDK for Windows* Forum.

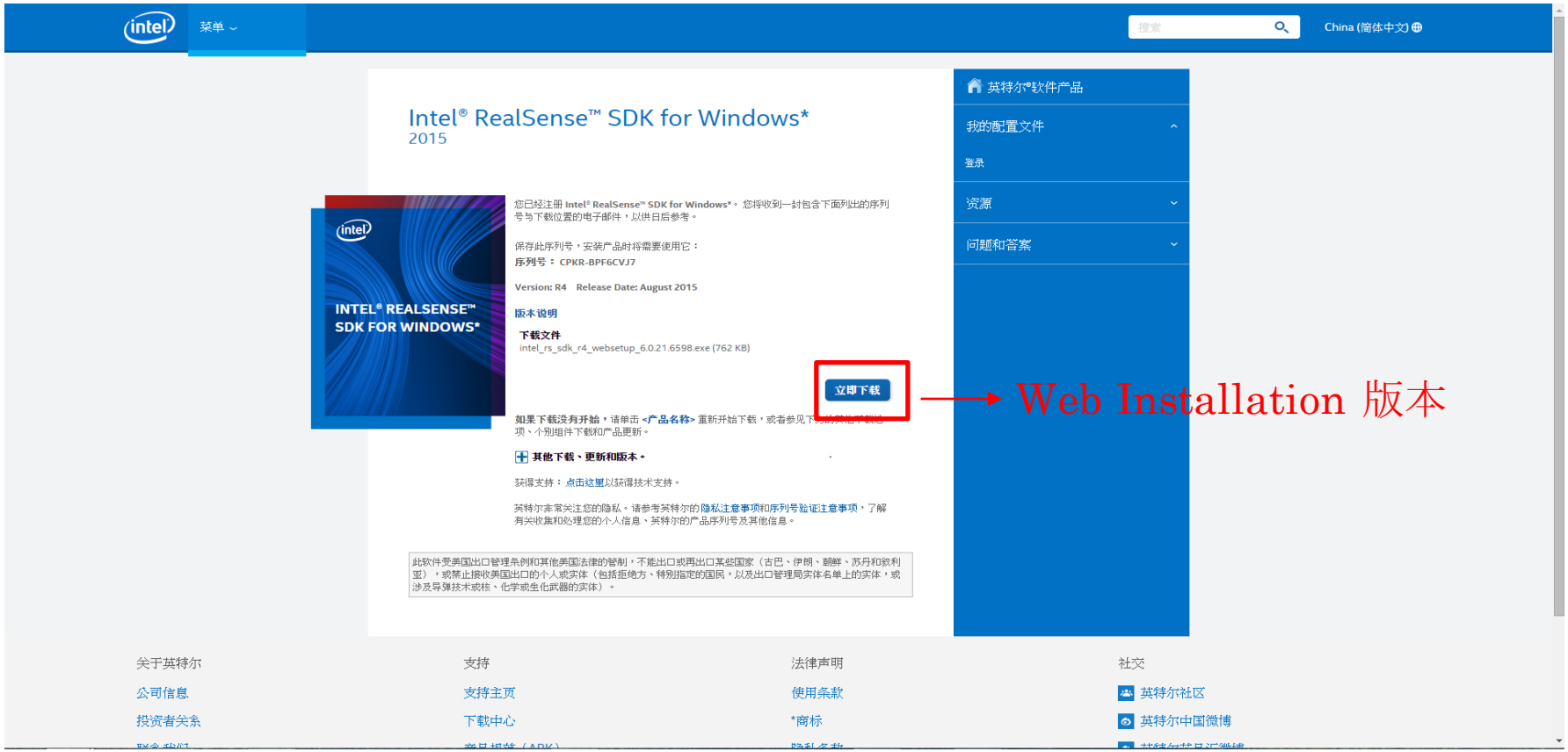
Learn more from the Intel® RealSense™ SDK for Windows* FAQ, guides and more.

Please do not reply to this email. This account is not monitored.

Intel Privacy Policy

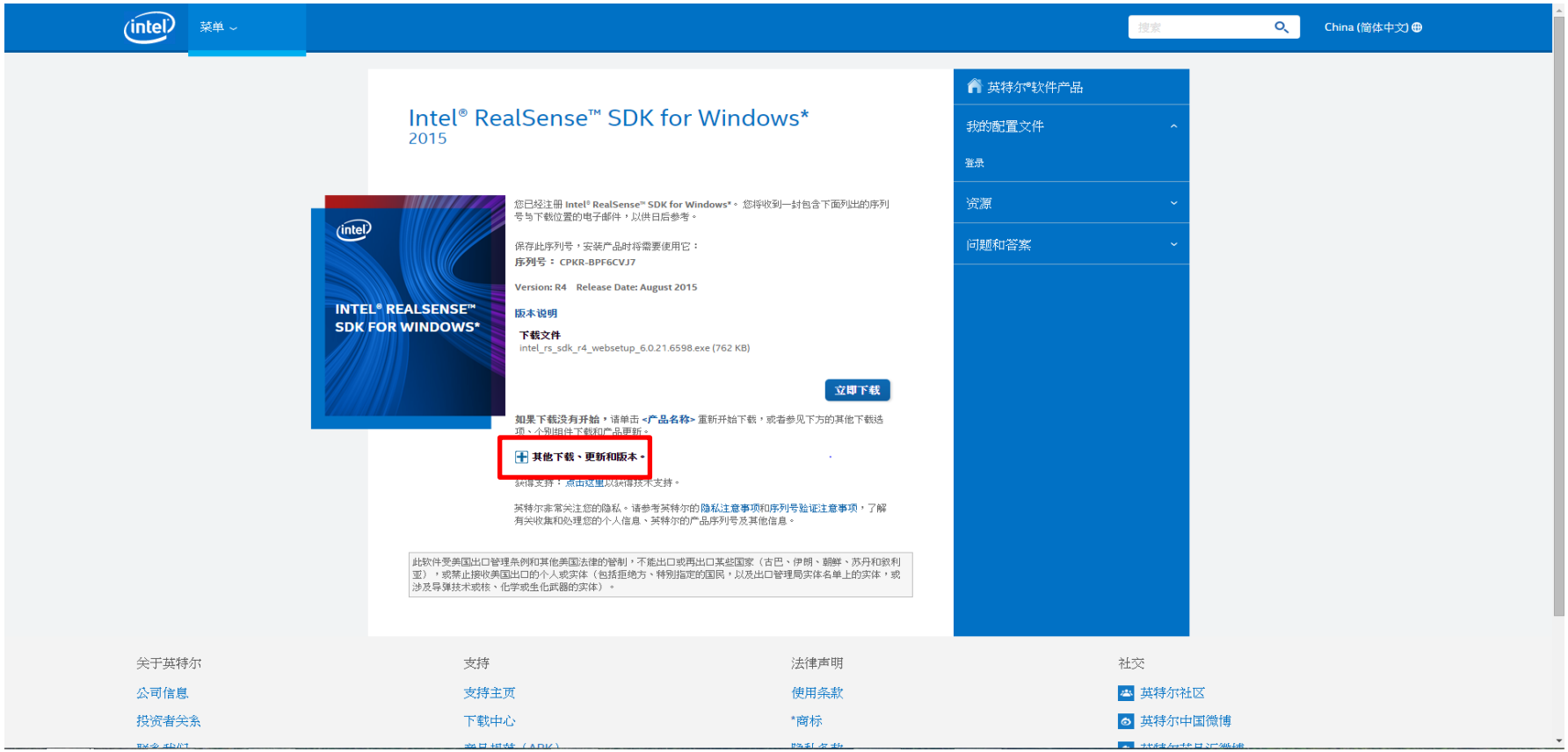
Software Development Kit (SDK) Downloading

- https://software.intel.com/en-us/intel-realsense-sdk/download



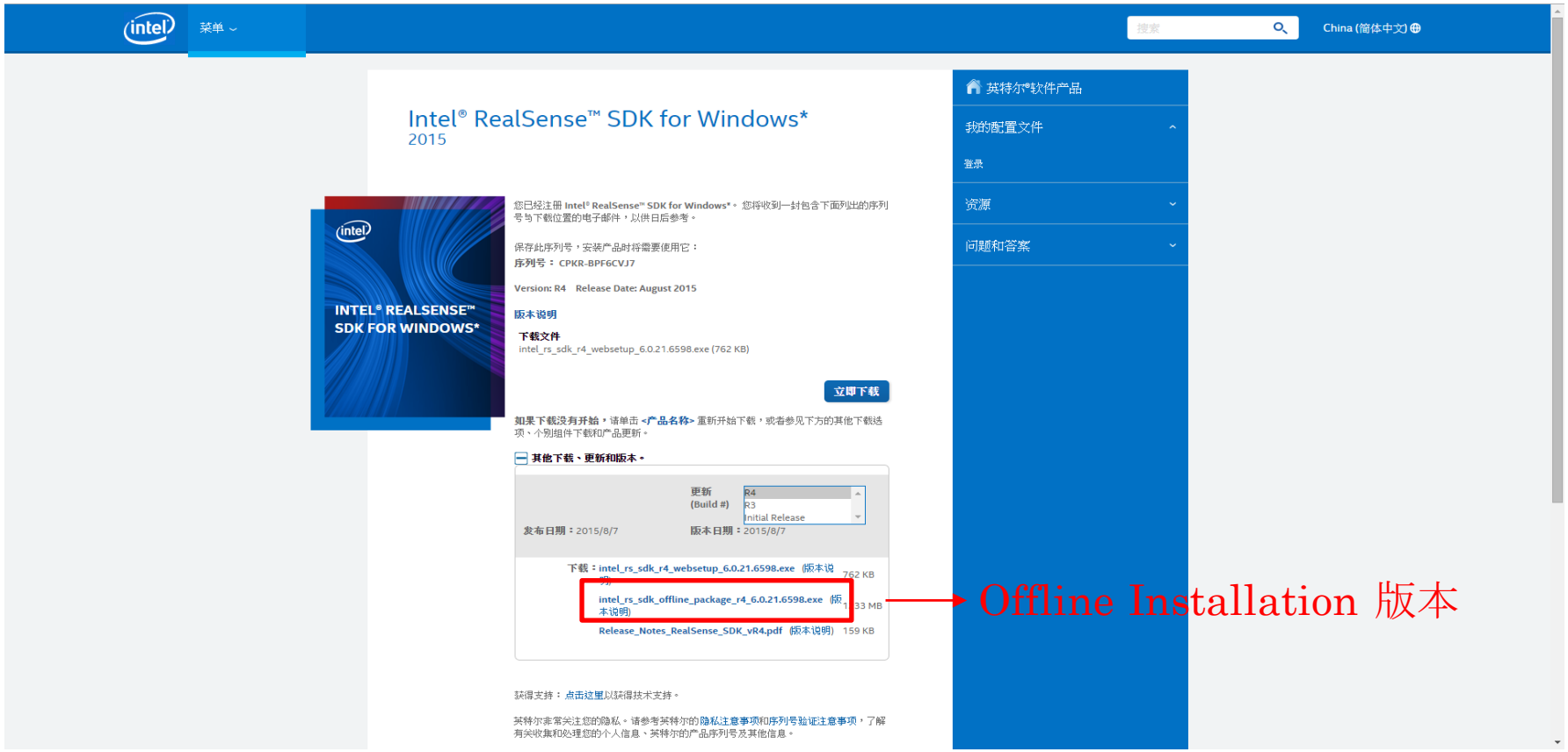
Software Development Kit (SDK) Downloading

- https://software.intel.com/en-us/intel-realsense-sdk/download



Software Development Kit (SDK) Downloading

- https://software.intel.com/en-us/intel-realsense-sdk/download

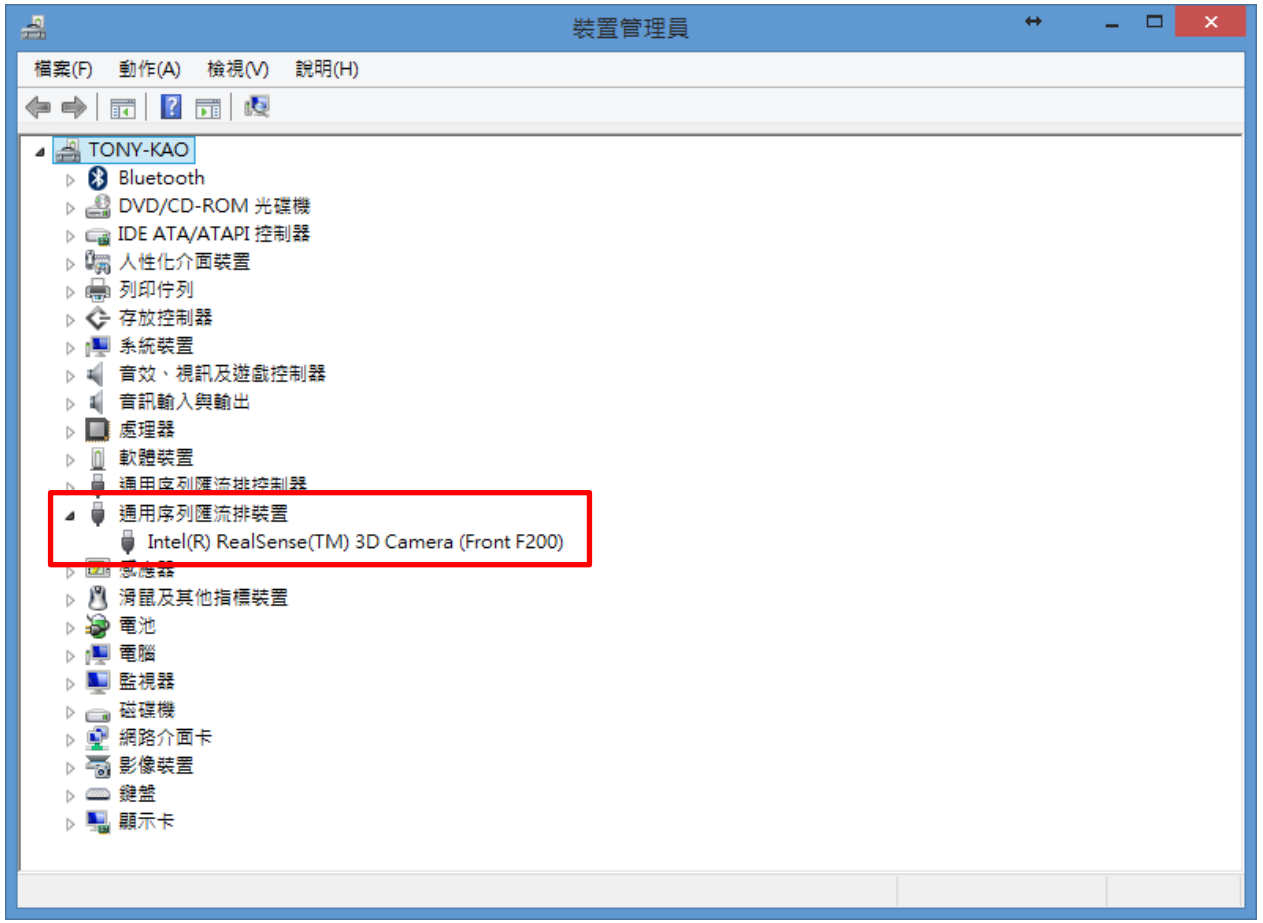


Depth Camera Manager (DCM) Installation

- 安裝前要先接上 **RealSense Camera !!**

Connect RealSense Camera to PC

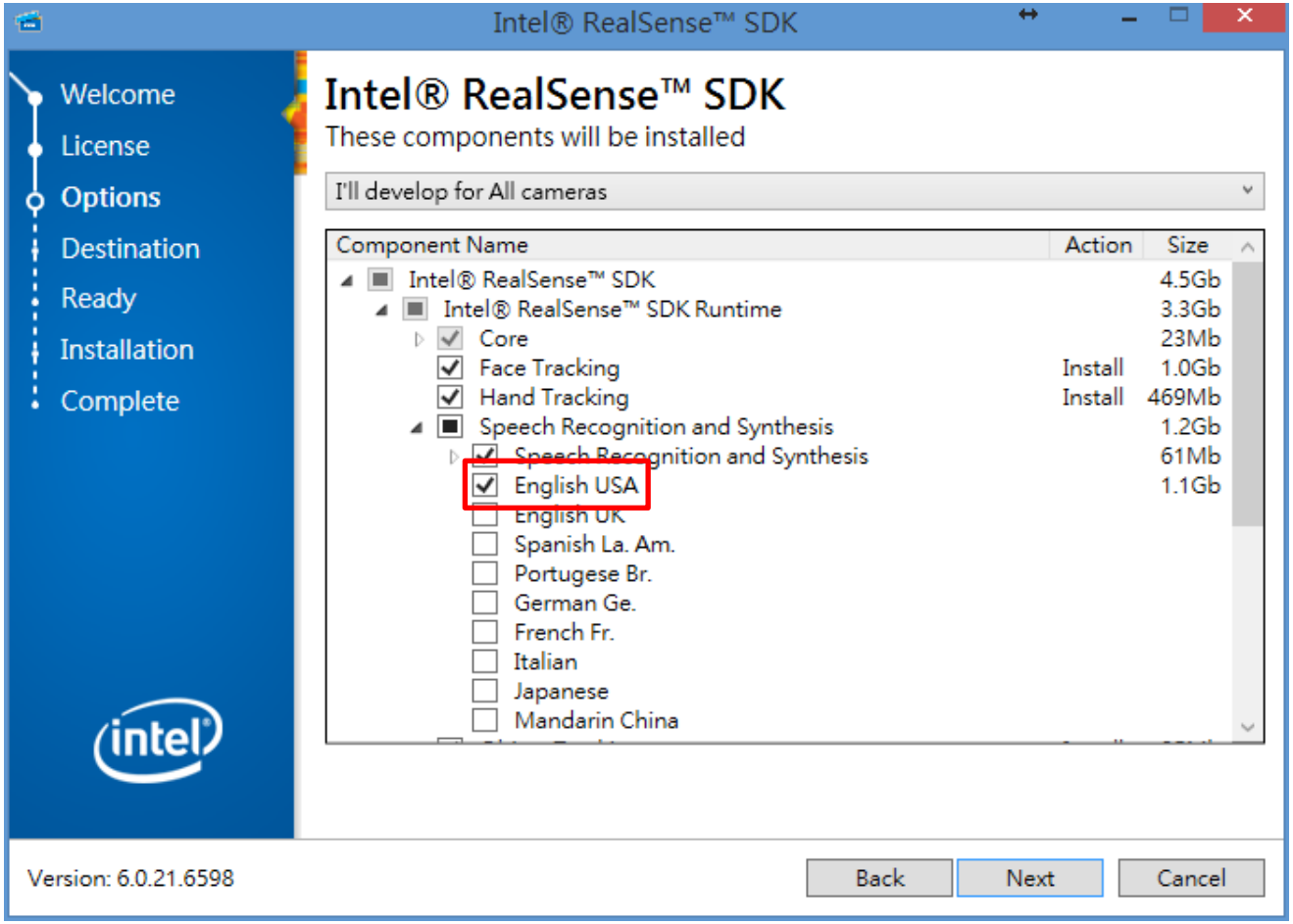
- 接上 RealSense 時, 會有兩次系統音(登登登)



Depth Camera Manager (DCM) Installation

Depth Camera Manager (DCM) Installation

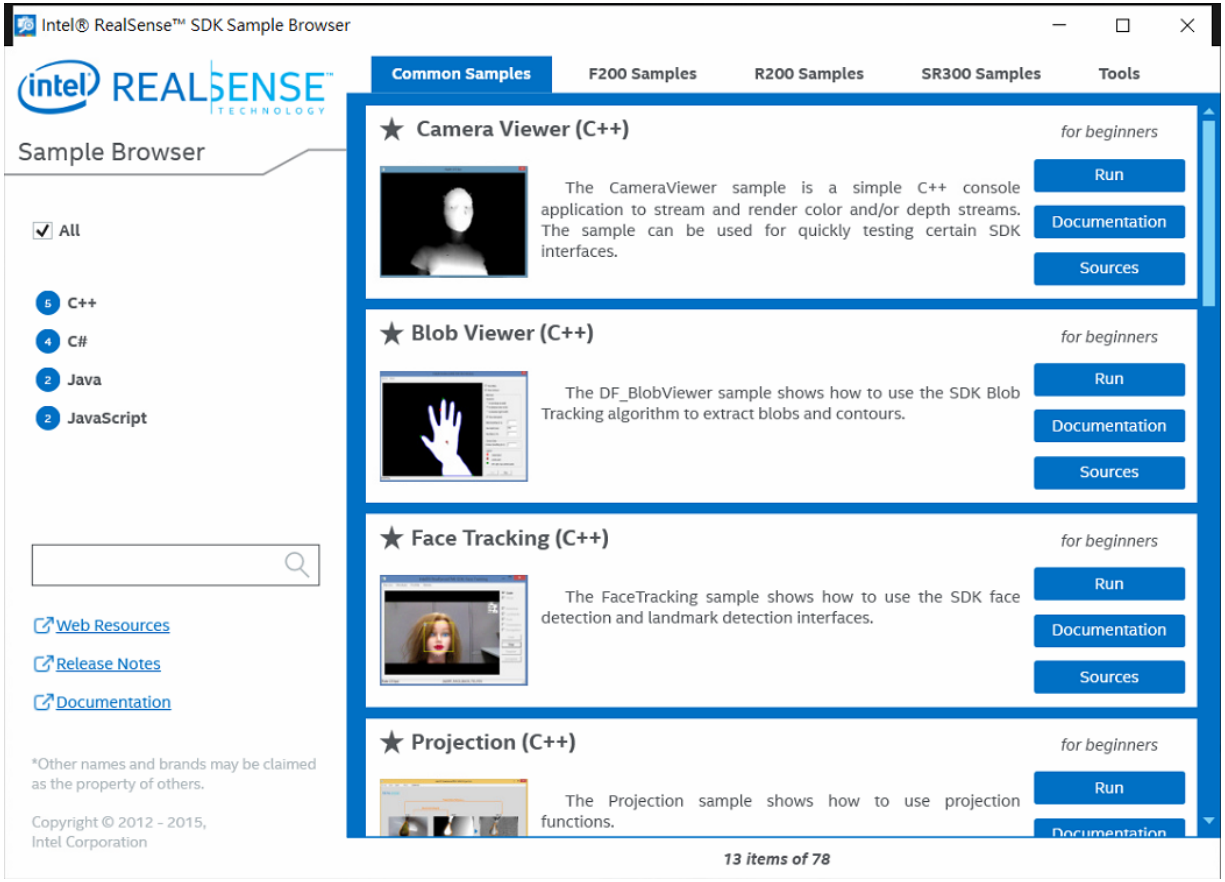
Software Development Kit (SDK) Installation



SDK Sample Browser

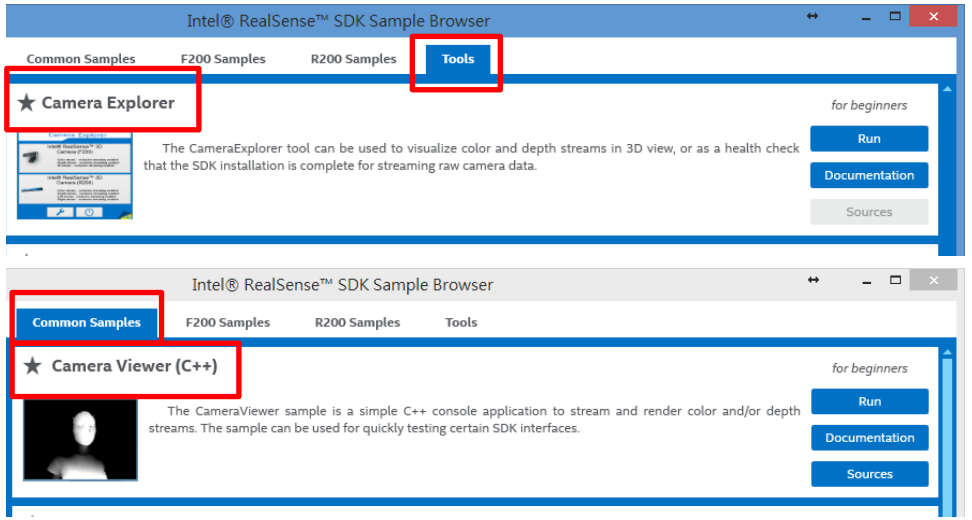
SDK Samples

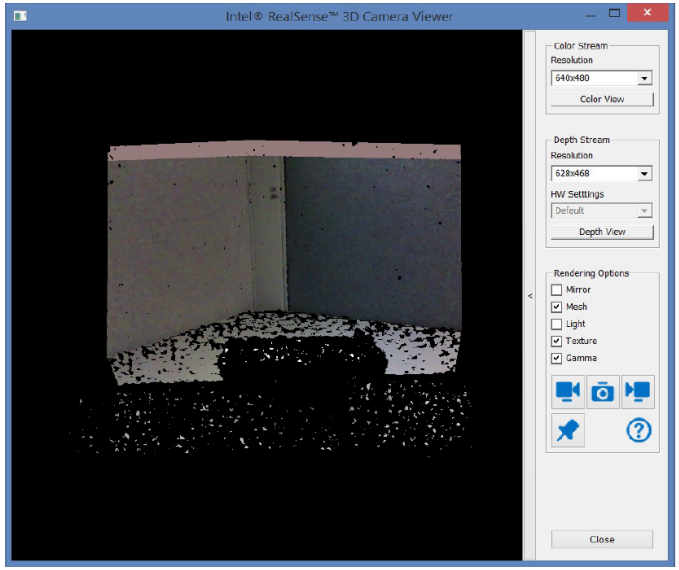
- Programming Language
  - C++
  - C#
  - Java
  - JavaScript
- Platform
  - Windows
  - Unity



SDK Installation Check

- Camera Explorer
- Camera Viewer



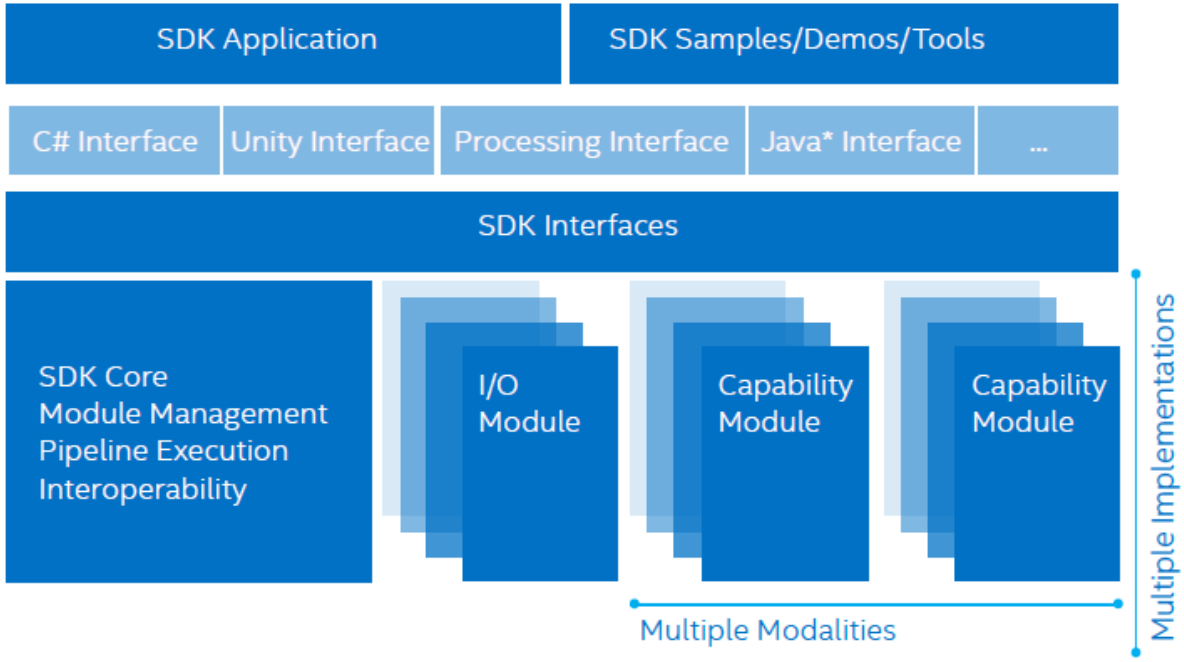


Camera Explorer Camera Selection Window

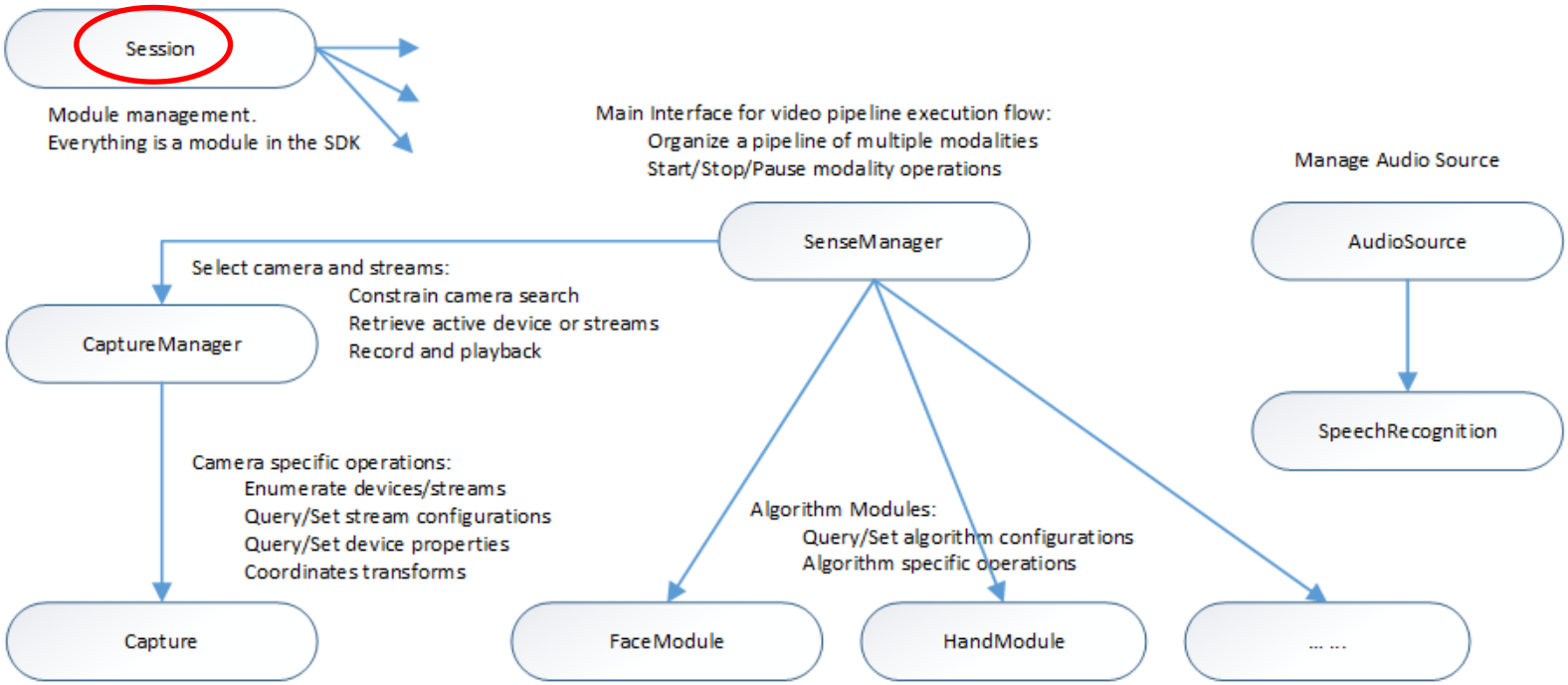
Camera Explorer Streaming Window

Intel® RealSense™ SDK Architecture

- I/O Modules
  - Retrieve input from an input device or send output to an output device.
- Algorithm Modules
  - Include various pattern detection and recognition algorithms that are critical ingredients of innovative human computer experience.
    - Face Recognition
    - Gesture Recognition
    - Speech Recognition
    - Text to Speech
    - …

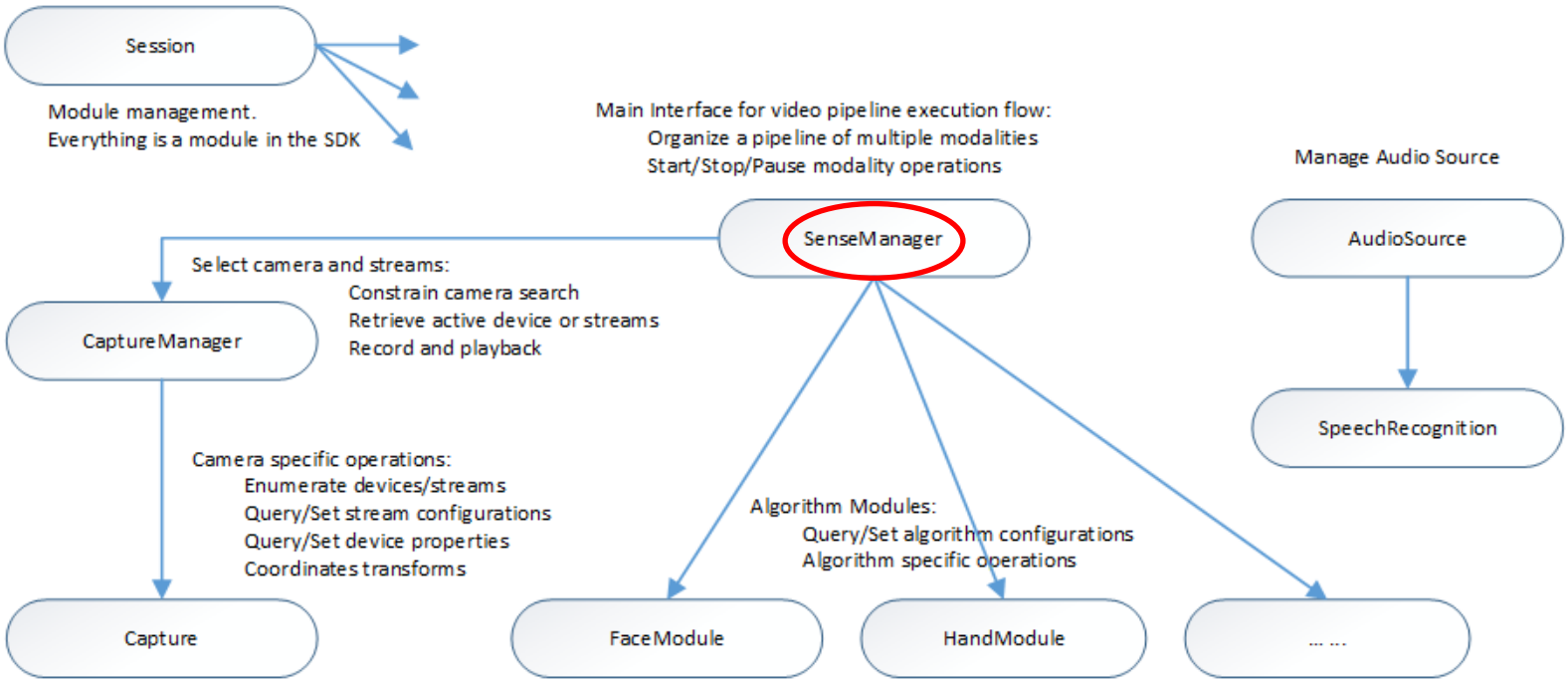


SDK C++/C# Interface Hierarchy



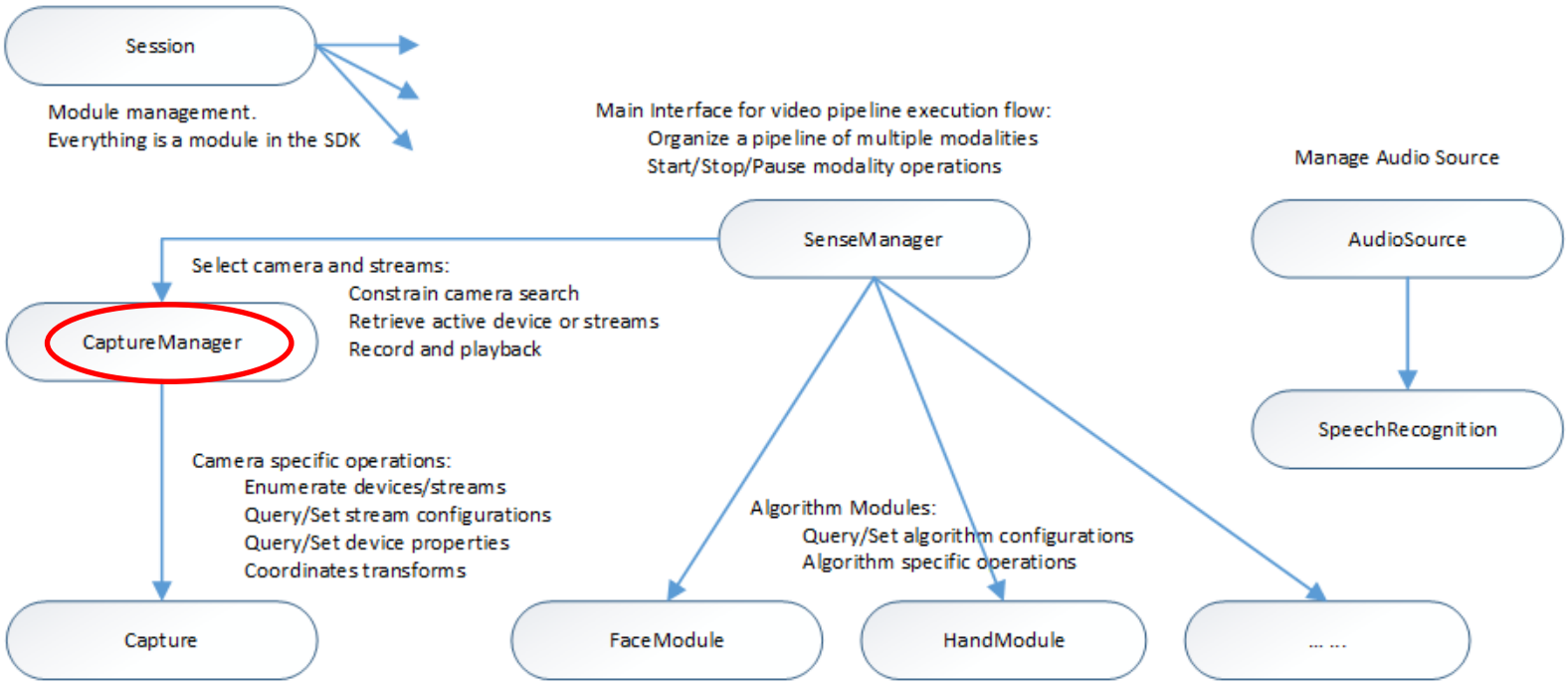
- Session Interface
  - Manages the following modules: I/O modules, algorithm modules, and any other SDK interface implementations.

SDK C++/C# Interface Hierarchy



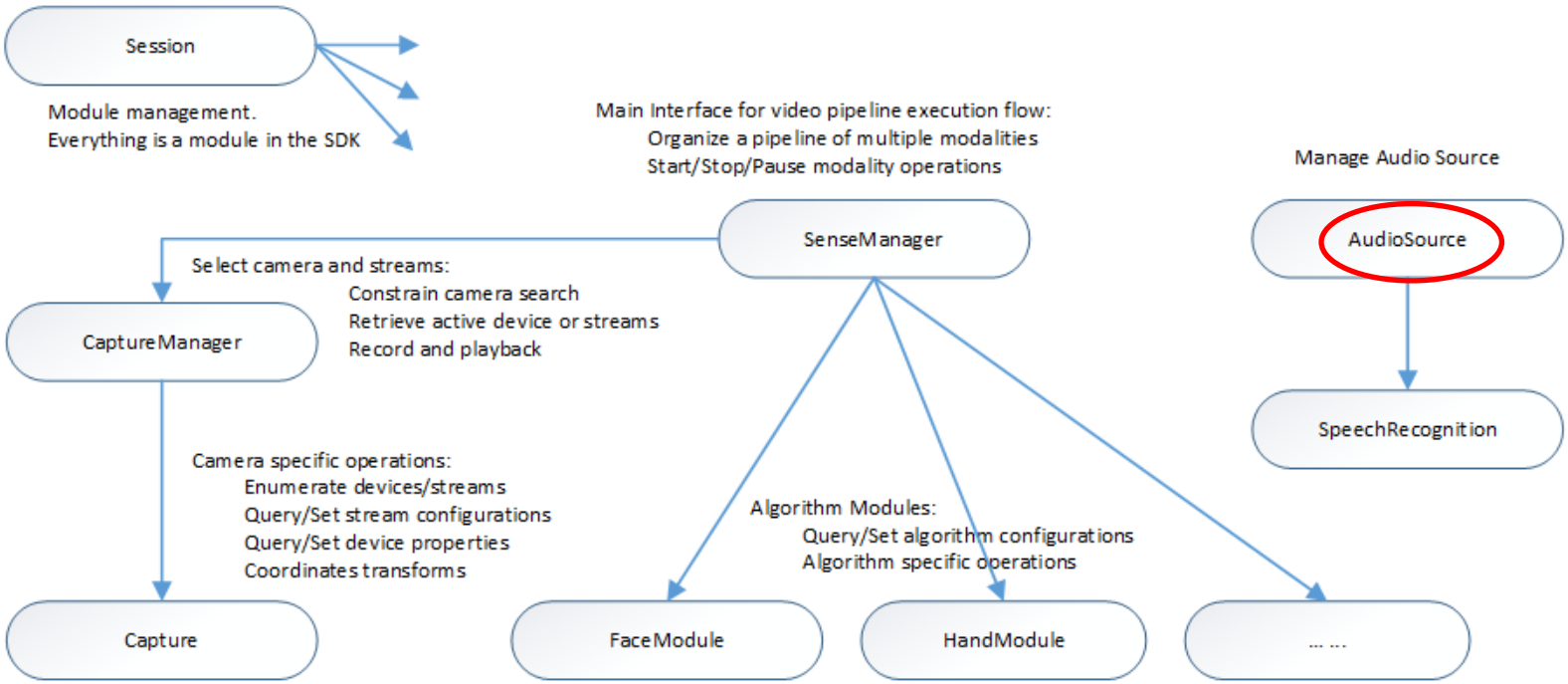
- SenseManager Interface
  - This interface organizes a multi-modal pipeline (that contains an I/O device and multiple algorithm modules) and controls the execution of the pipeline such as starting, stopping, pausing and resuming the pipeline.

SDK C++/C# Interface Hierarchy



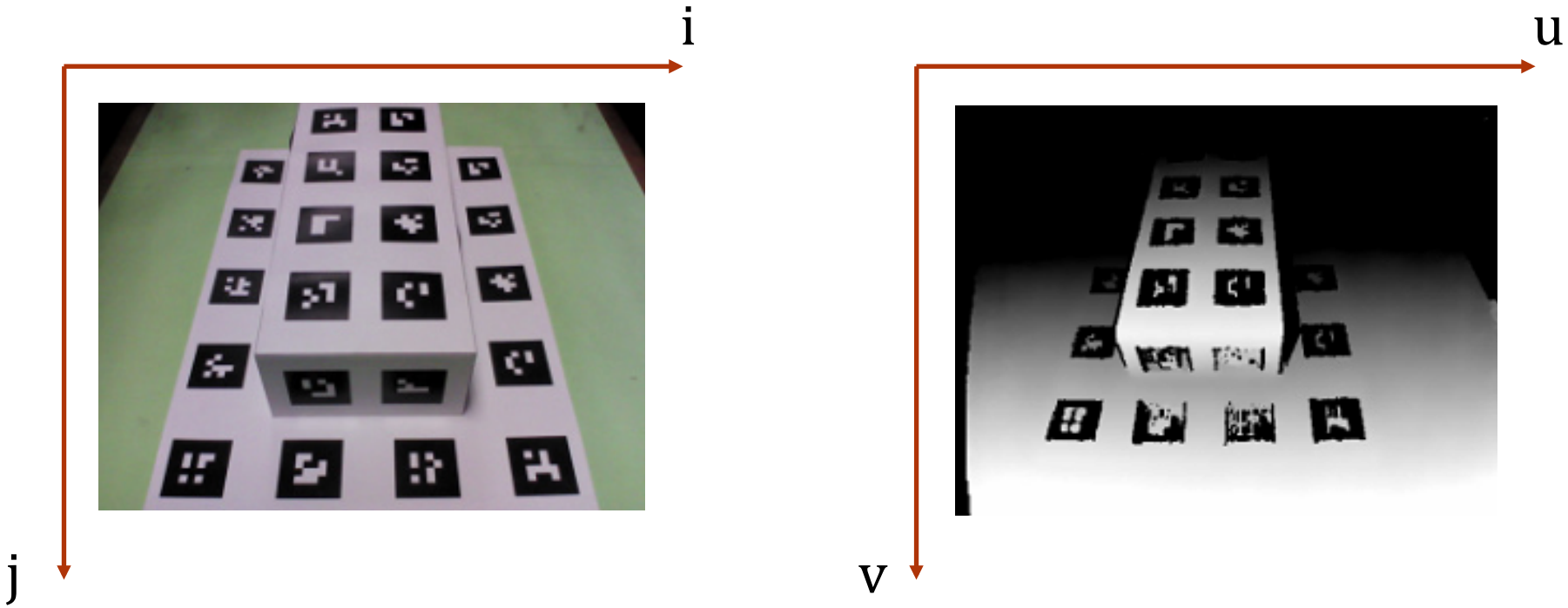
- The SenseManager interface uses the CaptureManager interface to select the I/O device and color/depth/audio streams.
- Retrieve the CaptureManager instance (from the SenseManager interface) to constrain the device search and/or to set file recording and playback during the pipeline initialization.

SDK C++/C# Interface Hierarchy



- AudioSource Interface
  - Manages the audio source and specific voice features directly in the module interface.

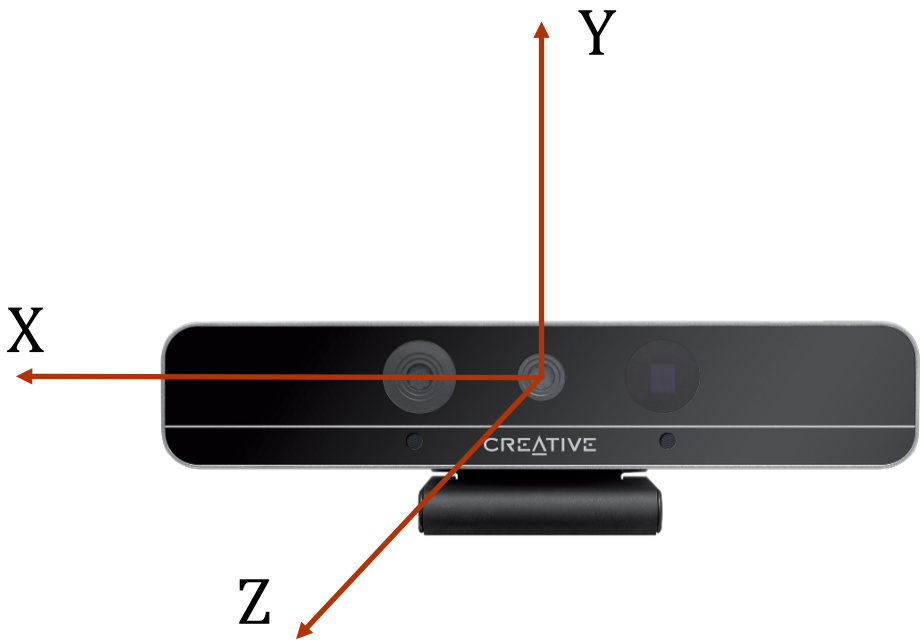
Image Coordinate System



Color Image

Depth Image

World Coordinate System



Hardware & Software Requirements and Supported Tools

Required Hardware	A system with a minimum of a **4th generation Intel® Core™ processor**, either IA-32 or Intel® 64, with integrated or peripheral depth camera
Required OS	**Microsoft Windows 8.1 OS (64-bit) Desktop Mode** Microsoft Windows 8.1 New UI (Metro) Mode (coming soon)**
Supported Programming Languages	C++, C#, Java*, JavaScript
Supported IDE	**Microsoft Visual Studio* C++ 2010-2013 with service pack 1** **or newer**
Supported Development Optional Tools	Microsoft .NET* 4.0 Framework for C# development Unity* PRO 4.1.0 or later for Unity game development Processing* 2.1.2 or later for Processing framework development Java JDK 1.7.0_11 or higher for Java development

Effective Depth Range of the SDK Algorithms

Effective Depth Range of the SDK Algorithms

* Require at least 5000 pixels at 640x480 for proper detection, which translates to

Distance	Minimum Rectangular Object Size
30 cm	4.5 cm x 3.5 cm
60 cm	9 cm x 7 cm
100 cm	16 cm x 11 cm
150 cm	23 cm x 18 cm
180 cm	28 cm x 21 cm

Algorithm		Operating Range	
		Detection (cm)	Tracking (cm)
Face	Detection	25~75	30~100
	Landmark	30~100	30~100
	Recognition	30~80	30~80
	Expression	30~100	30~100
	Pulse	30~60	30~60
	Pose	30~100	30~100
Hand	Segmentation**	20~80	20~80
	Tracking**	20~60	20~60
	Gesture**	20~60	20~60
Others	Blob Segmentation	20~85	20~85
	Object Tracking	30~180*	30~180
	Emotion	25~100	25~100

Effective Depth Range of the SDK Algorithms

** Require a minimum palm size of 5.5 cm (about a fix-year old kid palm size) and hand interaction speed within the following range:

Resolution	Speed Limitation
HVGA	2 m/s
VGA	0.75 m/s

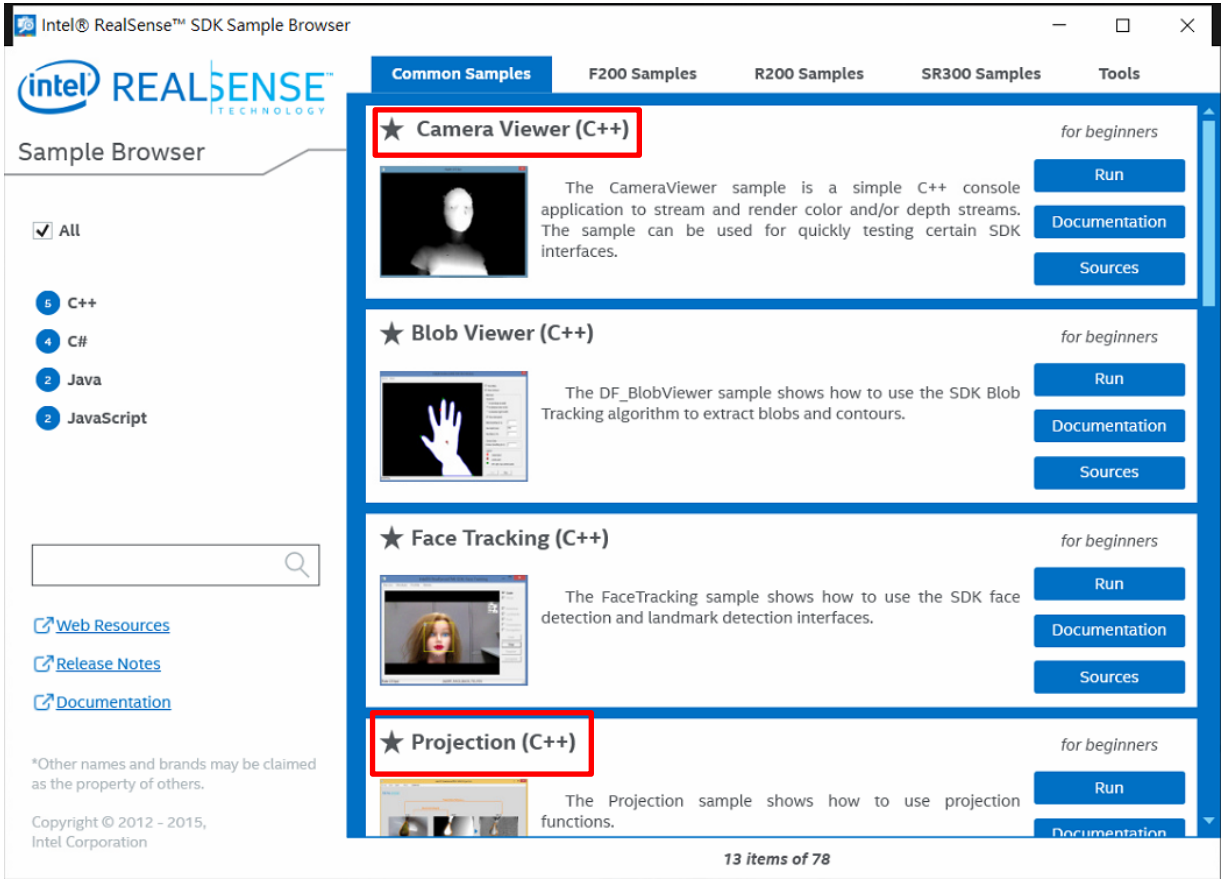
Algorithm		Operating Range	
		Detection (cm)	Tracking (cm)
Face	Detection	25~75	30~100
	Landmark	30~100	30~100
	Recognition	30~80	30~80
	Expression	30~100	30~100
	Pulse	30~60	30~60
	Pose	30~100	30~100
Hand	Segmentation**	20~80	20~80
	Tracking**	20~60	20~60
	Gesture**	20~60	20~60
Others	Blob Segmentation	20~85	20~85
	Object Tracking	30~180*	30~180
	Emotion	25~100	25~100

Two Examples

- Configuring Project Setting
- Camera Viewer
- 3D Data Acquisition (an Example)

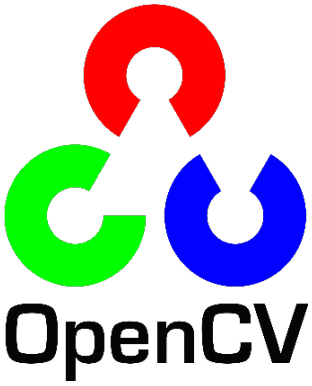
SDK Samples

- Programming Language
  - C++
  - C#
  - Java
  - JavaScript
- Platform
  - Windows
  - Unity



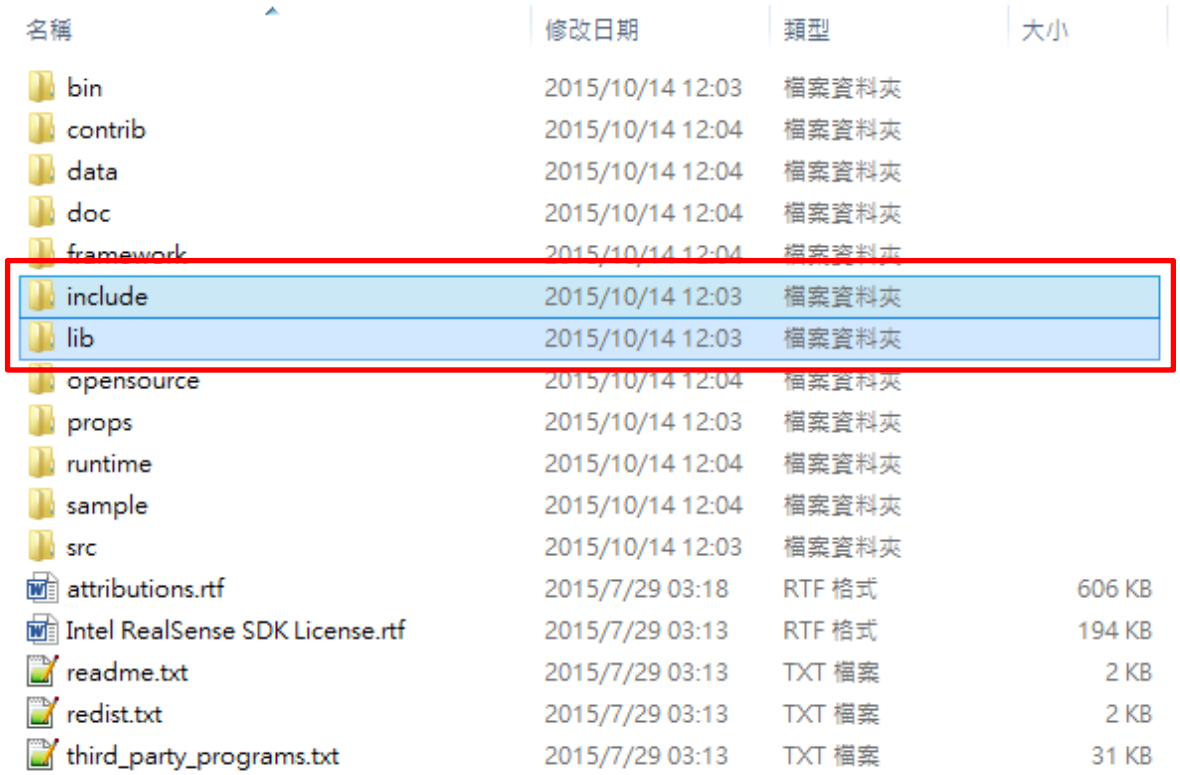
Library & Tool

- Microsoft Visual Studio Professional 2013
- OpenCV 2.4.12
  - core
  - highgui
- RealSense SDK R4



Configuring Project Setting

- …/Intel/RSSDK/



名稱	修改日期	類型	大小
bin	2015/10/14 12:03	檔案資料夾	
contrib	2015/10/14 12:04	檔案資料夾	
data	2015/10/14 12:04	檔案資料夾	
doc	2015/10/14 12:04	檔案資料夾	
framework	2015/10/14 12:04	檔案資料夾	
include	2015/10/14 12:03	檔案資料夾	
lib	2015/10/14 12:03	檔案資料夾	
opensource	2015/10/14 12:04	檔案資料夾	
props	2015/10/14 12:03	檔案資料夾	
runtime	2015/10/14 12:04	檔案資料夾	
sample	2015/10/14 12:04	檔案資料夾	
src	2015/10/14 12:03	檔案資料夾	
attributions.rtf	2015/7/29 03:18	RTF 格式	606 KB
Intel RealSense SDK License.rtf	2015/7/29 03:13	RTF 格式	194 KB
readme.txt	2015/7/29 03:13	TXT 檔案	2 KB
redist.txt	2015/7/29 03:13	TXT 檔案	2 KB
third_party_programs.txt	2015/7/29 03:13	TXT 檔案	31 KB

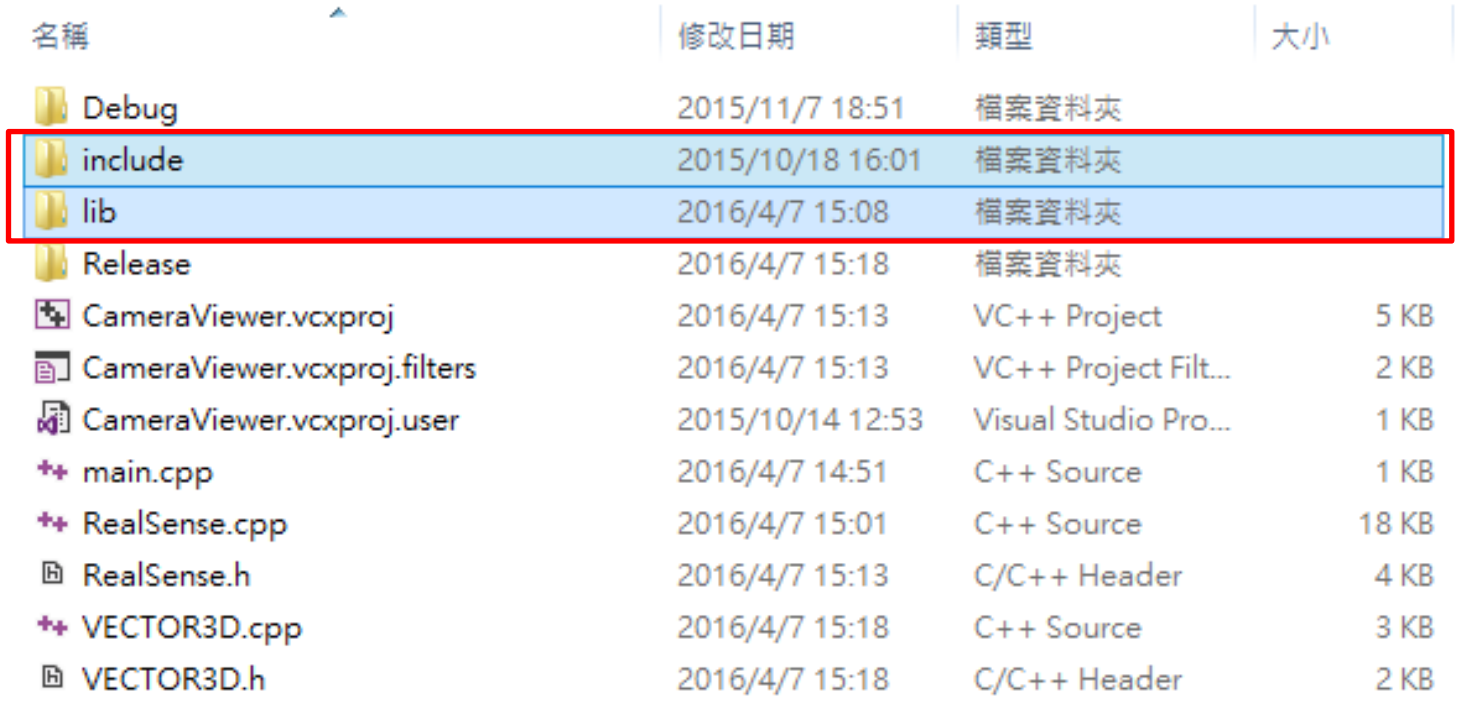
Configuring Project Setting

- …/Intel/RSSDK/lib/Win32/

名稱	修改日期	類型	大小
libpxc.lib	2015/7/29 04:11	Object File Library	96 KB
libpxc_d.lib	2015/7/29 04:11	Object File Library	92 KB
libpxcmd.lib	2015/7/29 04:12	Object File Library	96 KB
libpxcmd_d.lib	2015/7/29 04:12	Object File Library	92 KB

Configuring Project Setting

- .../(SolutionName)/(ProjectName)/



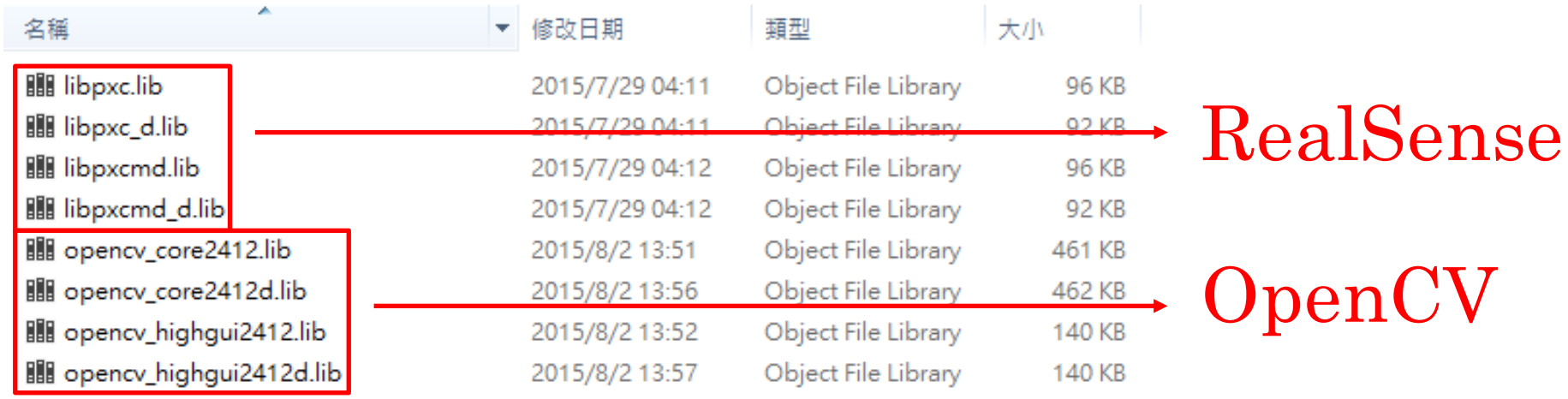
名稱	修改日期	類型	大小
Debug	2015/11/7 18:51	檔案資料夾	
include	2015/10/18 16:01	檔案資料夾	
lib	2016/4/7 15:08	檔案資料夾	
Release	2016/4/7 15:18	檔案資料夾	
CameraViewer.vcxproj	2016/4/7 15:13	VC++ Project	5 KB
CameraViewer.vcxproj.filters	2016/4/7 15:13	VC++ Project Filt...	2 KB
CameraViewer.vcxproj.user	2015/10/14 12:53	Visual Studio Pro...	1 KB
main.cpp	2016/4/7 14:51	C++ Source	1 KB
RealSense.cpp	2016/4/7 15:01	C++ Source	18 KB
RealSense.h	2016/4/7 15:13	C/C++ Header	4 KB
VECTOR3D.cpp	2016/4/7 15:18	C++ Source	3 KB
VECTOR3D.h	2016/4/7 15:18	C/C++ Header	2 KB

Configuring Project Setting

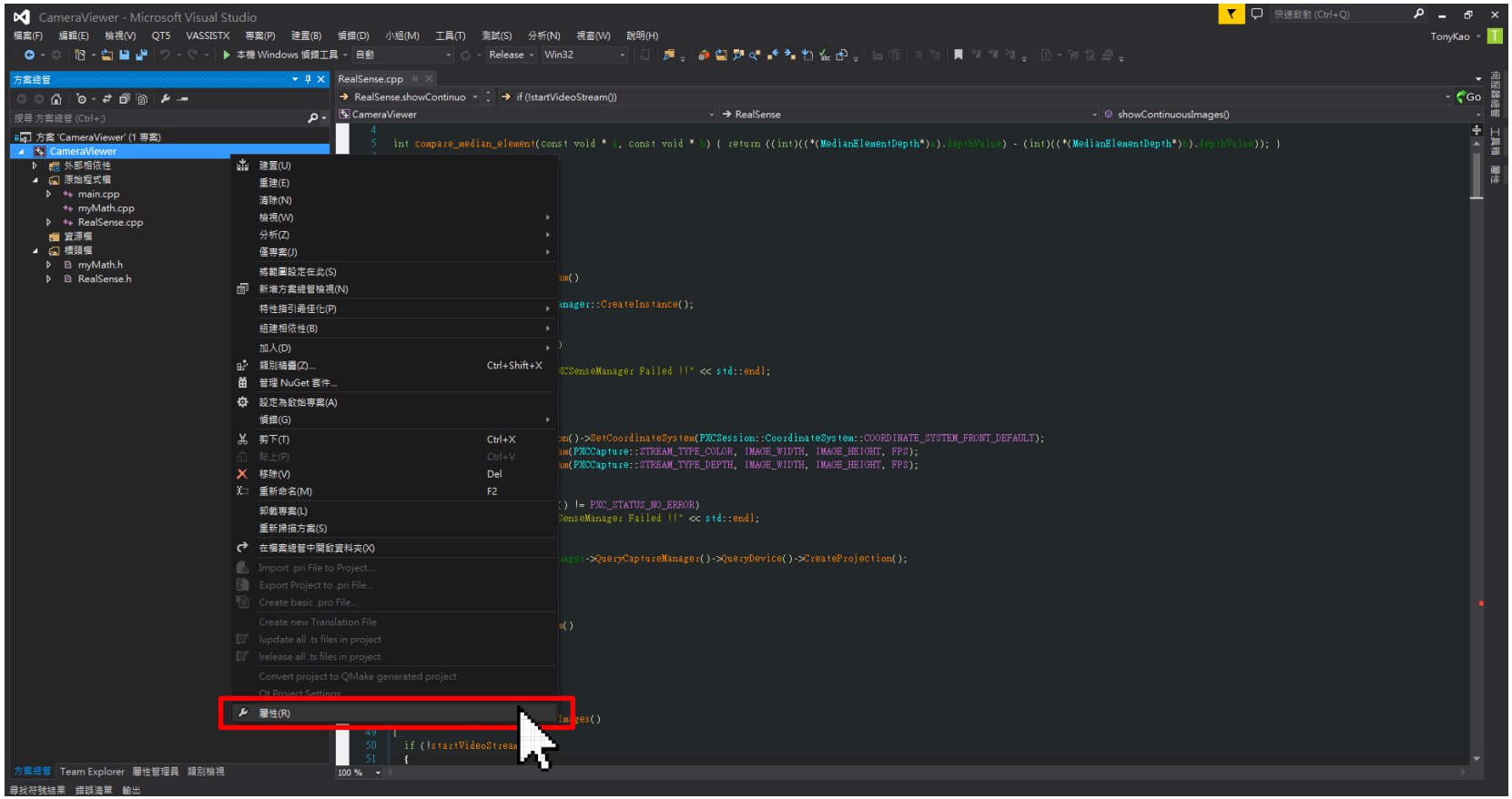
- .../(SolutionName)/(ProjectName)/include/

Configuring Project Setting

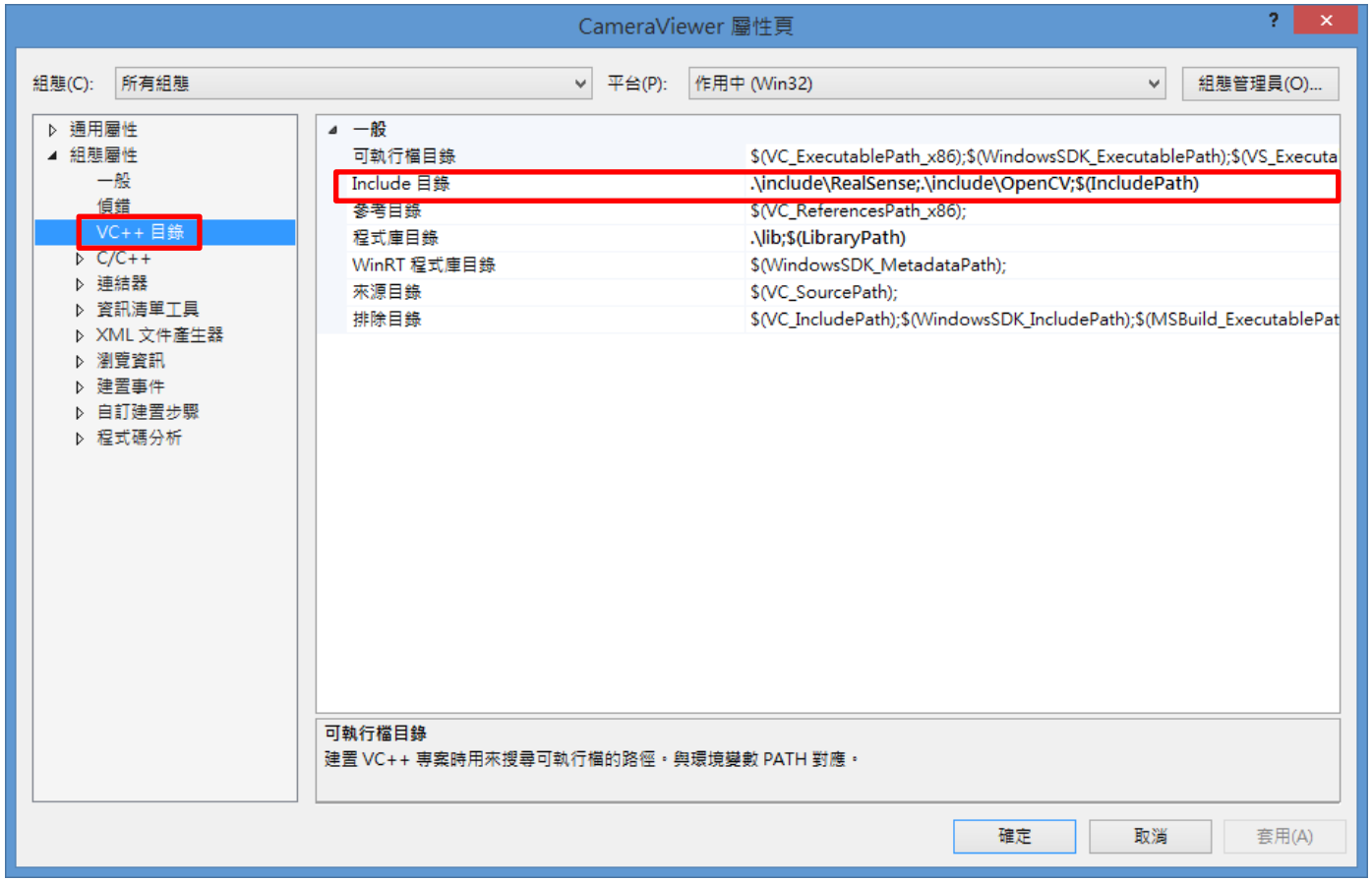
- .../(SolutionName)/(ProjectName)/lib



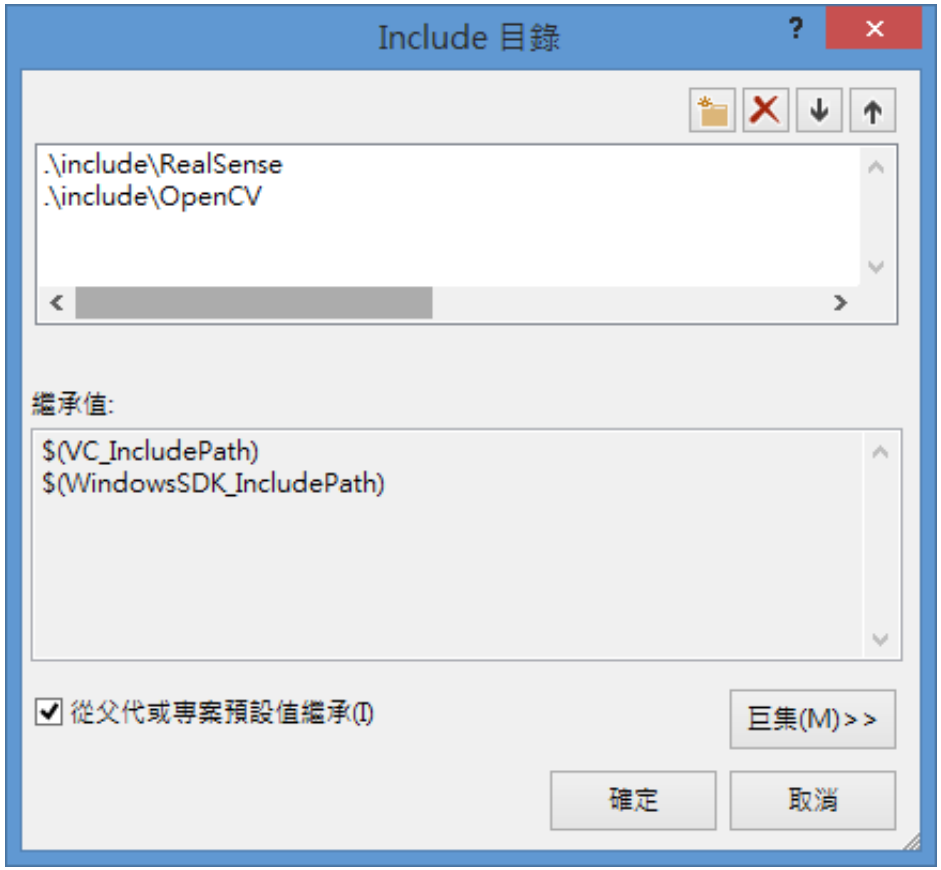
Configuring Project Setting



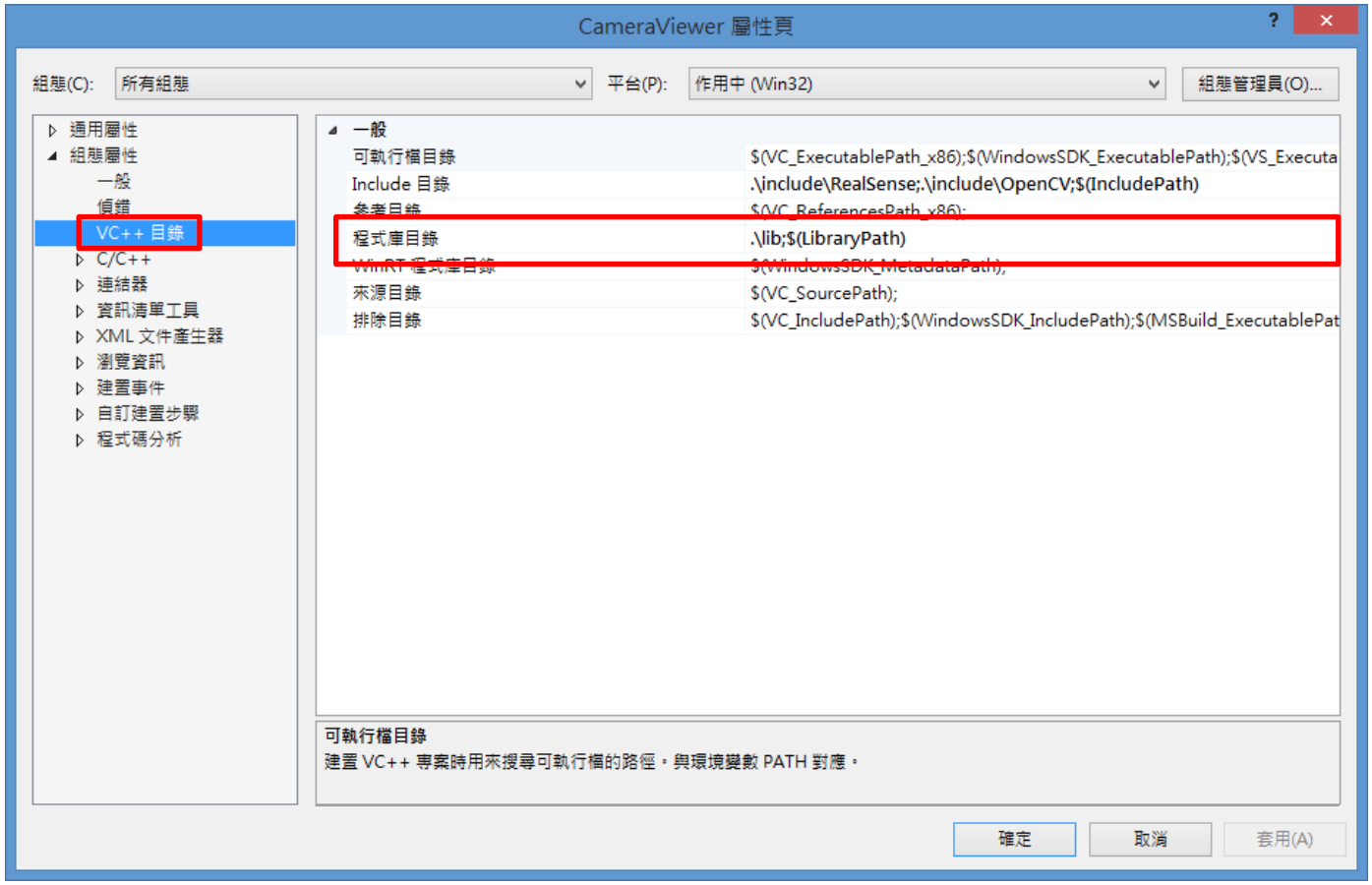
Configuring Project Setting - include



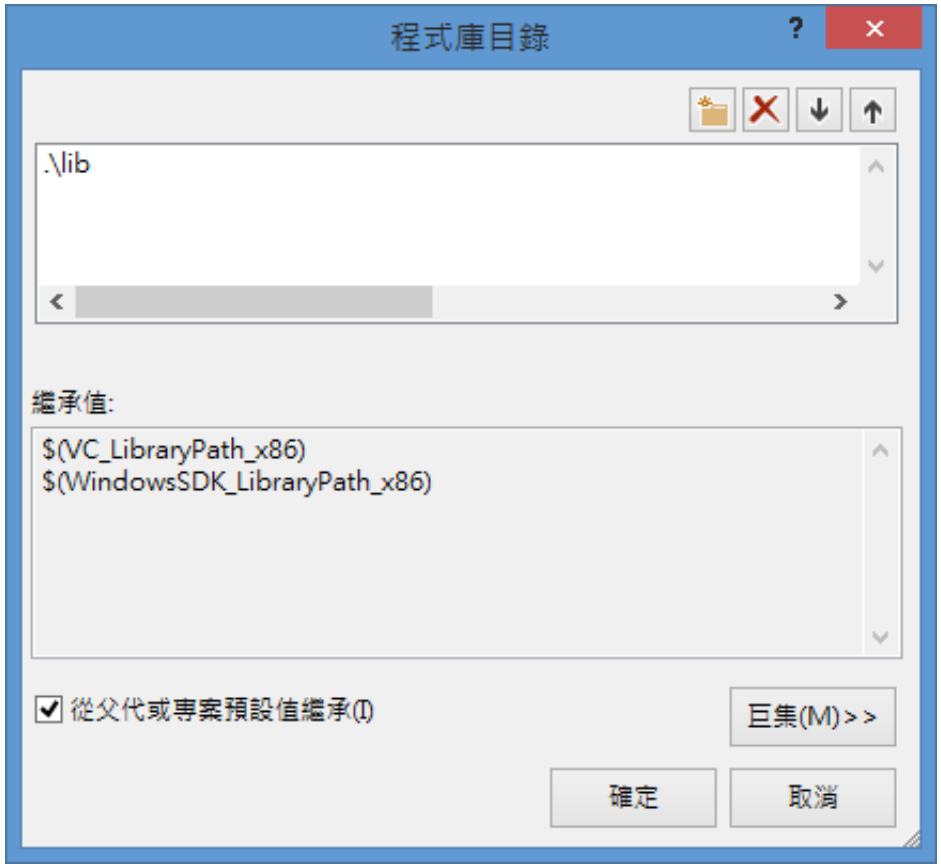
Configuring Project Setting - include



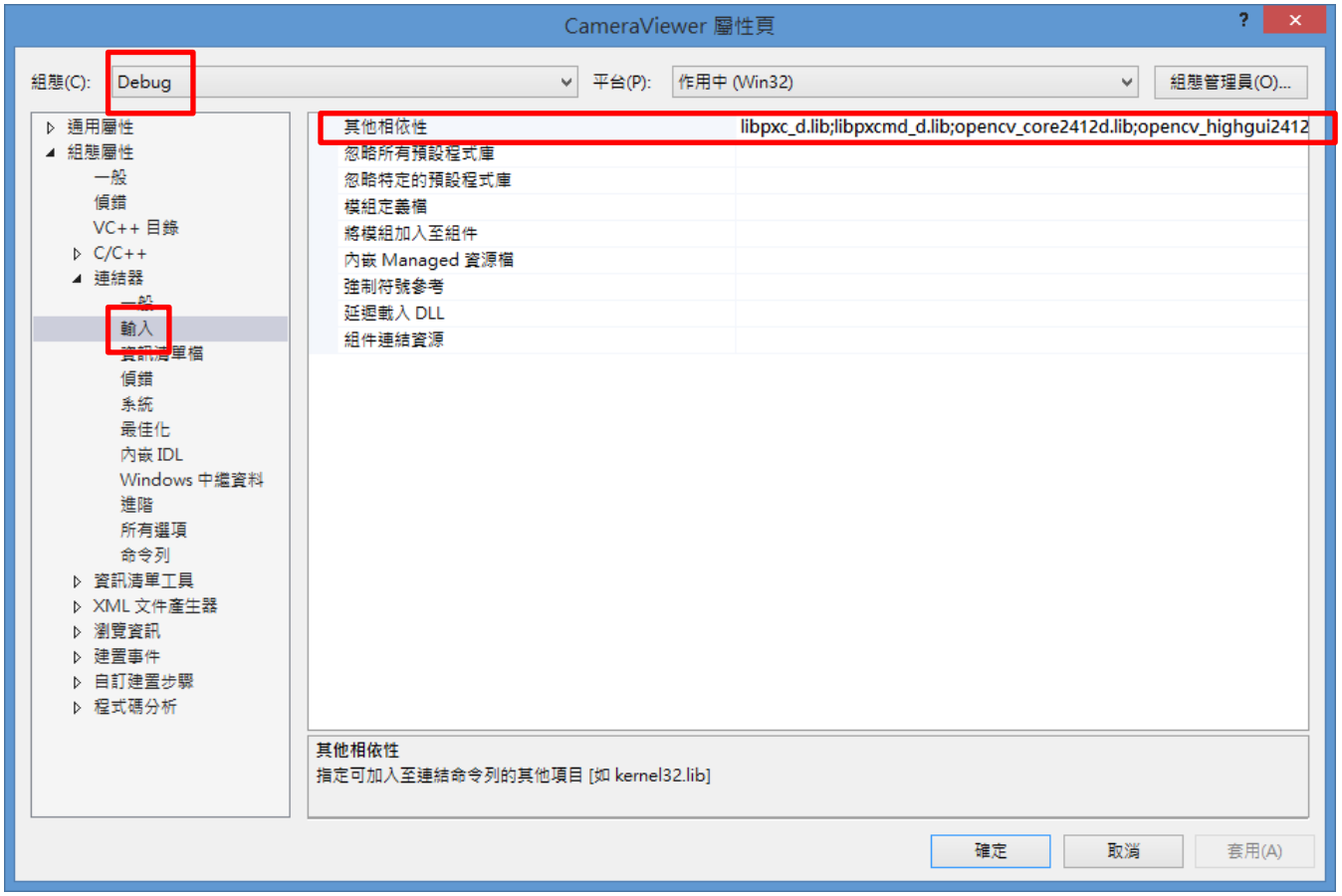
Configuring Project Setting - lib



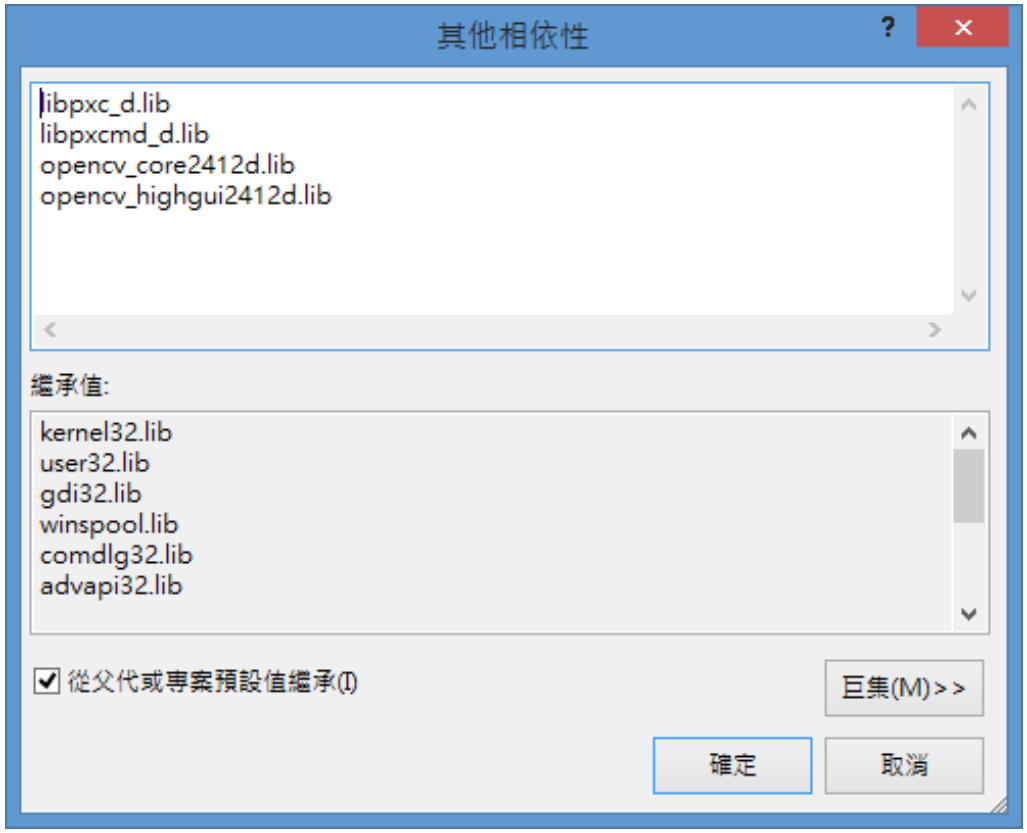
Configuring Project Setting - lib



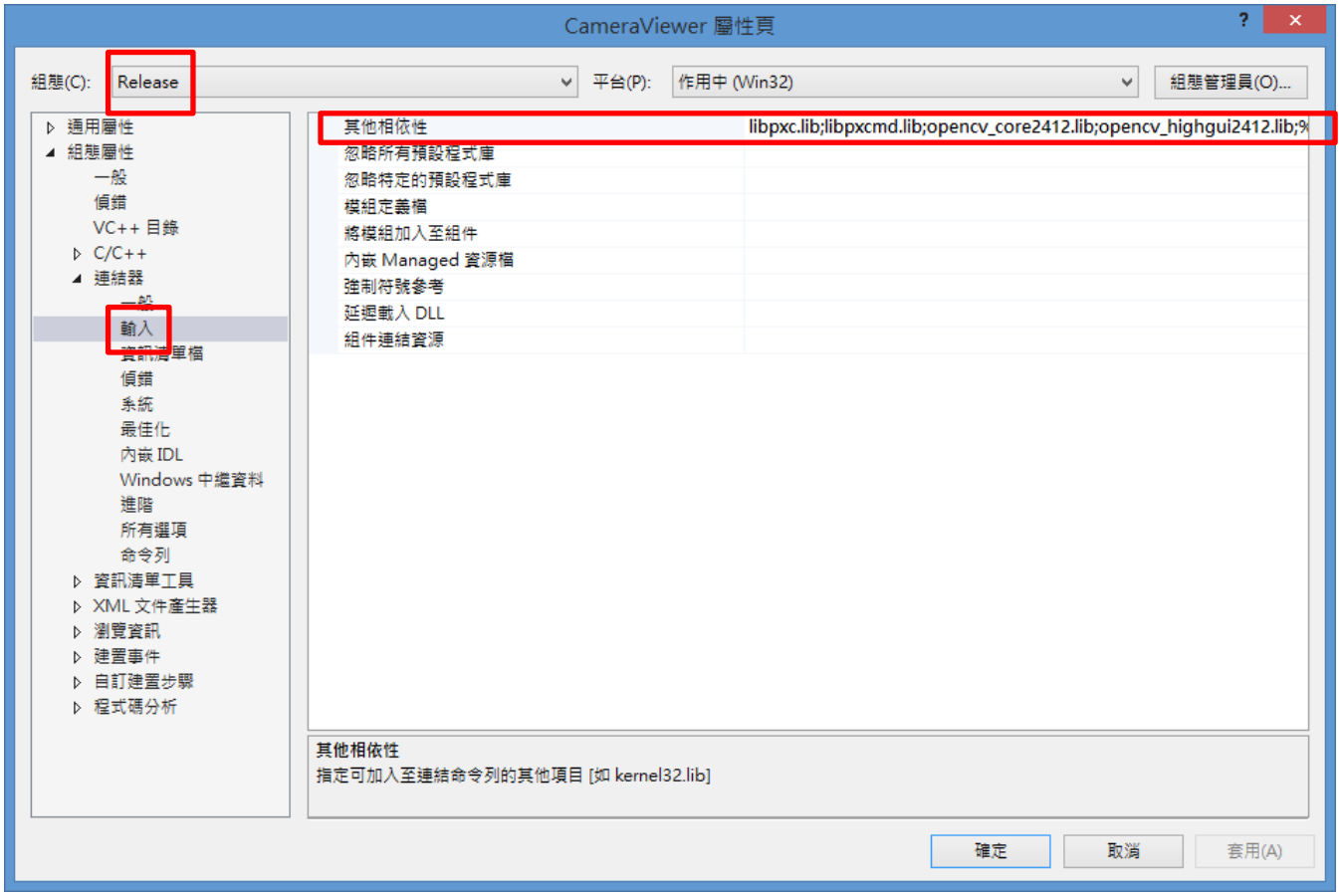
Configuring Project Setting - lib



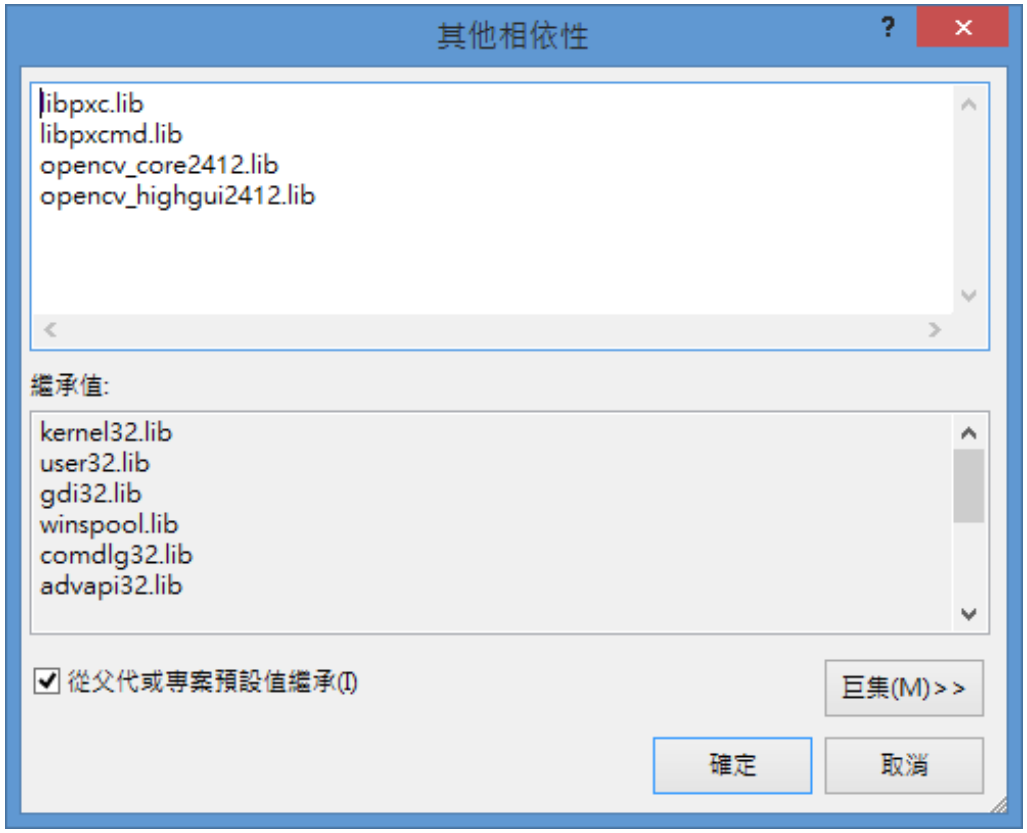
Configuring Project Setting - lib



Configuring Project Setting - lib



Configuring Project Setting - lib



Configuring Project Setting

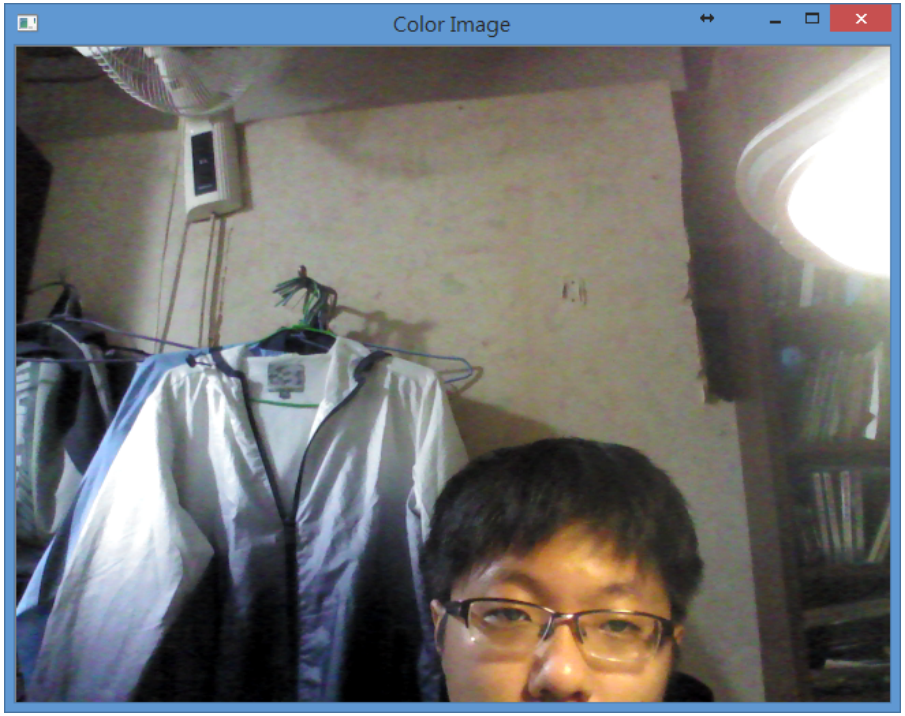
```
#pragma once
#include <iostream>
#include "pxcsensemanager.h"
#include "pxcprojection.h"
#include <opencv2/core/core.hpp>
#include <opencv2/highgui/highgui.hpp>
#include "VECTOR3D.h"
```

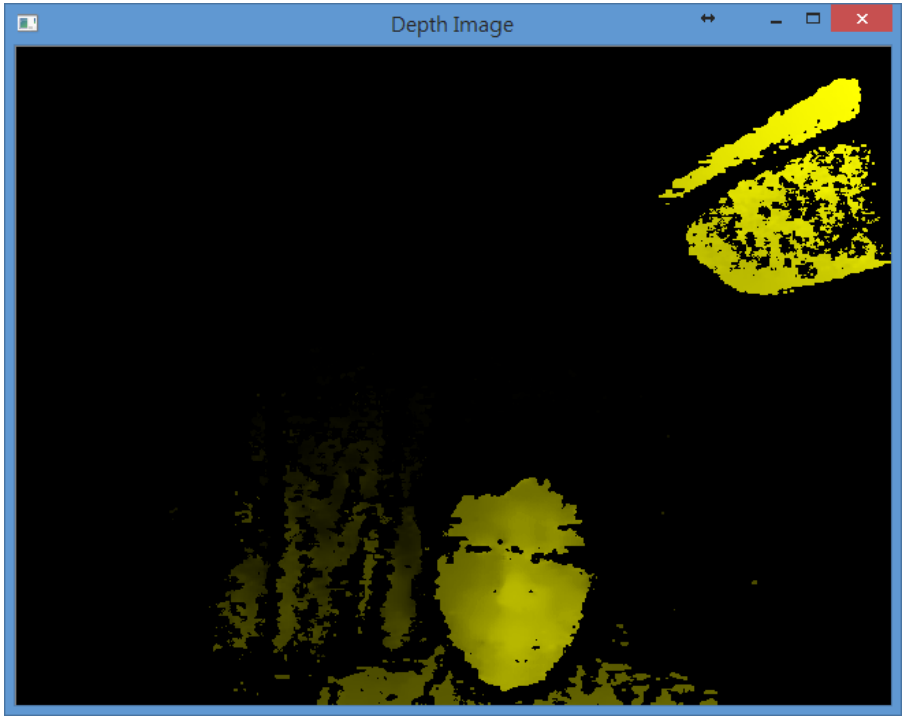
RealSense

OpenCV

Camera Viewer

- 取得1張 Color Image 及 1張 Depth Image, 並用 OpenCV 畫出來





Camera Viewer

- main

```
int main(int argc, char *argv[])
{
  RealSense mRealSense = RealSense();
  mRealSense.showVideoSteams();
  return 0;
}
```

Camera Viewer

- showVideoStreams() ：
  - 透過 OpenCV 顯示 Video Stream

```
void RealSense::showVideoSteams(VIDEO_STREAM_MODE mVideoStreamMode/* = COLOR_DEPTH_MODE*/)
{
  // 開啟 Video Stream
  startVideoStream(mVideoStreamMode);


  while (true)
  {
    PXCCapture::Sample *sample;
    cv::Mat mColorImg, mDepthImg, mColoredDepthImg2Show;


    // 取得 1 張 Video Frame (包括: 1 張 PXCImage 格式的 Color Image 和 1 張 PXCImage 格式的 Depth Image)
    sample = getVideoFrame();


    if (mVideoStreamMode == COLOR_MODE || mVideoStreamMode == COLOR_DEPTH_MODE)
    {
      PXCImg2OpenCVImg(sample->color, mColorImg, STREAM_COLOR);  // 將 PXCImage 格式的 Color Image 轉成 OpenCV 格式的 Color Image
      cv::imshow("Color Image", mColorImg);  // 將 Color Image 利用 OpenCV 顯示出來
    }


    if (mVideoStreamMode == DEPTH_MODE || mVideoStreamMode == COLOR_DEPTH_MODE)
    {
      PXCImg2OpenCVImg(sample->depth, mDepthImg, STREAM_DEPTH);  // 將 PXCImage 格式的 Depth Image 轉成 OpenCV 格式的 Depth Image
      getColoredDepthImg2Show(mDepthImg, mColoredDepthImg2Show);  // 將原始的 Depth Image 轉成彩色版可顯示的 Image
      //getGrayedDepthImg2Show(mDepthImg, mColoredDepthImg2Show);  // 將原始的 Depth Image 轉成灰階版可顯示的 Image
      cv::imshow("Depth Image", mColoredDepthImg2Show);  // 將 Depth Image 利用 OpenCV 顯示出來
    }


    mPXCSenseManager->ReleaseFrame();  // 移除 Video Frame


    // 若使用者按下 "q" 鍵, "ENTER" 鍵 或是 "SPACE" 鍵, 就結束顯示 Color Image 和 Depth Image
    int keyIndex = cv::waitKey(40);
    if (keyIndex == 'q' || keyIndex == 0xD || keyIndex == 0x20)
      break;
  }


  // 停止 Video Stream
  stopVideoStream();
}
```

Camera Viewer

- startVideoStream() :
  - 對 RealSense 做一些初始化的動作, 並開啟 Video Stream

```
void RealSense::startVideoStream(VIDEO_STREAM_MODE mVideoStreamMode/* = COLOR_DEPTH_MODE*/)
{
  // 創建一個 PXCSenseManager Instance
  mPXCSenseManager = PXCSenseManager::CreateInstance();
  while (mPXCSenseManager == NULL)
  {
    std::cout << "Create the PXCSenseManager Failed !!" << std::endl;
    mPXCSenseManager = PXCSenseManager::CreateInstance();
  }


  // 設定 World Coordinate System 類型
  mPXCSenseManager->QuerySession()->SetCoordinateSystem(PXCSession::COORDINATE_SYSTEM_FRONT_DEFAULT);


  // 設定要開啟哪一個 Video Stream 及相關參數
  imgWidth = 640;
  imgHeight = 480;
  videoStreamFPS = 30;
  if (mVideoStreamMode == COLOR_MODE || mVideoStreamMode == COLOR_DEPTH_MODE)
    mPXCSenseManager->EnableStream(PXCCapture::STREAM_TYPE_COLOR, imgWidth, imgHeight, videoStreamFPS);

  if (mVideoStreamMode == DEPTH_MODE || mVideoStreamMode == COLOR_DEPTH_MODE)
    mPXCSenseManager->EnableStream(PXCCapture::STREAM_TYPE_DEPTH, imgWidth, imgHeight, videoStreamFPS);


  // 開啟 Video Stream
  if(mPXCSenseManager->Init() != PXC_STATUS_NO_ERROR)
    std::cout << "Init the PXCSenseManager Failed !!" << std::endl;


  // 跟 Depth Vdieo Stream 有關的東西
  if (mVideoStreamMode == DEPTH_MODE || mVideoStreamMode == COLOR_DEPTH_MODE)
  {
    getDepthMaxVal();  // 計算 Depth Maximum Value
    mPXCProjection = mPXCSenseManager->QueryCaptureManager()->QueryDevice()->CreateProjection();  // 創建一個 Projection Insatnce
  }
}
```

Camera Viewer

- stopVideoStream() ：
  - 停止 Video Stream

```
void RealSense::stopVideoStream()
{
  mPXCSenseManager->Close();  // 停止 Video Stream
  mPXCSenseManager->Release();  // 移除 PXCSenseManager Instance
}
```

Camera Viewer

- getVideoFrame() :
  - 取得 1 張 Video Frame

```
PXCCapture::Sample* RealSense::getVideoFrame()
{
  // 取得 1 張 Video Frame
  mPXCSenseManager->AcquireFrame();
  return mPXCSenseManager->QuerySample();  // 回傳取得的 Video Frame (Sample 格式)
}
```

Camera Viewer

- PXCImg2CVImg() ：
  - 將 1 張 PXCImage 格式的 Image 轉成 OpenCV 格式的 Image
  - 依據輸入參數 (mVideoStreamType) 不同, 可轉換 Color Image 或 Depth Image

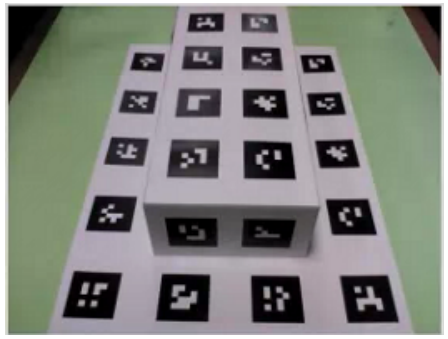
```
void RealSense::PXCImg2OpenCVImg(PXCImage *mPXCImg, cv::Mat &mOpenCVImg, VIDEO_STREAM_TYPE mVideoStreamType)
{
  PXCImage::ImageData data;
  int width = mPXCImg->QueryInfo().width;
  int height = mPXCImg->QueryInfo().height;
  int mOpenCVImgType;


  if (mVideoStreamType == STREAM_COLOR)
  {
    mPXCImg->AcquireAccess(PXCImage::ACCESS_READ, PXCImage::PIXEL_FORMAT_RGB24, &data);
    mOpenCVImgType = CV_8UC3;
  }
  else if (mVideoStreamType == STREAM_DEPTH)
  {
    mPXCImg->AcquireAccess(PXCImage::ACCESS_READ, PXCImage::PIXEL_FORMAT_DEPTH, &data);
    mOpenCVImgType = CV_16UC1;
  }


  mOpenCVImg = cv::Mat(cv::Size(width, height), mOpenCVImgType, data.planes[0]);


  mPXCImg->ReleaseAccess(&data);
}
```

Two Examples

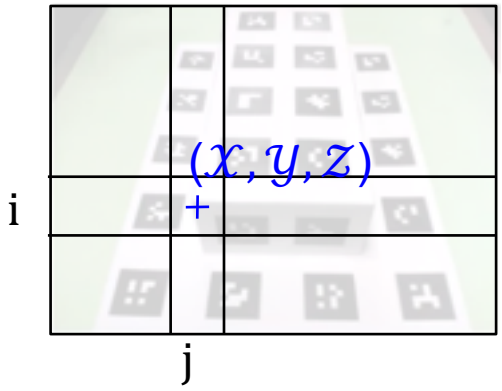
- Configuring Project Setting
- Camera Viewer
- 3D Data Acquisition (an Example)

3D Data Acquisition (an Example)

- 取得1張 Color Image 及 1張 Color-Aligned 3D Coordinate Image
- 包含以下步驟：
  1. Capture Color & Depth Images
  2. Noise Reduction
  3. Registration of Color & Depth Images
  4. Coordinate Transformation

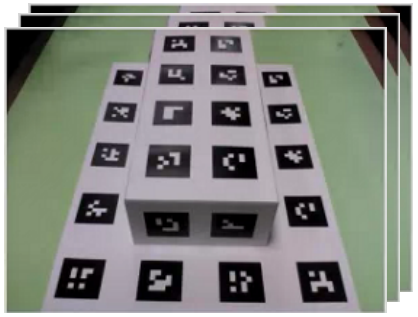


Color Image

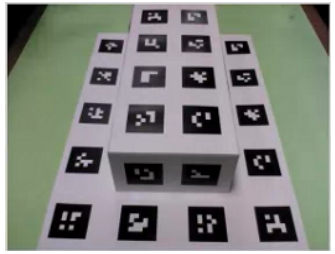


Color-Aligned
3D Coordinate Image

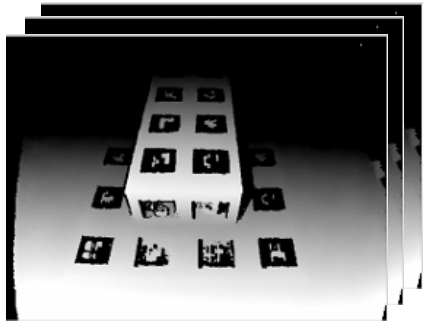
Flow Chart of 3D Data Acquisition (an Example)



Color Images



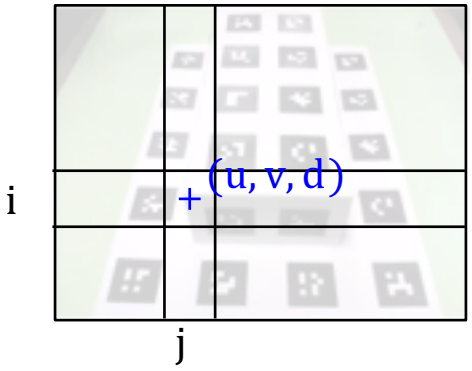
Color Image



Depth Images

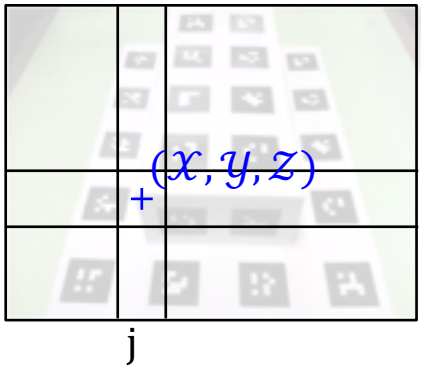
Median Depth Image

F_1



Color-Aligned Depth Image

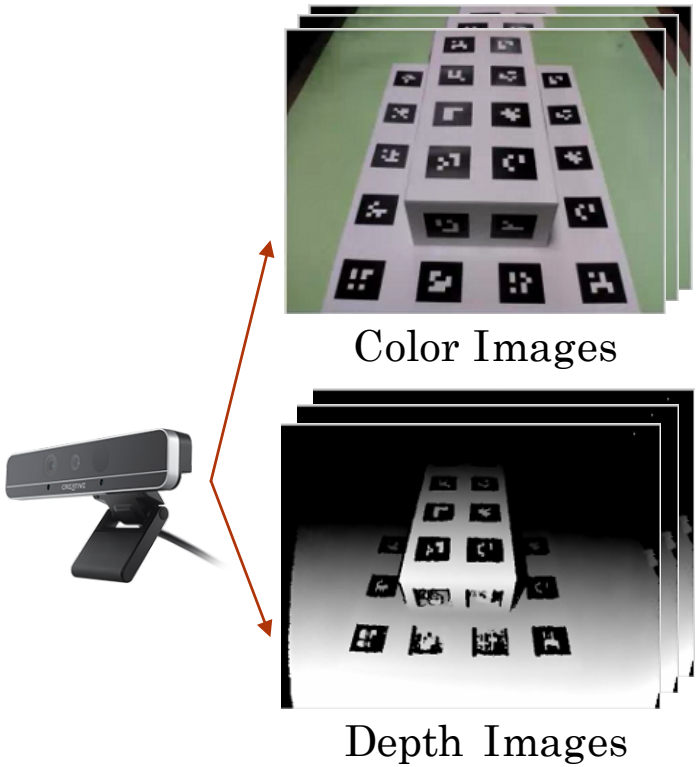
F_2



Color-Aligned 3D Coordinate Image

1. Capture Color & Depth Images

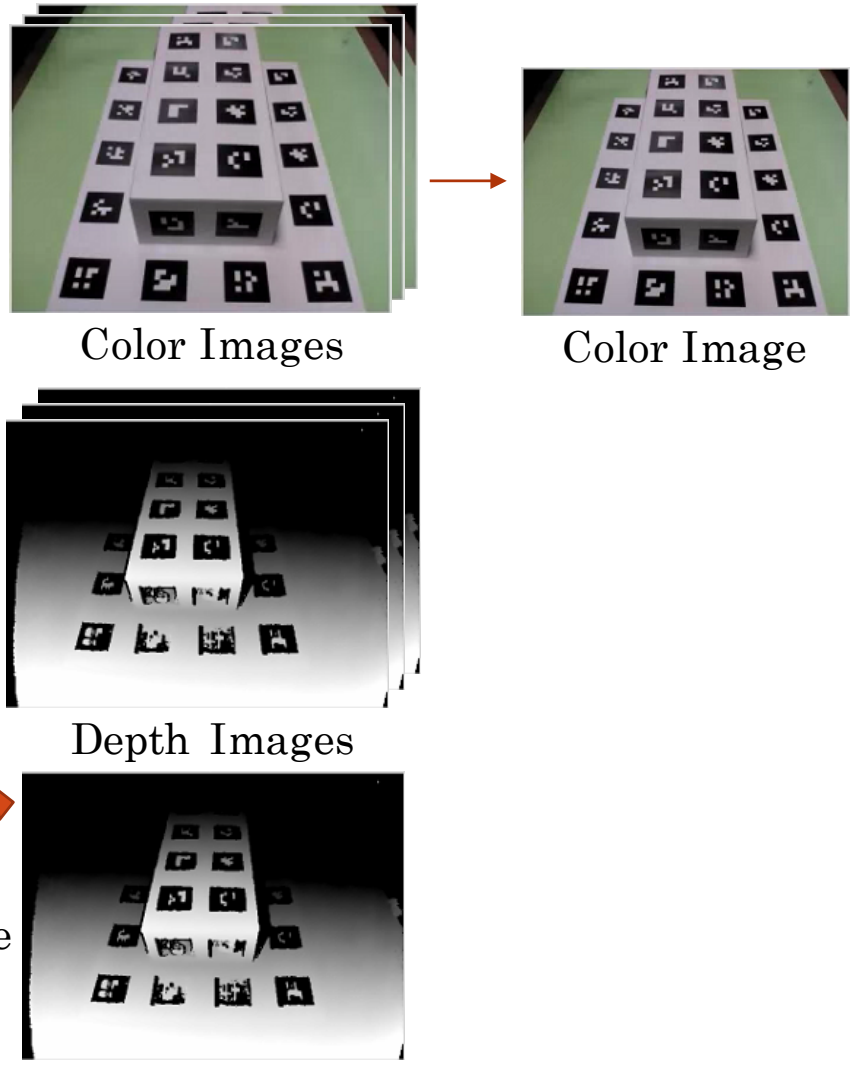
- 取得20組 color images 和 depth images



2. Noise Reduction

- 取20張 Color Images 中的第1張作為所要的 Color Image

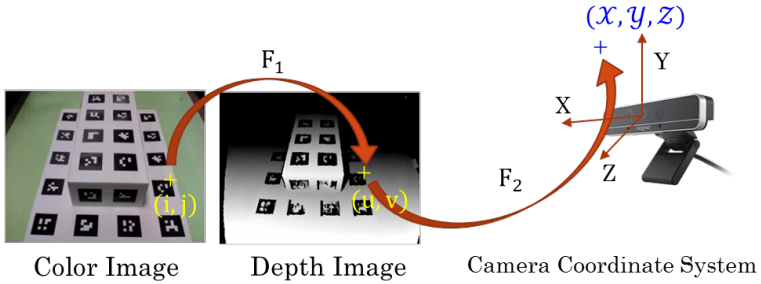
- 針對 depth image 上每個點的 depth 值分別取 20個值中的"中位數"當作所要的值, 以取得一張新的 depth image, 並紀錄每個 depth 值採用的是哪張 image

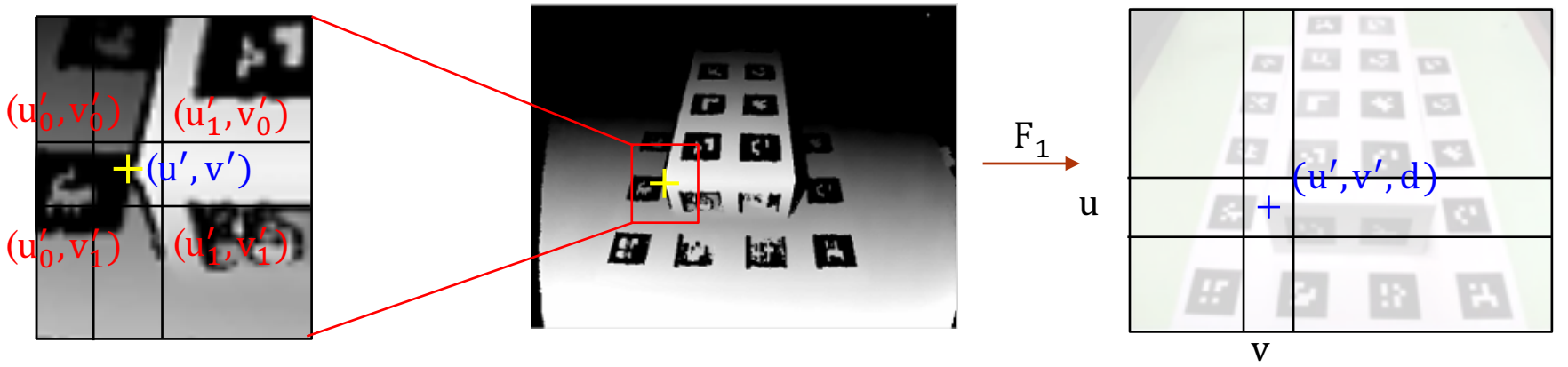


3. Registration of Color & Depth Images

- F_1: pxcStatus QueryInvUVMap(PXCImage *depth, PXCPointF32 *inv_uvmap)

$u'_{\%}$: percentage of width
$v'_{\%}$: percentage of height
$u' = u'_{\%} *$ width
$v' = v'_{\%} *$ height





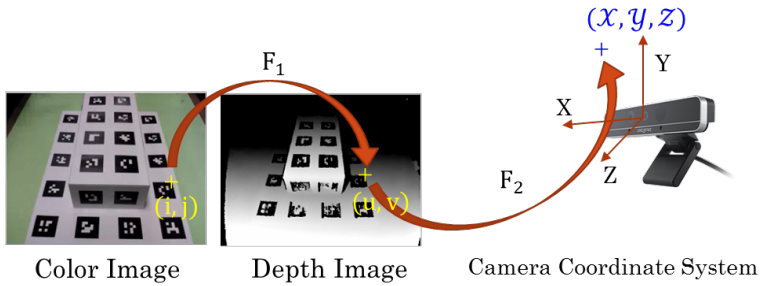
Bilinear Interpolation :

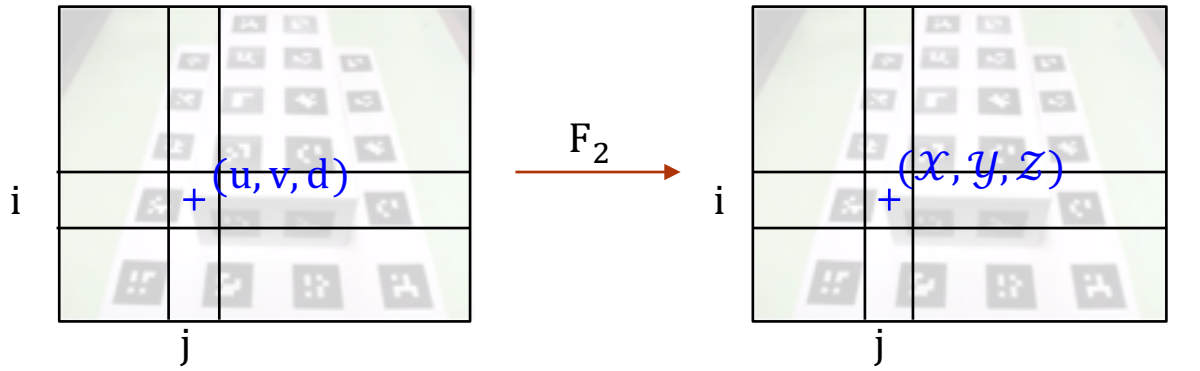
$$f(u',v') = \frac{f(u'_0,v'_0)(u'_1-u')(v'_1-v')+f(u'_0,v'_1)(u'-u'_0)(v'_1-v')+f(u'_1,v'_0)(u'_1-u')(v'-v'_0)+f(u'_1,v'_1)(u'-u'_0)(v'-v'_0)}{(u'_1-u'_0)(v'_1-v'_0)}$$

$f(u',v')$: depth value of pixel (u',v') on the depth image

3. Coordinate Transformation

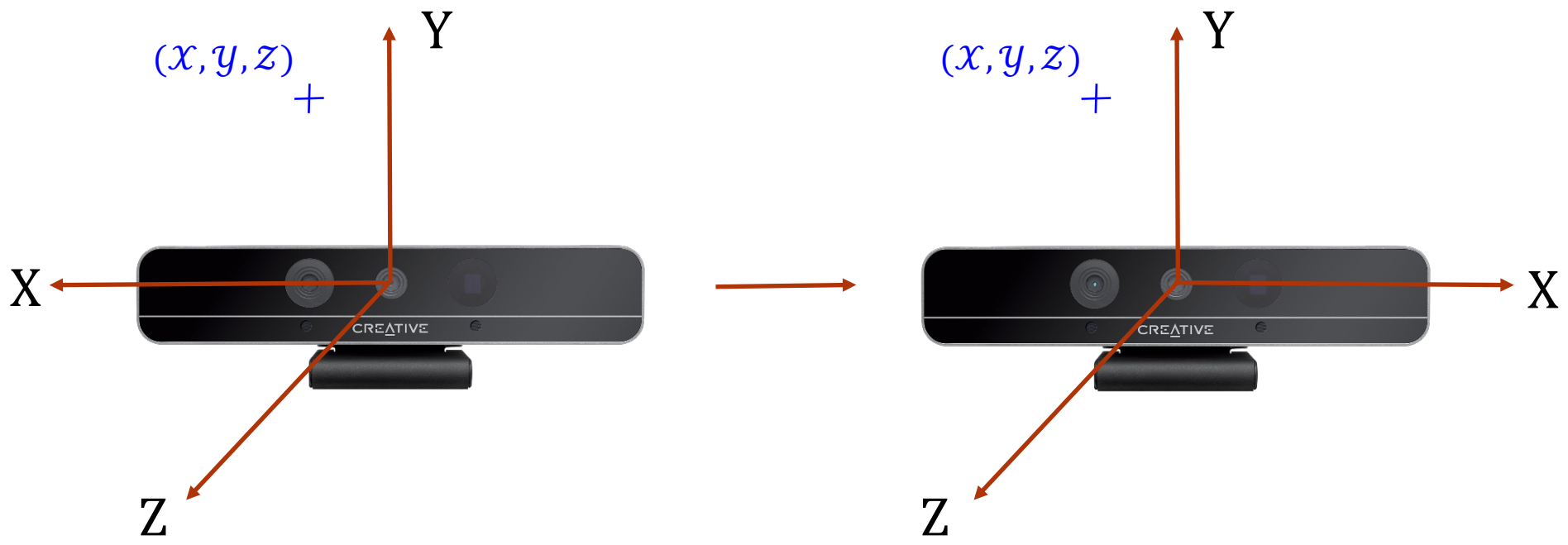
- F_2: pxcStatus ProjectDepthToCamera(pxcI32 npoints, PXCPoint3DF32 *pos_uvz, PXCPoint3DF32 *pos3d)





3. Coordinate Transformation

- 將所有的 camera coordinates 從左手座標系轉換成右手座標系
  - 方法: 將座標值 x 變號



3D Data Acquisition (an Example) - Program

```
int main(int argc, char *argv[])
{
  cv::Mat mColorImg;
  std::vector<POINT3D> m3DCoords;

  RealSense mRealSense = RealSense();
  mRealSense.captureImgs(20, RealSense::COLOR_DEPTH_MODE);
  mRealSense.getColorImgColorAligned3DCoords(mColorImg, m3DCoords);

  // 存檔
  saveColorImg(mColorImg);
  savemColorAligned3DCoords(m3DCoords);


  return 0;
}
```

3D Data Acquisition (an Example) - Program

```
void RealSense::getColorImgColorAligned3DCoords(cv::Mat &mColorImg, std::vector<POINT3D> &mColorAligned3DCoords)
{
  std::vector<cv::Mat> mAllTheTimesCVDepthImgs;
  cv::Mat mMedianCVDepthImg;
  PXCImage *mDepthImg;
  cv::Mat mColorAlignedDepthImg;


  PXCImg2OpenCVImg(mAllTheTimesPXCColorImgs.front(), mColorImg, STREAM_COLOR);
  getAllTheTimesCVDepthImgs(mAllTheTimesCVDepthImgs);
  getMedianDepthImg(mAllTheTimesCVDepthImgs, mMedianCVDepthImg);
  mDepthImg = OpenCVImg2PXCImg(mMedianCVDepthImg, STREAM_DEPTH);
  convertPXCDepthImg2ColorAlignedDepthImg(mDepthImg, mColorAlignedDepthImg);
  convertColorAlignedDepthImg2ColorAligned3DCoords(mColorAlignedDepthImg, mColorAligned3DCoords);
  convertCoordSysLeft2Right(mColorAligned3DCoords);  // 將 Color-Aligend 3D Coordinates 由 左手座標系 -> 右手座標系
}
```

1. Capture Color & Depth Images

```
bool RealSense::captureImgs(int imgCapTimes, VIDEO_STREAM_MODE mVideoStreamMode/* = COLOR_DEPTH_MODE*/)
{
  if (mVideoStreamMode == COLOR_MODE || mVideoStreamMode == COLOR_DEPTH_MODE)
  {
    for (int i = 0; i < mAllTheTimesPXCColorImgs.size(); i++)
      mAllTheTimesPXCColorImgs[i]->Release();
    mAllTheTimesPXCColorImgs.clear();
  }

  if (mVideoStreamMode == DEPTH_MODE || mVideoStreamMode == COLOR_DEPTH_MODE)
  {
    for (int i = 0; i < mAllTheTimesPXCDepthImgs.size(); i++)
      mAllTheTimesPXCDepthImgs[i]->Release();
    mAllTheTimesPXCDepthImgs.clear();
  }


  // 開啟 Video Stream
  startVideoStream(mVideoStreamMode);


  // if open color video stream, delay for color camera adjusting exposure automatically.
  if (mVideoStreamMode == COLOR_MODE || mVideoStreamMode == COLOR_DEPTH_MODE)
    Sleep(350);
```

1. Capture Color & Depth Images

```
// 取像 (imgCapTimes 次)
for (int i = 0; i < imgCapTimes; i++)
{
  std::cout << "Capture the " << i << "th Image." << std::endl;
  PXCCapture::Sample *sample;


  // 取得 1 張 Video Frame (包括: 1 張 PXCImage 格式的 Color Image 和 1 張 PXCImage 格式的 Depth Image)
  sample = getVideoFrame();
  if (sample == NULL ||
      ((mVideoStreamMode == COLOR_MODE || mVideoStreamMode == COLOR_DEPTH_MODE) && !sample->color) ||
      ((mVideoStreamMode == DEPTH_MODE || mVideoStreamMode == COLOR_DEPTH_MODE) && !sample->depth))
  {
    i--;
    continue;
  }
  else
  {
    if (mVideoStreamMode == COLOR_MODE || mVideoStreamMode == COLOR_DEPTH_MODE)
      mAllTheTimesPXCColorImgs.push_back(getPXCImgColorCopy(sample->color));


    if (mVideoStreamMode == DEPTH_MODE || mVideoStreamMode == COLOR_DEPTH_MODE)
      mAllTheTimesPXCDepthImgs.push_back(getPXCImgDepthCopy(sample->depth));
  }


  mPXCSenseManager->ReleaseFrame();
}


// 停止 Video Stream
stopVideoStream();
return true;
}
```

2. Noise Reduction

```
void RealSense::getMedianDepthImg(std::vector<cv::Mat> mAllTheTimesCVDepthImgs, cv::Mat &mMedianCVDepthImg)
{
  mMedianCVDepthImg = cv::Mat(mAllTheTimesCVDepthImgs.front().rows, mAllTheTimesCVDepthImgs.front().cols, CV_16UC1);


  // 對 Depth Image 上的每個點取中位數
  for (int i = 0; i < mAllTheTimesCVDepthImgs.front().rows; i++)
  {
    for (int j = 0; j < mAllTheTimesCVDepthImgs.front().cols; j++)
    {
      std::vector<DepthElement> mDepthElements;


      // 準備用來取中位數的 Array
      for (int k = 0; k < mAllTheTimesCVDepthImgs.size(); k++)
      {
        if (mAllTheTimesCVDepthImgs[k].at<ushort>(i, j) > 0.0f)  // Depth Value 要 > 0.0f 才取
        {
          DepthElement mDepthElement = DepthElement(k, mAllTheTimesCVDepthImgs[k].at<ushort>(i, j));
          mDepthElements.push_back(mDepthElement);
        }
      }


      if (mDepthElements.size() == 0)
      {
        mMedianCVDepthImg.at<ushort>(i, j) = 0;
        mMedianCVDepthImg.at<ushort>(i, j) = 0;
        mMedianCVDepthImg.at<ushort>(i, j) = 0;
      }
      else
      {
        DepthElement mMedianDepthElement;
        getMedianUShortElement(mDepthElements, mMedianDepthElement);  // 取中位數 UShortElement
        mMedianCVDepthImg.at<ushort>(i, j) = mMedianDepthElement.depthVal;
      }
    }
  }
}
```

3. Registration of Color & Depth Image

```
void RealSense::convertPXCDepthImg2ColorAlignedDepthImg(PXCImage* mPXCDepthImg, cv::Mat &mColorAlignedDepthImg)
{
  cv::Mat mDepthImg;
  mColorAlignedDepthImg = cv::Mat(imgHeight, imgWidth, CV_32FC3);
  std::vector<PXCPointF32> mInverseUVMap;


  PXCImg2OpenCVImg(mPXCDepthImg, mDepthImg, STREAM_DEPTH);


  // 透過 RealSense SDK 的 QueryInvUVMap(), 取得 Inverse UV Map
  mInverseUVMap.resize(imgWidth*imgHeight, PXCPointF32());
  mPXCProjection->QueryInvUVMap(mPXCDepthImg, &mInverseUVMap.front());


  for (int i = 0; i < mColorAlignedDepthImg.rows; i++)
  {
    for (int j = 0; j < mColorAlignedDepthImg.cols; j++)
    {
      int index = mColorAlignedDepthImg.cols*i + j;


      if (mInverseUVMap[index].x < 0.0f && mInverseUVMap[index].y < 0.0f)
      {
        mColorAlignedDepthImg.at<cv::Vec3f>(i, j)[0] = -1.0f;
        mColorAlignedDepthImg.at<cv::Vec3f>(i, j)[1] = -1.0f;
        mColorAlignedDepthImg.at<cv::Vec3f>(i, j)[2] = -1.0f;
      }
      else
      {
        float depthImg_u = mInverseUVMap[index].x*(float)imgWidth;
        float depthImg_v = mInverseUVMap[index].y*(float)imgHeight;
        mColorAlignedDepthImg.at<cv::Vec3f>(i, j)[0] = depthImg_u;
        mColorAlignedDepthImg.at<cv::Vec3f>(i, j)[1] = depthImg_v;
        mColorAlignedDepthImg.at<cv::Vec3f>(i, j)[2] = (float)getDepthValBilinearInter(mDepthImg, depthImg_u, depthImg_v);  // 透過 Bilinear Interpolation 取得較精確的 Depth Value
      }
    }
  }
}
```

Convert OpenCV Image to PXCImage

```
PXCImage* RealSense::OpenCVImg2PXCImg(cv::Mat mOpenCVImg, VIDEO_STREAM_TYPE mVideoStreamType)
{
  PXCImage *mPXCImg = NULL;
  PXCImage::ImageData data;
  int width = mOpenCVImg.cols;
  int height = mOpenCVImg.rows;


  PXCImage::ImageInfo mImgInfo;
  memset(&mImgInfo, 0, sizeof(mImgInfo));
  mImgInfo.width = width;
  mImgInfo.height = height;
```

```
startVideoStream(COLOR_DEPTH_MODE);
if (mVideoStreamType == STREAM_COLOR)
{
  uchar *colorValues = new uchar[width*height * 3];
  for (int i = 0; i < mOpenCVImg.rows; i++)
  {
    for (int j = 0; j < mOpenCVImg.cols; j++)
    {
      int index = mOpenCVImg.cols*i + j;
      colorValues[index + 0] = mOpenCVImg.at<cv::Vec3b>(i, j)[2];
      colorValues[index + 1] = mOpenCVImg.at<cv::Vec3b>(i, j)[1];
      colorValues[index + 2] = mOpenCVImg.at<cv::Vec3b>(i, j)[0];
    }
  }


  mImgInfo.format = PXCImage::PIXEL_FORMAT_RGB24;
  mPXCImg = mPXCSenseManager->QuerySession()->CreateImage(&mImgInfo);


  mPXCImg->AcquireAccess(PXCImage::ACCESS_WRITE, PXCImage::PIXEL_FORMAT_RGB24, &data);
  memcpy(data.planes[0], colorValues, sizeof(uchar)*width*height * 3);
  delete[] colorValues;
}
```

Convert OpenCV Image to PXCImage

```
else if (mVideoStreamType == STREAM_DEPTH)
{
  ushort *depthValues = new ushort[width*height];
  for (int i = 0; i < mOpenCVImg.rows; i++)
  {
    for (int j = 0; j < mOpenCVImg.cols; j++)
    {
      int index = mOpenCVImg.cols*i + j;
      depthValues[index] = mOpenCVImg.at<ushort>(i, j);
    }
  }


  mImgInfo.format = PXCImage::PIXEL_FORMAT_DEPTH;
  mPXCImg = mPXCSenseManager->QuerySession()->CreateImage(&mImgInfo);


  mPXCImg->AcquireAccess(PXCImage::ACCESS_WRITE, PXCImage::PIXEL_FORMAT_DEPTH, &data);
  memcpy(data.planes[0], depthValues, sizeof(ushort)*width*height);
  delete[] depthValues;
}


mPXCImg->ReleaseAccess(&data);
stopVideoStream();
return mPXCImg;
}
```

3. Registration of Color & Depth Image

```
ushort RealSense::getDepthValBilinearInter(cv::Mat mDepthImg, float u, float v)
{
  float u0 = (int)u, u1 = (int)u + 1, v0 = (int)v, v1 = (int)v + 1;
  float depthVal_u0_v0 = (float)mDepthImg.at<ushort>(v0, u0);  float depthVal_u0_v1 = (float)mDepthImg.at<ushort>(v1, u0);
  float depthVal_u1_v0 = (float)mDepthImg.at<ushort>(v0, u1);  float depthVal_u1_v1 = (float)mDepthImg.at<ushort>(v1, u1);

  float depthValue;


  if (depthVal_u0_v0 == 0 || depthVal_u0_v1 == 0 || depthVal_u1_v0 == 0 || depthVal_u1_v1 == 1)
    depthValue = depthVal_u0_v0;
  else
  {
    depthValue = (depthVal_u0_v0*(u1 - u)*(v1 - v) +
                  depthVal_u0_v1*(u1 - u)*(v - v0) +
                  depthVal_u1_v0*(u - u0)*(v1 - v) +
                  depthVal_u1_v1*(u - u0)*(v - v0)) / ((u1 - u0)*(v1 - v0));
  }


  return (ushort)depthValue;
}
```

3. Coordinate Transformation

```
void RealSense::convertColorAlignedDepthImg2ColorAligned3DCoords(cv::Mat mColorAlignedDepthImg, std::vector<POINT3D> &mColorAligned3DCoords)
{
  mColorAligned3DCoords.clear();
  std::vector<PXCPoint3DF32> depthImgPts;


  // 準備 RealSense SDK 的 ProjectDepthToCamera() 需要的 Data
  for (int i = 0; i < mColorAlignedDepthImg.rows; i++)
  {
    for (int j = 0; j < mColorAlignedDepthImg.cols; j++)
    {
      PXCPoint3DF32 depthImgPt = { (float)j, (float)i, mColorAlignedDepthImg.at<cv::Vec3f>(i, j)[2] };
      depthImgPts.push_back(depthImgPt);
    }
  }


  // 透過 RealSense SDK 的 ProjectDepthToCamera(), 取得 Camera Coordinate
  PXCPoint3DF32 invalidPt3D = { -1.0f, -1.0f, -1.0f };
  std::vector<PXCPoint3DF32> m3DCoords(depthImgPts.size(), invalidPt3D);
  mPXCProjection->ProjectColorToCamera(depthImgPts.size(), &depthImgPts.front(), &m3DCoords.front());


  for (int i = 0; i < m3DCoords.size(); i++)
    mColorAligned3DCoords.push_back(POINT3D(m3DCoords[i].x, m3DCoords[i].y, m3DCoords[i].z));
}
```

3. Coordinate Transformation

```
void RealSense::convertCoordSysLeft2Right(std::vector<POINT3D> &m3DCoords)
{
  for (int i = 0; i < m3DCoords.size(); i++)
    if (m3DCoords[i].x != -1.0f || m3DCoords[i].y != -1.0f || m3DCoords[i].z != -1.0f)
      m3DCoords[i].x = -m3DCoords[i].x;
}
```

Resources

- RealSense (F200) Developer Zone
  https://software.intel.com/en-us/RealSense/F200Camera
- Intel® RealSense™ SDK 載點
  https://software.intel.com/en-us/intel-realsense-sdk/download
- Intel® RealSense™ SDK Documentation
  https://software.intel.com/sites/landingpage/realsense/camera-sdk/v1.1/documentation/html/index.html?docm_legal_information.html
- Intel® RealSense™ SDK Forum
  https://software.intel.com/en-us/Forums/realsense