Depth Imaging with the Asus Xtion Pro Live Sensor and the KIPR Link (Part 2)

Jeremy Rand
Team SNARC (Sooners / Norman Advanced Robotics Coalition)
jeremy.rand@ou.edu

Depth Imaging with the Asus Xtion Pro Live Sensor and the KIPR Link (Part 2)

9 Welcome to Part 2

Welcome back. Part 2 continues where Part 1 left off, so if you haven't read Part 1 yet, you should do so before continuing. And now, let's continue!

10 Accessing OpenNI and libkovan Simultaneously

The OpenNI examples aren't linked with libkovan, and therefore cannot access any of the Link's hardware. Obviously, this is no good for Botball purposes. We looked in `OpenNI2-master/OpenNI2-master/Samples/SimpleRead/Makefile` and found this line:

```
USED_LIBS += OpenNI2
```

This line is used to specify which libraries the program links to. So we just needed to figure out which libraries the Link's GUI automatically links programs to. After much searching through the Link's filesystem and KIPR's GitHub, we found this information buried in one of the Link's configuration files (`/etc/kovan/platform.hints`):

```
[General]
LD_FLAGS=-lkovan -lm -lpthread
C_FLAGS=-include kovan/kovan.h -include stdio.h -include target.h -include math.h
CPP_FLAGS=-include kovan/kovan.hpp -include stdio.h -include target.h -include math.h
```

The LD line lists the libraries with which KISS-IDE programs are linked. So, we modified the SimpleRead makefile:

```
USED_LIBS += OpenNI2
USED_LIBS += kovan
USED_LIBS += m
USED_LIBS += pthread
```

We then had to #include some headers in the program. We decided to use the C headers from the above list, not the CPP headers, because we were more familiar with programming the Link in C than C++. (You can access the C Link API from C++ mode without any issues that we've found.) So, the following #includes were added to SimpleRead's main.cpp:

```
#include <kovan/kovan.h>
#include <stdio.h>
#include <target.h>
#include <math.h>
```

We also added `fd(0);` at the beginning of the main program loop, and `ao();` at the end, to see if libkovan was working. We also replaced `while (!wasKeyboardHit())` (which terminates the loop when a button on the keyboard is pressed) with `while (!a_button())` since there isn't a keyboard on the Link.

We then copied over the new main.cpp via SFTP, ran "make clean" followed by "make" via SSH, and ran the bootloader from the Link GUI. The motor port started up, followed by the Xtion data being displayed. Even better, the lag was gone... maybe libkovan is patching something involving standard output, which is necessary to have good performance in the on-screen display? Pressing the A button on the Link's GUI shut off both the Xtion data and the motor.

11 `make clean`

The Link does not appear to have any persistent clock (any hackers want to prove us wrong?). As a result, it has no way of knowing whether a file is newer than another file. This breaks the makefile system, which only rebuilds files which are new. The only reliable solution we've found is to run `make clean` prior to running `make`.

We created a shell script which we use to rebuild the example code (place this in the same directory as main.cpp):

```
make clean
make
sync
```

We want to make a script which uses SCP to transfer over the source code from a PC, and then automatically runs the script above, but we haven't yet done this. (We're just using Filezilla to transfer it at the moment.)

12 Using the Depth Data

As demonstrated in the SimpleRead example program, accessing the depth camera is easy:

```
DepthPixel* pDepth = (DepthPixel*)frame.getData();
```

pDepth can then be accessed as an array. The DepthPixel type is just a typedef for a 16-bit unsigned integer (units are nominally 1mm). By default, a frame has resolution of 160×120. To increase the resolution to 640×480, we were able to use this code snippet (inserted between `depth.create()` and `depth.start()`):

```
VideoMode mMode;
mMode.setFps(30);
mMode.setResolution(640,480);
mMode.setPixelFormat(openni::PIXEL_FORMAT_DEPTH_1_MM);
rc = depth.setVideoMode(mMode);
if (rc != STATUS_OK)
{
	printf("Couldn't set resolution\n%s\n", OpenNI::getExtendedError());
	return 3;
}
```

Be warned that 640×480 will use much more CPU time if you're doing any kind of significant processing on the data. However, we are using 640×480 in KIPR Open with no trouble, although our processing is relatively simple.

To access a pixel with coordinates *(x,y)* when the horizontal resolution is *w*, use `pDepth[y*w + x]`.

Remember that you must call `getData()` as described above whenever you want to update the frame.

13 Potential Applications

The Xtion could be used for a wide range of Botball tasks, including object detection, robot localization, and obstacle avoidance. We were able to build RangeTrack (an object detection and localization library we designed for the ET sensor in 2011 [6]) with minimal effort on the Link, and adding an Xtion reader driver was trivial. In preliminary testing, RangeTrack is easily able to detect objects such as Pringles cans at a high frame rate. On the KIPR Open game board, we are successfully finding certain game board landmarks with no trouble.

We haven't used the RGB features of the Xtion at all yet, but we imagine they would be quite useful as well. Integrating them with the Link's OpenCV API would be an interesting project. The Link's vision system is designed to be extensible (e.g. to access the AR.Drone's cameras), so it could potentially be easy to add a new video source which accesses OpenNI.

The one downside of the Xtion is its minimum range: at least approximately 400mm have to be present between the Xtion and an object for the object to be visible to its depth cameras. This means that, like the ET sensor, robots have to be designed with this in mind. Placing the Xtion near the rear of the robot may be an easy fix for many situations.

14 Advantages of the Link's Architecture

The Link has taken a lot of criticism from Botballers. While some of this criticism is fair, comparatively little attention has been paid to the Link's advantages over its predecessor, the CBC. The Link's use of the Angstrom platform makes development much less tedious than the very quirky CBC. Installing libusb-1.0 was as easy as running a package operation, and the kernel is new enough that most software will "just work." Building software libraries/programs of significant size such as OpenNI can be done on-board the Link, without a cross-compiler, something that was impossible with the CBC. The presence of an SFTP daemon also makes things easier.

Yes, the Link has some bugs. But its overall design shows much more promise than the CBC. Making OpenNI run on the CBC would have been just about impossible. The same goes for the OpenCV-based vision system. We suggest that the next time you consider Link-bashing, consider that the Link v2 will probably be better than the CBCv2, due to the design changes which went into the Link v1.

15 Implications for Botball

We think that the Xtion would have a strong impact on Botball if added to the kit, mostly for the better. However, some may argue that including such a powerful sensor could damage the educational value of the game, particularly if object detection, robot localization, or obstacle avoidance are built into the firmware. (Notably, ROS [7] already supports these features.)

These concerns are similar to those expressed about PID motor control by Handy Board and Interactive C inventor Fred Martin in his paper, *Real Robots Don't Drive Straight* [8]. The issue boils down to the question of whether including something this powerful in the Botball kit, pre-built, harms students' exposure to important principles of robotics which are taking place under the hood. We think this is a significant concern. However, we don't think that this concern

merits not including the Xtion in the kit.

This concern could be addressed in a number of ways. One method is to provide bare-bones libraries, but leave enough bugs and missing features that students will be rewarded by hacking the system themselves (and thus learning about the innards of the system). This has been done in Botball many times, and many teams attempt to work beyond the libraries provided by KIPR. Another alternative method is to provide advanced libraries, but only include one Xtion in the kit, so that teams have to improvise for their second robot. This has also been done in Botball, though less often. We generally think that the former option is preferable. In any event, we believe that "real robots" don't reinvent the wheel unless they have to, and learning how sensors such as the Xtion can be used serves a practical educational benefit.

There are also other issues that can come up with using the Xtion in Botball. It may make semi-autonomous cheating easier, because it could potentially be used to recognize gestures and signals made by team members, which would be less noticeable to judges than waving a colored object in front of the robot. It could also be used to quickly recognize an opponent robot, which could be used both for good (navigating around a block) or evil (smashing into an opponent) purposes. Our opinion is that these issues can be worked around by educating judges about suspicious behavior, and few teams will ever try to cheat anyway. (If a team wants to compete in a semi-autonomous robot competition, they're probably not in Botball.) In addition, the ability to recognize opponent robots has already been possible for years via hacking the camera libraries to handle inverse color models [9]. To date, no Botball team has attempted to use this feature to do anything malicious, so empirical evidence suggests that there is not anything to worry about.

16 Conclusion

The Asus Xtion Pro Live sensor holds the ability to change the Botball scene, and making it work with the Link is relatively easy. We think teams participating in the KIPR Open or the KIPR Autonomous Aerial Vehicle Contest should consider whether the Xtion will help them -- there is a good chance that it will.

We would like to thank Ross Mead for providing general guidance on this topic. We would also like to thank Dr. Andrew Fagg for providing an Xtion to test with, and the Alcott Middle School Botball Team for letting us use their Link for testing.

We'd definitely be interested in hearing what kinds of things people do with the Xtion and the Link; if you've done something awesome/interesting, stop by the Botball Community [10] and let us know.

Happy Sensing!

References

[6] J. Rand and E. Curtis. *RangeTrack: Object Detection, Object Identification, and Robot Localization with the Sharp "ET" Rangefinder*. Proceedings of the 2011 Global Conference on Educational Robotics, 2011.
[7] ROS Contributors. ROS (Robot Operating System). http://www.ros.org/wiki/ , 2013.
[8] F. Martin. *Real Robots Don't Drive Straight*. https://www.aaai.org/Papers/Symposia/Spring/2007/SS-07-09/SS07-09-020.pdf , 2007.
[9] J. Rand. *CBC Hacking 2011: Vision Enhancements and Sensor Speedups*. Proceedings of the 2011 Global Conference on Educational Robotics, 2011.
[10] Botball Youth Advisory Council. Botball Community. http://community.botball.org , 2013.