ROS + OpenNI2 + NiTE2

After weeks banging my head with OpenNI version 1.5.4 that comes with my ROS fuerte installation, I finally come to the conclusion that OpenNI 1.5.4 is highly frustrating, difficult to use, and has very low code readability. Or maybe it is just me.

It is time to migrate to OpenNI2, the latest version of the library that has been completely hauled with new architecture, with (much) better code readability. And another good thing about this release is that this will not mess the other version of OpenNI, so we can still work with both version in the same time.

The installation process is pretty straight forward. We can get the installation files after registering on OpenNI website. Then just simply run the install script from each folder.

Then, to use OpenNI2 and NiTE2 with ROS, we need to add some parameters to the CMakeLists.txt of our ROS project/package to link them with the libraries. Here's mine:

```
cmake_minimum_required(VERSION 2.4.6)
include($ENV{ROS_ROOT}/core/rosbuild/rosbuild.cmake)

# Change two lines below according to your installation
#
set(OPENNI2_DIR /home/ariandy/src/OpenNI-Linux-x64-2.2/)
set(NITE2_DIR /home/ariandy/src/NiTE-Linux-x64-2.2/)
rosbuild_init()

#set the default path for built executables to the "bin" directory
set(EXECUTABLE_OUTPUT_PATH ${PROJECT_SOURCE_DIR}/bin)
#set the default path for built libraries to the "lib" directory
set(LIBRARY_OUTPUT_PATH ${PROJECT_SOURCE_DIR}/lib)

link_directories(${OPENNI2_DIR}/Redist)
include_directories(${OPENNI2_DIR}/Include)

link_directories(${NITE2_DIR}/Redist)
include_directories(${NITE2_DIR}/Include)

rosbuild_add_executable(testing src/main.cpp)
target_link_libraries(testing OpenNI2 NiTE2)
```

Now grab any sample program codes from the OpenNI2 or NiTE2 and put it inside our ROS package for testing. It should compile just fine.

There's still one issue though. NiTE2 uses machine learning method for the human recognition and

also skeleton fitting, which relies heavily on training data. It keeps the training data on `NiTE2` folder inside `NiTE-Linux-*/Samples/Bin` folder. And somehow, when NiTE2 initializes it will look for the training data relative to the path (e.g. your executable is at `/home/user`, then it will look for `/home/user/NiTE2/*`). That is a bummer, since we can run ROS executable (node) regardless of the path and this NiTE2 thing defeats the purpose. Workaround is by navigating first to `NiTE-Linux-*/Samples/Bin/` or `NiTE-Linux-*/Redist/` then do `rosrun your_package your_node`, otherwise it won't find the training data.

Posted by ariandy on July 10, 2013

https://ariandy1.wordpress.com/2013/07/10/ros-openni2-nite2/

Getting Raspberry Pi, OpenNI2, and Asus Xtion Pro Live To Work Together

UPDATE Feb 28, 2013:
Source: **http://www.hirotakaster.com/archives/2013/01/raspberry-pi-and-openni2.php**

Notes:
– It works on my 256MB Raspberry Pi with Asus Xtion Pro Live tested through **powered USB Hub from Belkin**.
– Camera viewer that is shipped with OpenNI (NiViewer or SimpleViewer) will not work because it's built with OpenGL. Raspberry Pi doesn't support OpenGL. So to get camera visualization we have to use OpenCV. In the source above he uses OpenCV from raspbian repository. But since I'm gonna do image processing with OpenCV so I prefer to **install it manually**.
– Building OpenNI2 from source will take a lot of time. To save the fuss you can grab the pre-compiled Raspberry Pi package from Hirotaka's website above (OpenNI version 2.0.0), or my package (version 2.1.0, size ca. 1.5MB) **here**.

For OpenNI2 installation, first install the dependencies:

```
1 sudo apt-get install git g++ python libusb-1.0-0-dev freeglut3-dev do
```

Please note that `doxygen` and `graphviz` needs 600-ish MB to download (5 minutes at ca. 2 MByte/s), and they will take around 900MB of your SD Card space once installed. They are needed to compile the documentation. Once OpenNI2 is built, we do not need this two packages anymore (I think). So if you have limited internet speed, this step itself will take a lot of time, not to mention Raspberry Pi is very slow when it comes to package installation. As mentioned before, you can just download the pre-compiled package and it will work just fine.

Now grab a copy of OpenNI2 source code from github:

```
1 git clone https://github.com/OpenNI/OpenNI2
```

Then there are two files that needed to be altered:
First `OpenNI2/ThirdParty/PSCommon/BuildSystem/Platform.Arm`. Change or comment this line:

```
1 CFLAGS += -march=armv7-a -mtune=cortex-a8 -mfpu=neon -mfloat-abi=soft
```

then replace or add with this:

```
CFLAGS += -mtune=arm1176jzf-s -mfpu=vfp -mfloat-abi=hard
```

The second file is `OpenNI2/Redist/Redist.py`. Go to line 534 to find this:

```
compilation_cmd = "make -j" + calc_jobs_number() + " CFG=" + configura
```

Then duplicate the line, comment the original and change the copied line:

```
#compilation_cmd = "make -j" + calc_jobs_number() + " CFG=" + configu
compilation_cmd = "make -j1" + " CFG=" + configuration + " PLATFORM="
```

Now let's build OpenNI2:

```
cd OpenNI2/
PLATFORM=Arm make
```

This took ca. 30-40 minutes on my Raspberry Pi.

Then create the OpenNI2 package:

```
cd Redist/
./ReleaseVersion.py arm
```

Now you can find the installer package (`OpenNI-Linux-Arm-2.1.0.tar.bz2`) in the folder `OpenNI2/Redist/Final`.

To install this package, simply unzip it to somewhere. I chose in `/usr/local/src`. You might need to change your group into staff so you have write permission in that folder. I'm not sure whether this is "safe" or not.

```
sudo usermod -a -G staff pi
```

Or just use sudo while copying.

```
cd Final/
cp OpenNI-Linux-Arm-2.1.0.tar.bz2 /usr/local/src
cd /usr/local/src/
tar -xjvf OpenNI-Linux-Arm-2.1.0.tar.bz2
```

Now that we have the installation package, let's install it:

```
cd OpenNI-2.1.0-arm/
sudo ./install.sh
```

Nothing will come up if you got it right. Now you can try if it works with your Asus Xtion. First make sure it's detected in your Raspberry Pi, check the output of `lsusb -vv`, it should come up somehow like this:

```
Bus 001 Device 006: ID 1d27:0600
Device Descriptor:
  bLength                18
  bDescriptorType         1
  bcdUSB               2.00
  bDeviceClass            0 (Defined at Interface level)
  bDeviceSubClass         0
  bDeviceProtocol         0
  bMaxPacketSize0        64
  idVendor           0x1d27
  idProduct          0x0600
  bcdDevice            0.01
  iManufacturer           2 PrimeSense
  iProduct                1 PrimeSense Device
  iSerial                 0

### DELETED ###

Device Qualifier (for other device speed):
  bLength                10
  bDescriptorType         6
  bcdUSB               2.00
  bDeviceClass            0 (Defined at Interface level)
  bDeviceSubClass         0
  bDeviceProtocol         0
  bMaxPacketSize0        64
  bNumConfigurations      1
Device Status:     0x0000
  (Bus Powered)
```

If it's giving

```
Bus 001 Device 006: ID 1d27:0600
Couldn't open device, some information will be missing
...
```

unplug and plug in other USB port. My 256MB Raspberry Pi is able to detect the sensor without powered USB hub, but it couldn't get any data out of it. Some say this is because this RPi version has lower USB bandwidth. But in **Hirotaka's website** he's connecting Xtion directly to his 512MB Raspberry Pi and it works just fine.

Then try to read the sensor data:

```
cd Samples/Bin
./SimpleRead
```

This is my output:

```
ariandy@raspberrypi /usr/local/src/OpenNI-2.1.0-arm/Samples/Bin $ ./S
Warning: USB events thread - failed to set priority. This might cause
[00000000]     3816
[00033369]     3816
[00066738]     3816
[00100107]     3816
[00133477]     3816
[00166846]     3816
[00200215]     3816
[00233584]     3816
[00266954]     3816
[00300323]     3816
```

If you get the same output, you should get something nice for yourself and celebrate!

Now we just have to make an OpenCV viewer program, because the default `SimpleViewer` will not compile on Raspberry Pi.

To be continued …