Kinect Server Data Acquisition:
Compilation and Running Requirements and Directions

Vahid Soleimani, Majid Mirmehdi, Dima Damen, Sion Hannuna, Massimo Camplani

This document introduces hardware and software resources required for compiling and running the source code of “Kinect data Acquisition” system [1]. It also explains how to establish the necessary libraries dependences to be able to successfully compile and run the source code. Further, the software specifications and details, and the format of its output data are described in this document.

1- Requirements and Dependencies

The following hardware, software and libraries are required to successfully compile and run the source code.

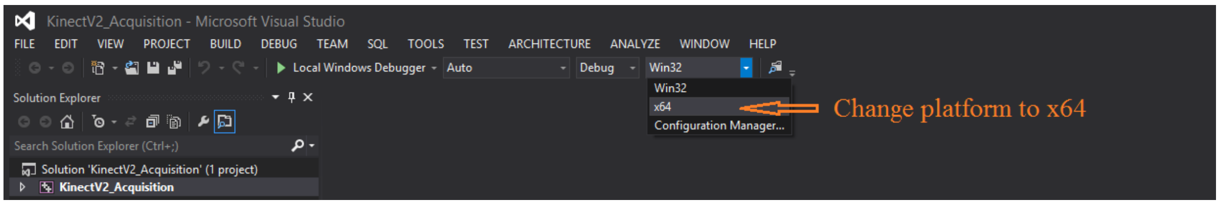
- A second generation of Kinect for Windows
- A Kinect−compatible Microsoft Windows (8.1 or 10) PC or laptop
- Microsoft Visual Studio C++
- Kinect for Windows SDK 2.0
- Intel® Threading Building Blocks (TBB)
- OpenCV

According to Microsoft, not all laptops and PCs are compatible with Kinect V2. A detailed information about hardware and software requirements can be found in this page and also here. The Kinect data acquisition source code has been developed and successfully compiled under Microsoft Visual Studio C++ 2012, but you can also use the later versions if the following dependencies have been provided for them. Kinect for Windows SDK 2.0, Intel® Threading Building Blocks and OpenCV are the libraries which are required for compiling and executing the source code. Establishment of these libraries dependencies in Visual Studio 2012 is explained in Section 2.

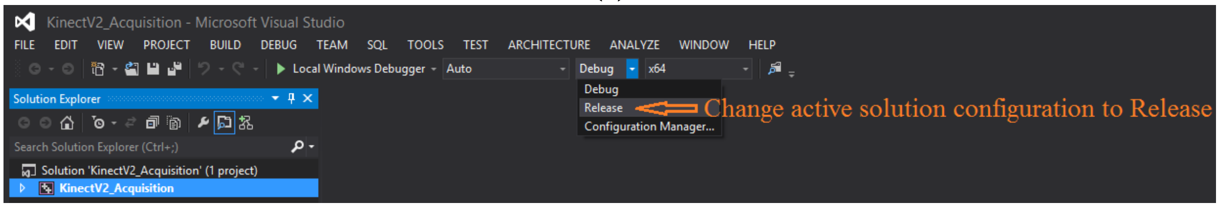
2- Visual Studio Dependencies Establishment

After opening the solution file, provided in the repository, active solution platform and configuration must be changed to x64 and Released, respectively, as shown in Fig. 1.

Kinect for Windows SDK 2.0 – This standard library is provided by Microsoft and is essential for making the Kinect work under Microsoft Windows. It is released under free commercial license and can be downloaded from here. After installation, SDK provides a default environment variable named *KINECTSDK20_DIR*, which is used to add the SDK include and library dependencies to Visual Studio. First, the project property page must be opened (Fig. 2a), and then project “Additional Include Directories” page can be accessed through steps 1 to 3 in Fig.2b.

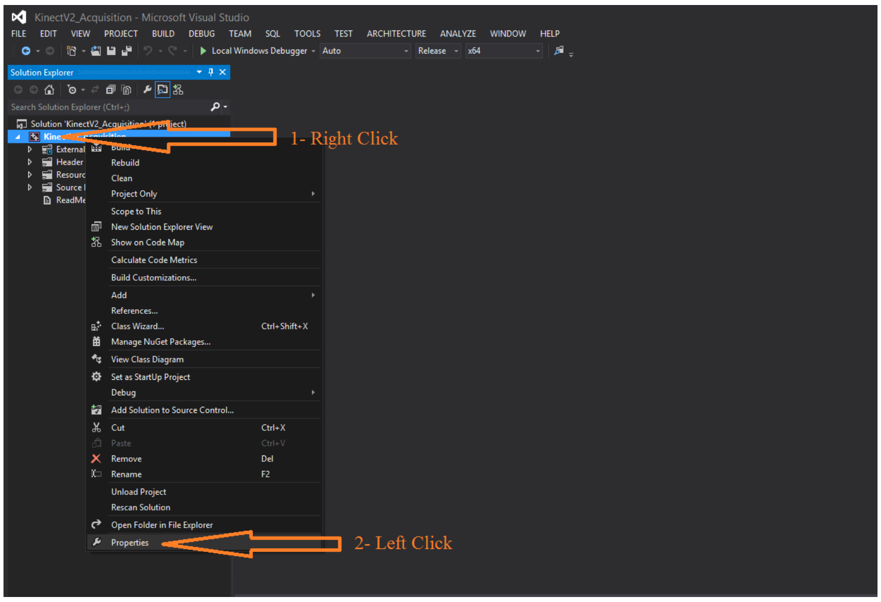

KinectV2_Acquisition - Microsoft Visual Studio
Debug
Win32
x64
Configuration Manager...
Change platform to x64
Solution 'KinectV2_Acquisition' (1 project)
KinectV2_Acquisition

(a)

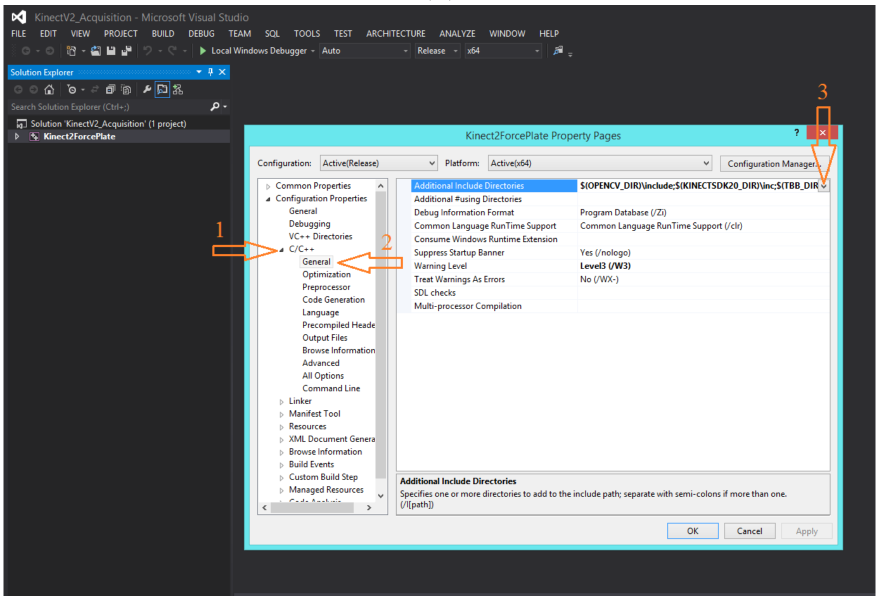

KinectV2_Acquisition - Microsoft Visual Studio
Debug
x64
Release
Configuration Manager...
Change active solution configuration to Release
Solution 'KinectV2_Acquisition' (1 project)
KinectV2_Acquisition

(b)

Fig.1: (a) Change the platform to x64, (b) Changing the active solution configuration to Release.



(a)



Configuration: Active(Release) Platform: Active(x64) Configuration Manager...

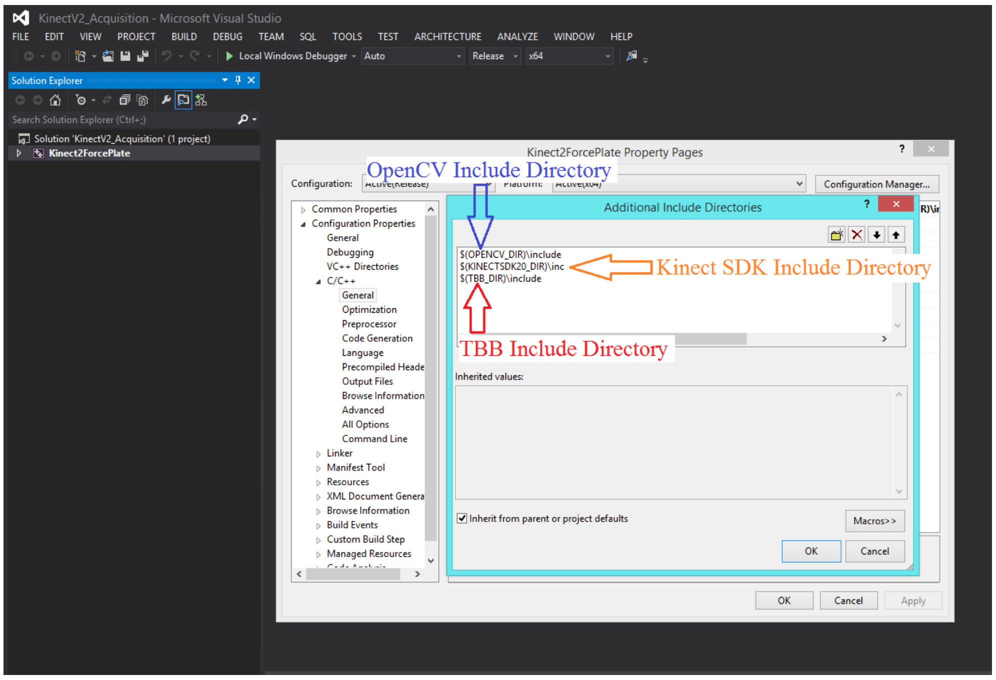
Additional Include Directories	$(OPENCV_DIR)\include;$(KINECTSDK20_DIR)\inc;$(TBB_DIR
Additional #using Directories	
Debug Information Format	Program Database (/Zi)
Common Language RunTime Support	Common Language RunTime Support (/clr)
Consume Windows Runtime Extension	
Suppress Startup Banner	Yes (/nologo)
Warning Level	Level3 (/W3)
Treat Warnings As Errors	No (/WX-)
SDL checks	
Multi-processor Compilation	

Additional Include Directories
Specifies one or more directories to add to the include path; separate with semi-colons if more than one. (/I[path])

(b)

Fig.2: (a) Accessing the “project property page”, (b) Accessing the “Additional Include Directories”.

Any required include directory can be added in “Additional Include Directories” page (Fig. 3), by either using the absolute path or using the environment variable. For the Kinect SDK, the default environment variable is used as shown in Fig. 3.

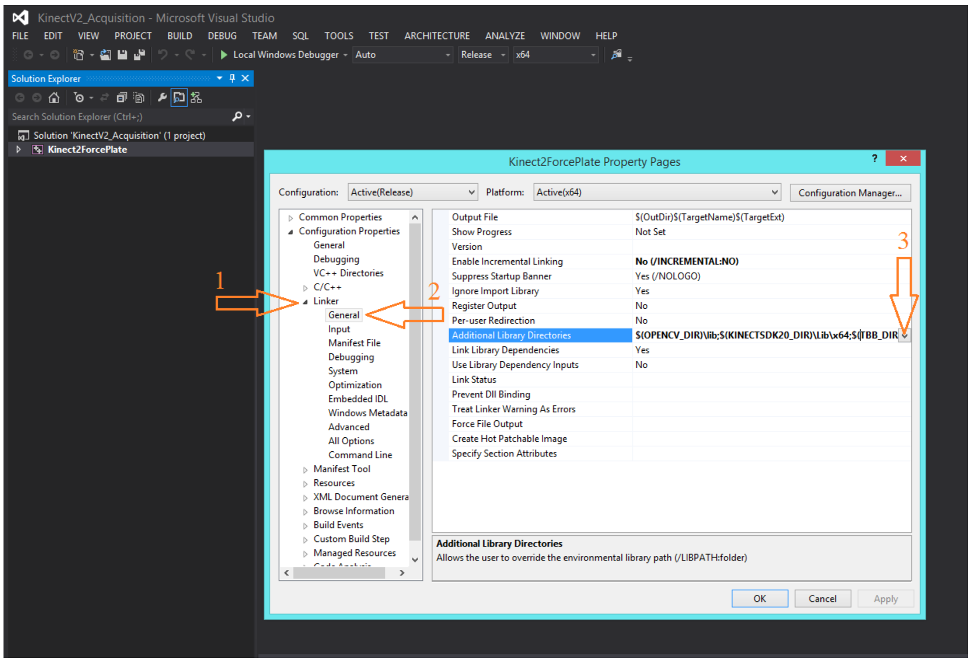

KinectV2_Acquisition - Microsoft Visual Studio
FILE EDIT VIEW PROJECT BUILD DEBUG TEAM SQL TOOLS TEST ARCHITECTURE ANALYZE WINDOW HELP
Local Windows Debugger Auto Release x64
Solution Explorer
Solution 'KinectV2_Acquisition' (1 project)
Kinect2ForcePlate
Kinect2ForcePlate Property Pages
Additional Include Directories
OpenCV Include Directory
Kinect SDK Include Directory
TBB Include Directory
Inherited values:
Inherit from parent or project defaults
Macros>>
OK
Cancel
Apply

Additional Include Directories:

```
$(OPENCV_DIR)\include
$(KINECTSDK20_DIR)\inc
$(TBB_DIR)\include
```

Fig.3: Adding required includes to the project “Additional Include Directories” page.

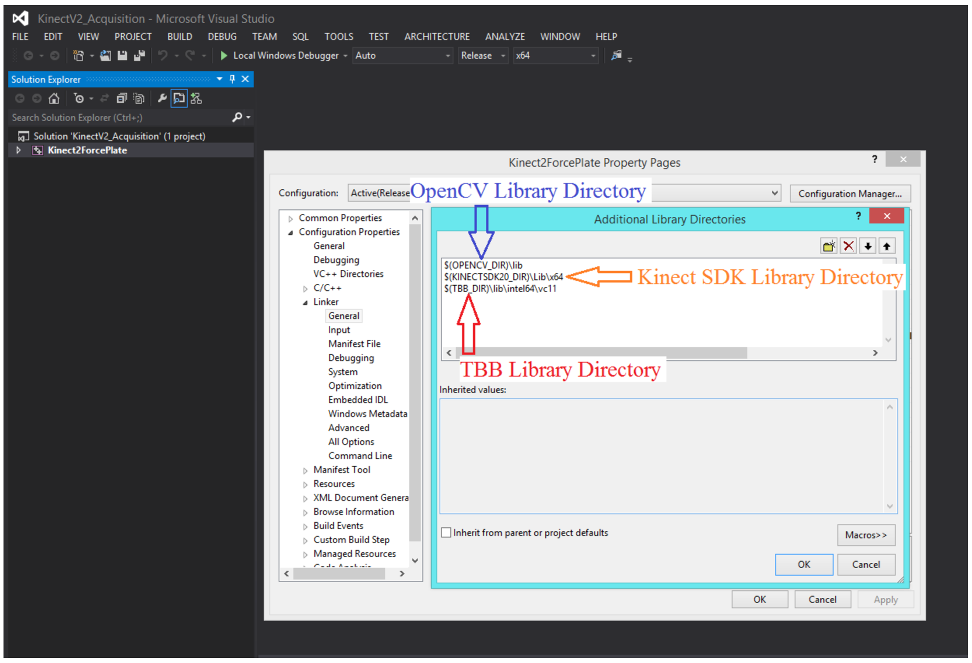
To add Kinect SDK library, first, the library path should be declared at “Additional Library Directories” in the project property page. To do this, after accessing to project property page (Fig. 2a), “Additional Library Directories” is accessible through steps 1 to 3 in Fig. 4a.



Output File	$(OutDir)$(TargetName)$(TargetExt)
Show Progress	Not Set
Version	
Enable Incremental Linking	**No (/INCREMENTAL:NO)**
Suppress Startup Banner	Yes (/NOLOGO)
Ignore Import Library	Yes
Register Output	No
Per-user Redirection	No
Additional Library Directories	**$(OPENCV_DIR)\lib;$(KINECTSDK20_DIR)\Lib\x64;$(TBB_DIR**
Link Library Dependencies	Yes
Use Library Dependency Inputs	No
Link Status	
Prevent Dll Binding	
Treat Linker Warning As Errors	
Force File Output	
Create Hot Patchable Image	
Specify Section Attributes	

Additional Library Directories
Allows the user to override the environmental library path (/LIBPATH:folder)

(a)



Kinect2ForcePlate Property Pages — Additional Library Directories

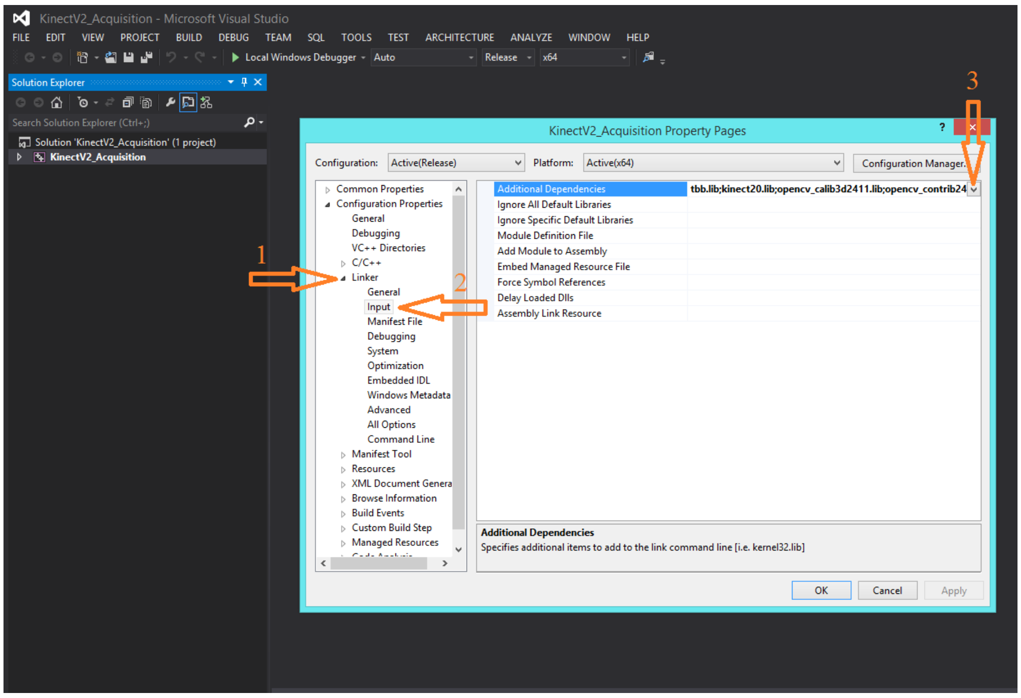
```
$(OPENCV_DIR)\lib
$(KINECTSDK20_DIR)\Lib\x64
$(TBB_DIR)\lib\intel64\vc11
```

(b)

Fig. 4: (a) Accessing “Additional Library Directories”, (b) Adding required library directories.

Then, Kinect SDK library path is added as *$(KINECTSDK20_DIR)\Lib\x64* (Fig. 4b).

Finally, libraries’ name that are used must be added to the “Additional Dependencies”. “Additional Dependencies” page can be accessed through steps 1 to 3 in Fig. 5a. For the Kinect SDK, the only library which must be added is *kinect20.lib*.

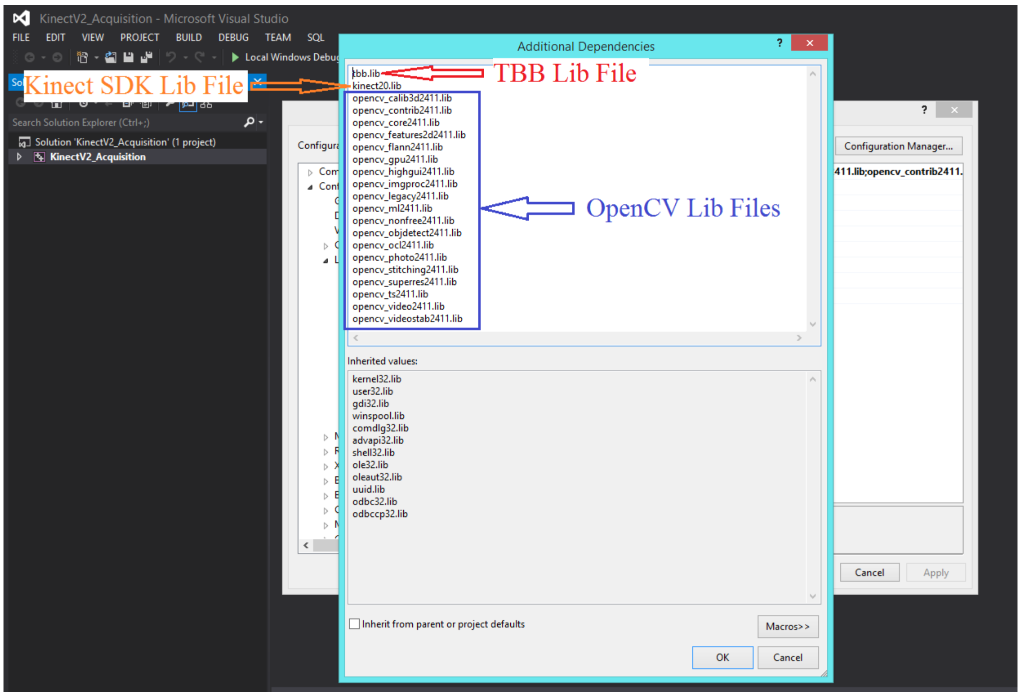


KinectV2_Acquisition Property Pages — Configuration: Active(Release), Platform: Active(x64)

Additional Dependencies: tbb.lib;kinect20.lib;opencv_calib3d2411.lib;opencv_contrib24

Additional Dependencies
Specifies additional items to add to the link command line [i.e. kernel32.lib]

(a)


KinectV2_Acquisition - Microsoft Visual Studio
FILE EDIT VIEW PROJECT BUILD DEBUG TEAM SQL
Local Windows Debug
Kinect SDK Lib File
Search Solution Explorer (Ctrl+;)
Solution 'KinectV2_Acquisition' (1 project)
KinectV2_Acquisition
Additional Dependencies
TBB Lib File
OpenCV Lib Files
Configuration Manager...
Inherit from parent or project defaults
Macros>>
OK
Cancel
Apply

```
tbb.lib
kinect20.lib
opencv_calib3d2411.lib
opencv_contrib2411.lib
opencv_core2411.lib
opencv_features2d2411.lib
opencv_flann2411.lib
opencv_gpu2411.lib
opencv_highgui2411.lib
opencv_imgproc2411.lib
opencv_legacy2411.lib
opencv_ml2411.lib
opencv_nonfree2411.lib
opencv_objdetect2411.lib
opencv_ocl2411.lib
opencv_photo2411.lib
opencv_stitching2411.lib
opencv_superres2411.lib
opencv_ts2411.lib
opencv_video2411.lib
opencv_videostab2411.lib
```

Inherited values:

```
kernel32.lib
user32.lib
gdi32.lib
winspool.lib
comdlg32.lib
advapi32.lib
shell32.lib
ole32.lib
oleaut32.lib
uuid.lib
odbc32.lib
odbccp32.lib
```

(b)

Fig. 5: (a) Accessing "Additional Dependencies", (b) Adding required library files.

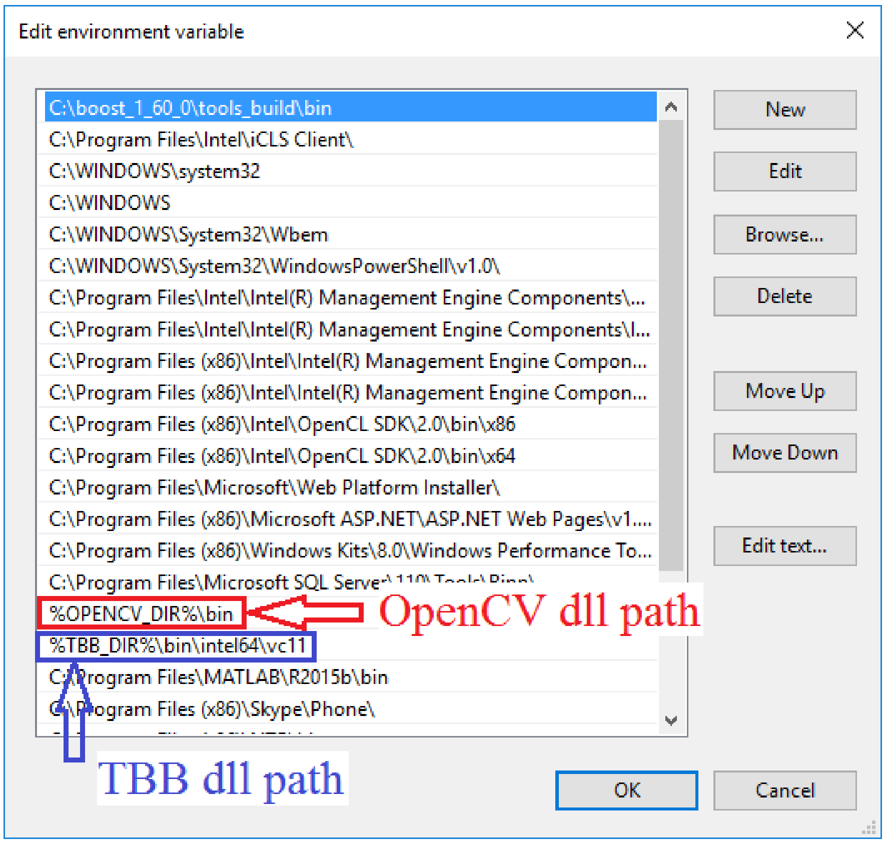
Intel® Threading Building Blocks (TBB) − This library is used to create parallel threads for acquiring, visualising and recording RGB, depth and skeleton data streams separately at full frame rate. This library is licensed as GPLv2. Although any version can be used for our purpose, the version we used and tested is "tbb43_20140724oss" and can be downloaded from here. After extracting, an environment variable should be defined for the main path (e.g. *C:\TBB2014\tbb43_20140724oss*). We named this variable as *TBB_DIR* and used to establish include and library dependencies similar to Kinect SDK. After accessing to project property page (Fig. 2a), *$(TBB_DIR)\include* is added to include directories (Fig.3). $(TBB_DIR)*\lib\intel64\vc11* is then added to the "Additional Library Directory" page using (Fig. 4b). If Visual Studio 2013 is used then *vc11* sub-folder of $(TBB_DIR)*\lib\intel64\vc11* needs to be changed to *vc12*. The only TBB library that is needed here is *tbb.lib* (Fig. 5b).

OpenCV − This is an open source computer vision library, which consists of considerable number of computer vision programming functions and is released under a BSD license. Similar to TBB, any version can be used for our purpose but we tested our code with pre-built libraries version 2.4.11. An official and comprehensive explanation about installing OpenCV on Windows can be found here. The determined environment variable for OpenCV (here named *OPENCV_DIR*) is used to establish the dependencies similar to TBB (Figs. 2−4). However, there are more library files that must be added to library "Additional Dependencies" as shown in Fig. 5b.

Dynamic Link Libraries (DLL) – In order to successfully run the Kinect data acquisition software, path of binary files of third party libraries i.e. Kinect SDK, TBB and OpenCV, must be added to the system path. For the Kinect SDK, it is added automatically by installing the SDK. However, for the TBB and OpenCV, the following paths must be added to the system path as shown in Fig. 6.

- %OPENCV_DIR%\bin
- %TBB_DIR%\bin\intel64\vc11

This can be done by accessing *Environment Variables* from the *System Properties*.


Edit environment variable

C:\boost_1_60_0\tools_build\bin
C:\Program Files\Intel\iCLS Client\
C:\WINDOWS\system32
C:\WINDOWS
C:\WINDOWS\System32\Wbem
C:\WINDOWS\System32\WindowsPowerShell\v1.0\
C:\Program Files\Intel\Intel(R) Management Engine Components\...
C:\Program Files\Intel\Intel(R) Management Engine Components\I...
C:\Program Files (x86)\Intel\Intel(R) Management Engine Compon...
C:\Program Files (x86)\Intel\Intel(R) Management Engine Compon...
C:\Program Files (x86)\Intel\OpenCL SDK\2.0\bin\x86
C:\Program Files (x86)\Intel\OpenCL SDK\2.0\bin\x64
C:\Program Files\Microsoft\Web Platform Installer\
C:\Program Files (x86)\Microsoft ASP.NET\ASP.NET Web Pages\v1....
C:\Program Files (x86)\Windows Kits\8.0\Windows Performance To...
C:\Program Files\Microsoft SQL Serve
%OPENCV_DIR%\bin
%TBB_DIR%\bin\intel64\vc11
C:\Program Files\MATLAB\R2015b\bin
C:\Program Files (x86)\Skype\Phone\

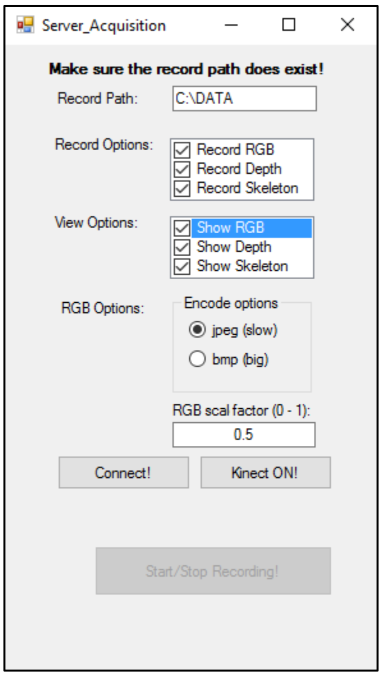
Fig. 6: Adding TBB and OpenCV libraries binary files to the system path.

3- Software and Output Data Specifications

After compiling and building the source code successfully, the application is ready to be used. Fig. 7 shows a sample running of the software.

As can be seen in Fig. 7a, the software is featured the following specifications:

1- Recording path.
2- RGB, depth and skeleton recording options.
3- RGB, depth and skeleton online visualisation options.
4- JPEG or BMP encoding option for saving RGB data frames.
5- RGB scaling factor (0-1) to reduce the RGB image size.
6- Start/Stop button.


Server_Acquisition

Make sure the record path does exist!

Record Path: C:\DATA

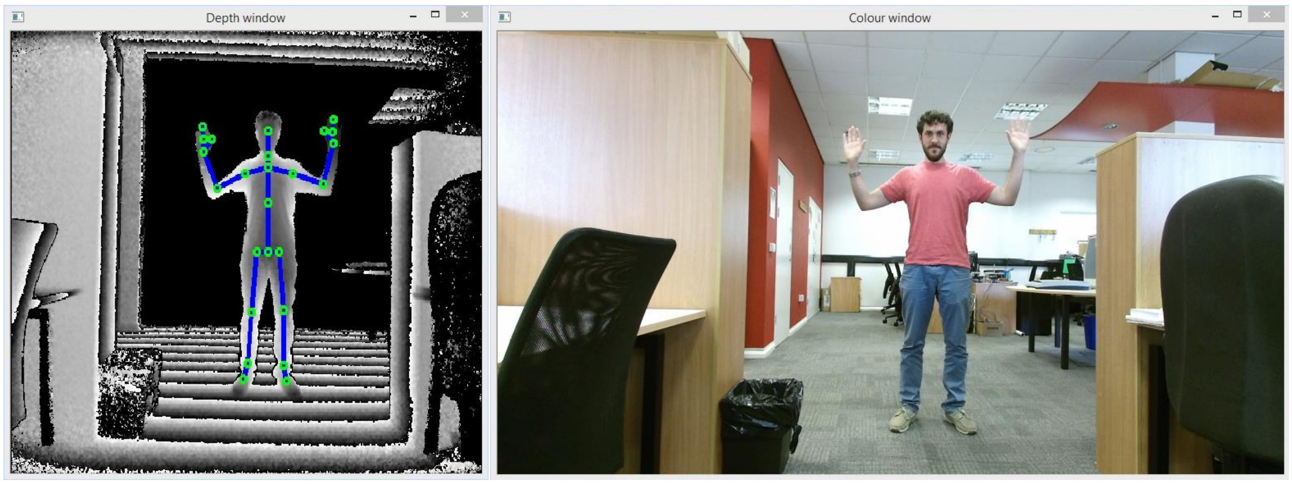
Record Options: [✓] Record RGB [✓] Record Depth [✓] Record Skeleton

View Options: [✓] Show RGB [✓] Show Depth [✓] Show Skeleton

RGB Options: Encode options [● jpeg (slow)] ○ bmp (big)

RGB scal factor (0 - 1): 0.5

(a)



(b)

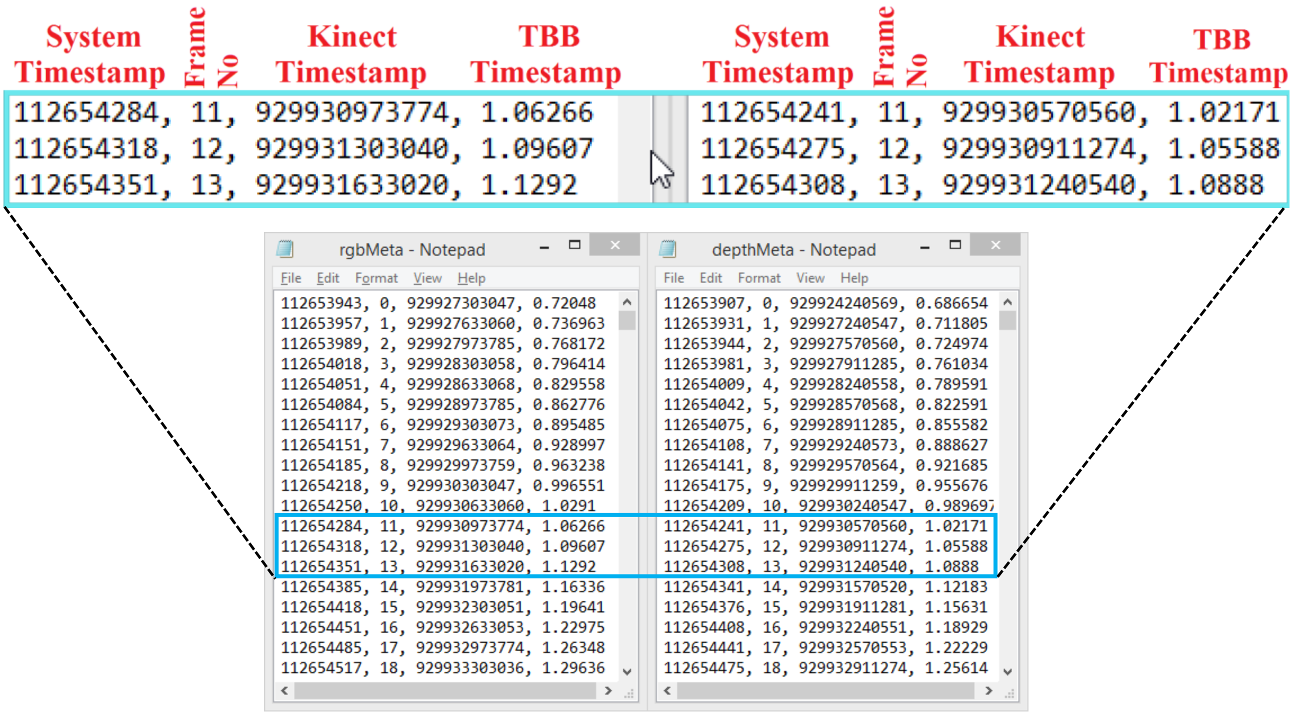
Fig. 7: A sample running of the Kinect data acquisition software.

Fig. 7b presents a sample of online visualisation of RGB, depth and skeleton data. In addition to saving RGB data frames in JPEG or BMP formats, and depth data frames in PNG format, the software also provides separate RGB, depth and skeleton Meta data for the whole captured sequence.

Before data capture starts, server and client machines must be locally networked and their system time must be synchronised using NTP. First, a TCP/IP connection is established at the beginning of capture by pushing the *connect!* button on both server and client applications. Note that the server button must be pressed first. Then, both Kinects must be turned on by pushing *Kinect ON!* buttons. Data acquisition in both server and client machines is starts by pressing the *start/stop Recording!* button of the Server Data Acquisition application.

RGB and depth Meta data – For each sequence, RGB and depth Meta data are stored in two separate text files, named “rgbMeta.txt” and “depthMeta.txt”. These files present the following information for each RGB and depth captured frames:

- system timestamp
- Kinect provided timestamp
- TBB thread timestamp



```
112654284, 11, 929930973774, 1.06266
112654318, 12, 929931303040, 1.09607
112654351, 13, 929931633020, 1.1292
```

```
112654241, 11, 929930570560, 1.02171
112654275, 12, 929930911274, 1.05588
112654308, 13, 929931240540, 1.0888
```

```
112653943, 0, 929927303047, 0.72048
112653957, 1, 929927633060, 0.736963
112653989, 2, 929927973785, 0.768172
112654018, 3, 929928303058, 0.796414
112654051, 4, 929928633068, 0.829558
112654084, 5, 929928973785, 0.862776
112654117, 6, 929929303073, 0.895485
112654151, 7, 929929633064, 0.928997
112654185, 8, 929929973759, 0.963238
112654218, 9, 929930303047, 0.996551
112654250, 10, 929930633060, 1.0291
112654284, 11, 929930973774, 1.06266
112654318, 12, 929931303040, 1.09607
112654351, 13, 929931633020, 1.1292
112654385, 14, 929931973781, 1.16336
112654418, 15, 929932303051, 1.19641
112654451, 16, 929932633053, 1.22975
112654485, 17, 929932973774, 1.26348
112654517, 18, 929933303036, 1.29636
```

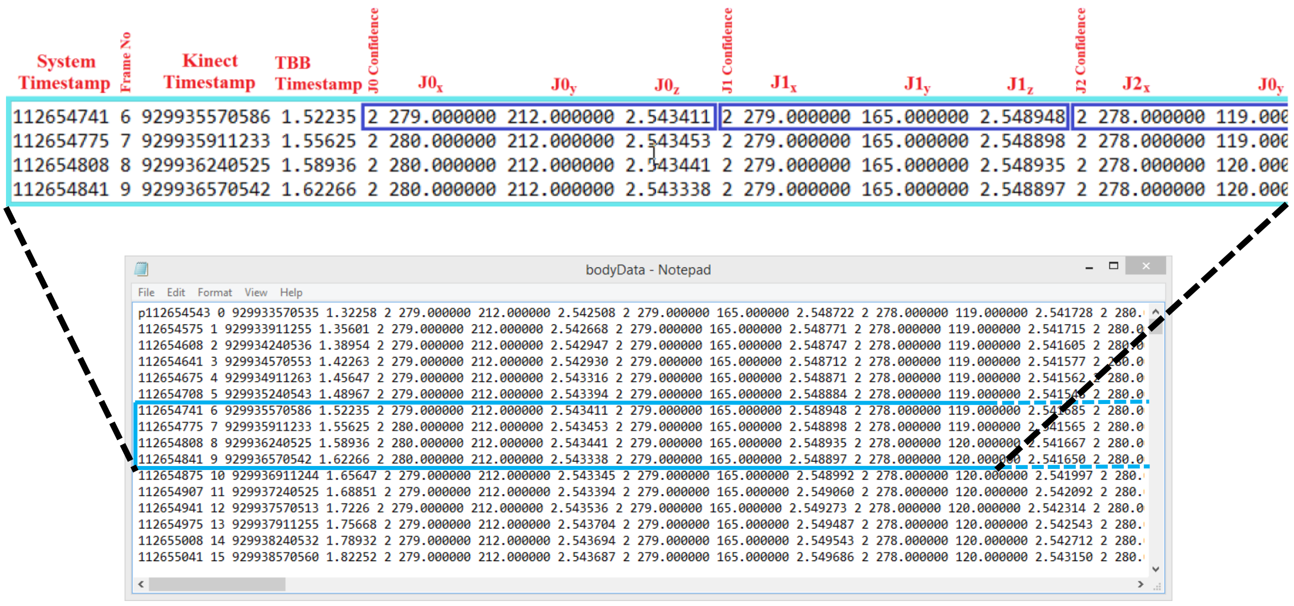
```
112653907, 0, 929924240569, 0.686654
112653931, 1, 929927240547, 0.711805
112653944, 2, 929927570560, 0.724974
112653981, 3, 929927911285, 0.761034
112654009, 4, 929928240558, 0.789591
112654042, 5, 929928570568, 0.822591
112654075, 6, 929928911285, 0.855582
112654108, 7, 929929240573, 0.888627
112654141, 8, 929929570564, 0.921685
112654175, 9, 929929911259, 0.955676
112654209, 10, 929930240547, 0.989697
112654241, 11, 929930570560, 1.02171
112654275, 12, 929930911274, 1.05588
112654308, 13, 929931240540, 1.0888
112654341, 14, 929931570520, 1.12183
112654376, 15, 929931911281, 1.15631
112654408, 16, 929932240551, 1.18929
112654441, 17, 929932570553, 1.22229
112654475, 18, 929932911274, 1.25614
```

Fig. 8: RGB and depth metadata files content.

Fig. 8 shows part of an RGB and depth Meta data files. Each line of these text files correspond to a single captured frame. As can be seen in Fig. 8, the first column is the system timestamp and the second column is the frame number, which is equal to the corresponding the RGB and depth file names. Note that RGB and depth frames must be matched based on the Kinect provided timestamps, not the frame numbers. The third column is Kinect provided timestamp and the last column is the TBB timestamp obtained from the RGB and depth frame acquisition thread.

Skeleton Meta data – If a person appears in the Kinect field of view, their skeleton data is obtained and stored in a text file, named “bodyMeta.txt”. Similar to RGB and depth Meta data, each line is related to the corresponding depth captured frames. In addition to system, Kinect and TBB timestamps that are similar to the RGB and depth Meta data with the same storing order, skeleton body data which includes 25 joints position and their detection confidence, are also stored in the skeleton Meta data file. A quadruple is associated with each joint, in which the first number states the joint tracking confidence. 0 means no joint is detected, 1 means joint is weekly detected and 2 means joint is fully detected. The other three numbers present each joint location, i.e.

(*x*, *y*, *z*), in which *x* and *y* are presented in pixel (to match depth image), and *z* is real world coordinate system (in *meter*). *x* and *y* can be easily converted to real world coordinate using the Kinect depth sensor camera parameters. Fig. 9 shows a part of skeleton Meta data file.



```
112654741 6 929935570586 1.52235 2 279.000000 212.000000 2.543411 2 279.000000 165.000000 2.548948 2 278.000000 119.000
112654775 7 929935911233 1.55625 2 280.000000 212.000000 2.543453 2 279.000000 165.000000 2.548898 2 278.000000 119.000
112654808 8 929936240525 1.58936 2 280.000000 212.000000 2.543441 2 279.000000 165.000000 2.548935 2 278.000000 120.000
112654841 9 929936570542 1.62266 2 280.000000 212.000000 2.543338 2 279.000000 165.000000 2.548897 2 278.000000 120.000
```

```
p112654543 0 929933570535 1.32258 2 279.000000 212.000000 2.542508 2 279.000000 165.000000 2.548722 2 278.000000 119.000000 2.541728 2 280.
112654575 1 929933911255 1.35601 2 279.000000 212.000000 2.542668 2 279.000000 165.000000 2.548771 2 278.000000 119.000000 2.541715 2 280.0
112654608 2 929934240536 1.38954 2 279.000000 212.000000 2.542947 2 279.000000 165.000000 2.548747 2 278.000000 119.000000 2.541605 2 280.0
112654641 3 929934570553 1.42263 2 279.000000 212.000000 2.542930 2 279.000000 165.000000 2.548712 2 278.000000 119.000000 2.541577 2 280.0
112654675 4 929934911263 1.45647 2 279.000000 212.000000 2.543316 2 279.000000 165.000000 2.548871 2 278.000000 119.000000 2.541562 2 280.0
112654708 5 929935240543 1.48967 2 279.000000 212.000000 2.543394 2 279.000000 165.000000 2.548884 2 278.000000 119.000000 2.541548 2 280.0
112654741 6 929935570586 1.52235 2 279.000000 212.000000 2.543411 2 279.000000 165.000000 2.548948 2 278.000000 119.000000 2.541685 2 280.0
112654775 7 929935911233 1.55625 2 280.000000 212.000000 2.543453 2 279.000000 165.000000 2.548898 2 278.000000 119.000000 2.541565 2 280.0
112654808 8 929936240525 1.58936 2 280.000000 212.000000 2.543441 2 279.000000 165.000000 2.548935 2 278.000000 120.000000 2.541667 2 280.0
112654841 9 929936570542 1.62266 2 280.000000 212.000000 2.543338 2 279.000000 165.000000 2.548897 2 278.000000 120.000000 2.541650 2 280.0
112654875 10 929936911244 1.65647 2 279.000000 212.000000 2.543345 2 279.000000 165.000000 2.548992 2 278.000000 120.000000 2.541997 2 280.
112654907 11 929937240525 1.68851 2 279.000000 212.000000 2.543394 2 279.000000 165.000000 2.549060 2 278.000000 120.000000 2.542092 2 280.
112654941 12 929937570513 1.7226 2 279.000000 212.000000 2.543536 2 279.000000 165.000000 2.549273 2 278.000000 120.000000 2.542314 2 280.0
112654975 13 929937911255 1.75668 2 279.000000 212.000000 2.543704 2 279.000000 165.000000 2.549487 2 278.000000 120.000000 2.542543 2 280.
112655008 14 929938240532 1.78932 2 279.000000 212.000000 2.543694 2 279.000000 165.000000 2.549543 2 278.000000 120.000000 2.542712 2 280.
112655041 15 929938570560 1.82252 2 279.000000 212.000000 2.543687 2 279.000000 165.000000 2.549686 2 278.000000 120.000000 2.543150 2 280.
```

Fig. 9: Skeleton metadata file content.

Order of the joints for each frame in the skeleton Meta data file (Fig. 9) is presented is the Table 1. This is a standard order introduced by Microsoft.

Table1: Skeleton metadata joints order and name.

J0	J1	J2	J3	J4	J5	J6	J7	J8	J9	J10	J11	J12	J13	J14	J15	J16	J17	J18	J19	J20	J21	J22	J23	J24
Spine–Base	Spine–Mid	Neck	Head	Shoulder–Left	Elbow–Left	Wrist–Left	Hand–Left	Shoulder–Right	Elbow–Right	Wrist–Right	Hand–Right	Hip–Left	Knee–Left	Ankle–Left	Foot–Left	Hip–Right	Knee–Right	Ankle–Right	Foot–Right	Spine–Shoulder	HandTip–Left	Thumb–Left	HandTip–Right	Thumb–Right

Note that the correct way to synchronise RGB, depth and skeleton frames, is using Kinect or system timestamps rather than the frame numbers.

References

[1] V. Soleimani, M. Mirmehdi, D. Damen, S. Hannuna, M. Camplani, “3D Data Acquisition and Registration Using Two Opposing Kinects”, *International Conference on 3D Vision*, Stanford, USA.