University of Pennsylvania
School of Engineering and Applied Science

CIS 194

Adding Video Support to Christopher Done's Haskell Kinect Library

Kevin Conley
kevincon@seas.upenn.edu

April 24, 2012

Contents

1 Overview 1
- 1.1 Introduction . . . 1
- 1.2 Goals . . . 1
- 1.3 Accomplishments . . . 1

2 Usage 2
- 2.1 Dependencies . . . 2
- 2.2 Installing the libfreenect Library . . . 2
- 2.3 Installing the freenect Haskell Library . . . 2
- 2.4 Installing the freenect Examples . . . 3

3 Journal 4
- 3.1 3/31/2012 . . . 4
- 3.2 4/12/2012 (Checkpoint) . . . 5
- 3.3 4/22/2012 . . . 5
- 3.4 4/23/2012 . . . 6

4 Acknowledgements 9

1 Overview

1.1 Introduction

The Microsoft Kinect is a $150 USB gaming device for the XBOX 360 consisting of a traditional RGB camera, an infrared camera/projector, a microphone array, and an accelerometer.

The Kinect has become quite popular among hackers for its use in robotics and computer vision. An open-source driver/library for the Kinect, called libfreenect, was created in November 2010 to facilitate the development of projects that used the Kinect.

In October 2011, Christopher Done started a Haskell library to wrap the libfreenect functionality. He called it "freenect." He completed support for the depth camera but not the video camera.

1.2 Goals

The goals of this project were to:

- Add Kinect video support to his Chris's freenect library: `https://github.com/chrisdone/freenect`
- Create a demonstration program using the library that grabs an RGB image from the Kinect and saves it to a file
- Create a demonstration program using the library that displays the real-time video from the Kinect in a window

1.3 Accomplishments

All three of the above goals were accomplished. The completed code is available on my github account at `https://github.com/kevincon/freenect`

2 Usage

The following procedure assumes you are installing in a Linux environment (specifically Ubuntu 10.04 32-bit). It also assumes that you already have Haskell/GHC installed.

2.1 Dependencies

```
sudo apt-get update
sudo apt-get install cmake
sudo apt-get install git-core
sudo apt-get install libusb-1.0-0-dev
sudo apt-get install freeglut3-dev
sudo apt-get install libxmu-dev
sudo apt-get install libxi-dev
sudo apt-get install libcv-dev
sudo apt-get install libhighgui-dev
```

2.2 Installing the libfreenect Library

To install the libfreenect library, first checkout the latest version of the source code from their github repository:

```
git clone https://github.com/OpenKinect/libfreenect.git
```

Next, move into the source folder and build the library using cmake.

```
cd libfreenect
mkdir build
cd build
cmake ..
```

Install it:

```
sudo make install
```

Lastly, ensure you never have to run libfreenect code as an administrator by copying the included udev rules:

```
cd ..
sudo cp platform/linux/udev/51-kinect.rules /etc/udev/rules.d
```

2.3 Installing the freenect Haskell Library

First check out the code from my github repository:

```
git clone git@github.com:kevincon/freenect.git
```

Next, move into the folder and install using cabal:

```
cd freenect
cabal install freenect.cabal
```

2.4 Installing the freenect Examples

In the freenect source folder:

```
cd examples
cabal install examples.cabal
```

If you encounter errors, you may need to install the glut and juicypixels hackage packages:

```
cabal update
cabal install glut
cabal install juicypixels
```

You can also run these examples directly. First navigate to the src directory:

```
cd src
```

To run the RGB.hs program, which grabs an RGB video frame from the Kinect and saves it as a file called ”output.bmp” in the same directory, run:

```
runhaskell RGB.hs
```

To run the GlutRGB.hs program, which displays the real-time RGB video stream from the Kinect in a window, run:

```
runhaskell GlutRGB.hs
```

3 Journal

3.1 3/31/2012

First I followed Chris's README instructions to install the freenect drivers/libs for Ubuntu, everything installed successfully without any problems.

I tested the installation by running the glview demo in the bin directory. Video and depth frames were successfully displayed.

Next I forked Chris's github project for the Haskell Kinect interface (http://github.com/chrisdone/freenect) to my own github account (http://github.com/kevincon/freenect) and cloned it on my machine.

I saw that Chris's example Haksell programs use the Freenect Haskell module he made, but they're in different folders so I thought to install the Freenect module to some standard lib place.

I tried running "cabal install freenect.cabal" in the github repo, this looked like it's correct but libfreenect.h could not be found. I went back to the libfreenect source on my computer and ran "sudo make install." This copied all of the files to various locations, I noticed it copied the libs I copied previously manually to a different location, so maybe Chris discovered that Haskell looks for the libs in /usr/lib instead of /usr/local/lib.

After that, "cabal install freenect.cabal" still didn't work, so I added "/usr/local/include/libfreenect " as an include-dir in the file freenect.cabal myself. This got it working, installing the freenect library in /home/kevin/.cabal/lib/freenect-1.1.1/ghc-7.0.4 and registering the module with ghc.

Next I went to Chris's examples folder and tried running "runhaskell Depth.hs." I got an error about "libusb couldn't open USB device /dev/bus/usb/001/015: Permission denied." I reran the command with sudo and it looked like the example ran successfully, here's the output:

```
Devices: 1
Selected devices: [Camera]
Opened device 0.
Setted depth callback.
Write Reg 0x0105 <= 0x00
Write Reg 0x0006 <= 0x00
Write Reg 0x0012 <= 0x03
Write Reg 0x0013 <= 0x01
Write Reg 0x0014 <= 0x1e
Write Reg 0x0006 <= 0x02
Write Reg 0x0017 <= 0x00
Started depth stream.
Processing
Payload: fromList [887,887,888,887,889,892,892,892,892,892,893,893,895
Finished processing events.
```

```
Write Reg 0x0006 <= 0x00
```

Next, I found a simple C program at `http://openkinect.org/wiki/C_Sync_Wrapper` that uses the libfreenect C_sync wrapper to synchronously grab an RGB frame from the Kinect and display it using opencv. This worked for me, displaying real-time video in an opencv window.

3.2 4/12/2012 (Checkpoint)

I looked through all of Chris's code to get a good understanding of how it works using FFI to call C functions from the libfreenect library.

3.3 4/22/2012

I learned how to get rid of the "Permission denied" error when trying to use the libfreenect library. The easy way to do this is to go to the source folder for libfreenect and go to platform/linux/udev. Inside that folder is a file called 51-kinect.rules that you need to copy to /etc/udev/rules.d. So run

```
sudo cp 51-kinect.rules /etc/udev/rules.d
```

and that will get rid of the permission denied error forever (assuming a Debian/Ubuntu distro).

In an email Chris suggested I try compiling the freenect examples directly, which automatically compiles the rest of the freenect library. So I went into the examples folder and ran "cabal install examples.cabal". It gave me an error about the libfreenect lib location, so I edited the examples.cabal file the same way I previously edited the freenect.cabal file. Running cabal install again almost worked but it complained about not having glut. I thought cabal would work like apt-get and automatically get the dependencies it needs, but I just ran

```
cabal install glut
```

and then I was able to successfully compile the freenect examples. I found the Glut.hs example to run really slow for me, which I'm going to blame on this laptop's poor hardware for now. It looks much better in Chris's YouTube video: `http://www.youtube.com/watch?v=as2syH8Y8yc`

After looking over the code one more time to make sure I understood how all the pieces connected, I started adding video support. This mostly consisted of copying Chris's existing depth-related functions and replacing the word "depth" with "video". The libfreenect library does a pretty great job of abstracting the differences between depth-related functions and video-related functions. I had to add additional helper functions in the cbits/freenect-helpers.c file following Chris's lead, and I aldo had to do foreign imports for those functions in the FFI.hs file.

Once I added all the code I needed, I recompiled the freenect library and started editing the Depth.hs example to become RGB.hs. Instead of printing out a few depth pixel values, I wanted it to print out the RGB color values of the video. I compiled my RGB.hs file and tried it out. It appeared to work, here's my output:

```
Devices: 1
Selected devices: [Camera]
Opened device 0.
Setted video callback.
Write Reg 0x000c <= 0x00
Write Reg 0x000d <= 0x01
Write Reg 0x000e <= 0x1e
Write Reg 0x0005 <= 0x01
Write Reg 0x0047 <= 0x00
Started video stream.
Processing
Payload: fromList [22395,31559,16216,22144,34104,14422,21630
Finished processing events.
Write Reg 0x0005 <= 0x00
```

Next I wanted to try adapting Chris's Glut.hs example to display the RGB video in real-time. One thing I discovered I should do is change the pointer size of the video payload from Word16 to Word8. This would make it easier to access the RGB bytes, since in raw RGB mode they are stored as one Byte after another: Red first, Green second, Blue third. I also realized the values I got from the payload above consist of the RGB bytes concatenated.

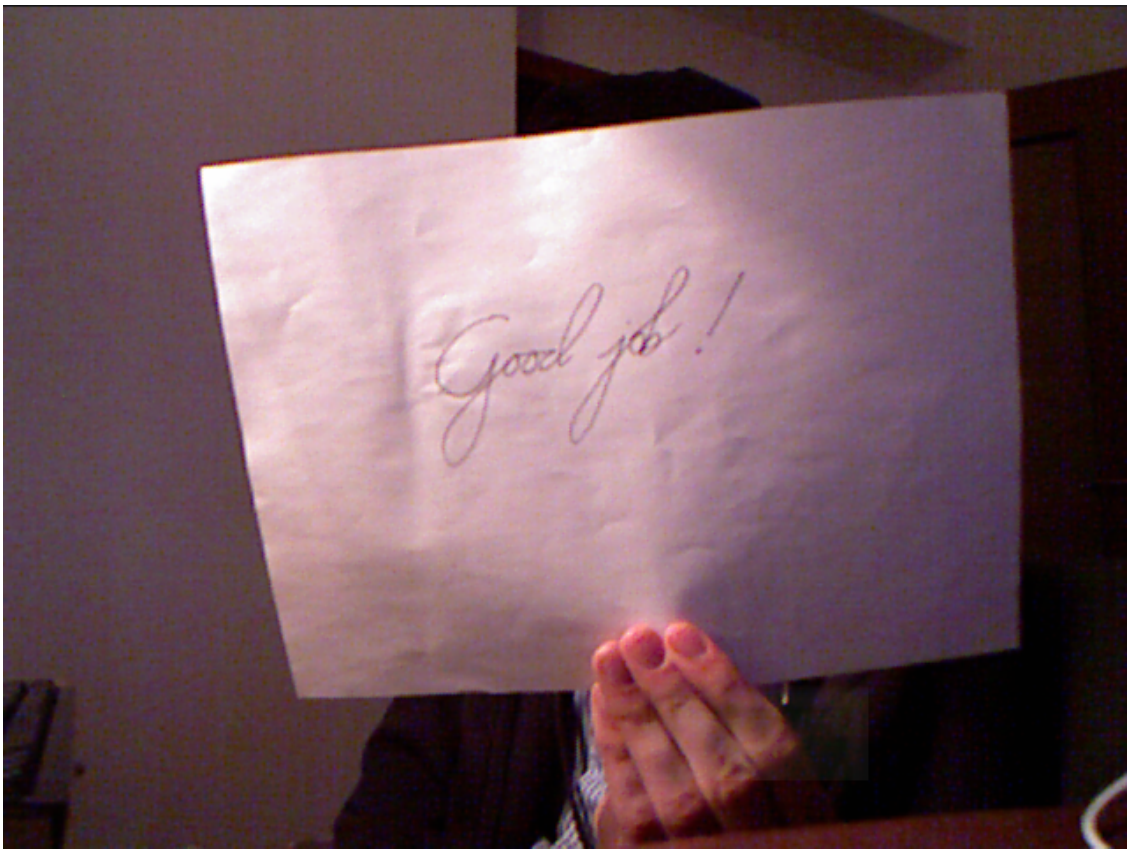
I got the GlutRGB.hs file to a point where I was ready to test it, and when I did, I got an image, but it was grayscale and something was wrong with it. It looked like multiple images overlayed on top of another. I played around with it for a while but wasn't getting anywhere, so I thought I'd try to do something simpler: saving a frame to a .BMP file. I used the JuicyPixels library for this purpose. It has a nice BMP output function.

I edited RGB.hs to a point where it should save a BMP file, and wouldn't you know it, I still got the weird grayscale effect. I decided to try changing the video mode from RGB to Bayer, and success! I got a coherent image from the Kinect. But it was still in grayscale.

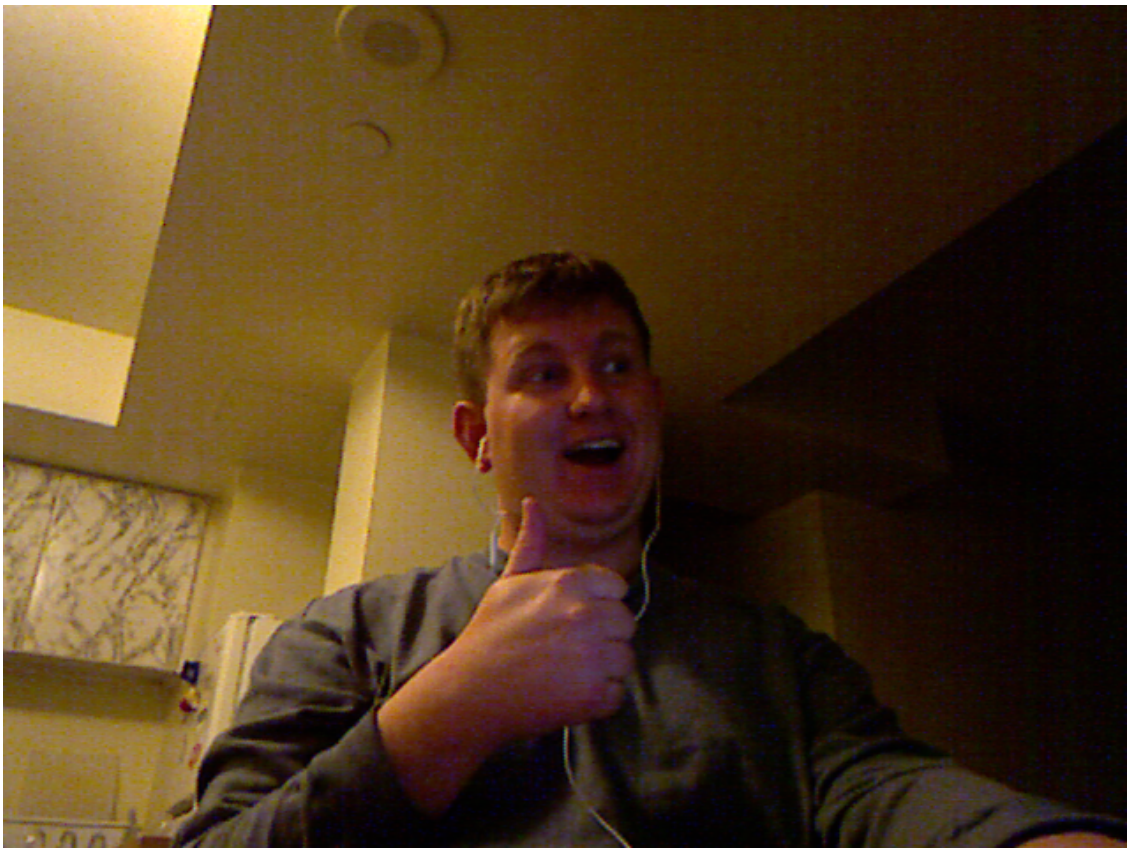
At this point I decided to give up for the night and email Chris to see if he had some ideas.

3.4 4/23/2012

Chris replied to my email and apparently there was just one mistake I was making regarding the indexing of the pixels. I was not multiplying by 3, which is necessary because of the 3 Bytes per pixel. I blame working too late. Chris sent me this image of the code working on his end:



Anyway, with that idea in mind I was able to also fix the RGB.hs file which now correctly outputs an ”output.bmp” file representing one RGB video frame grab. Here’s the result from my end:



Lastly, I edited Chris's README file to reflect some of the beginner pitfalls and startup instructions I experienced while working on this project.

With this complete, all of my goals had been reached.

4 Acknowledgements

The author would like to acknowledge Christopher Done for being a mentor for this project and for creating the freenect library in the first place. Additionally, thanks to Brent Yorgey for being such an amazing teacher, and to John Mayer for convincing me to take CIS 194 one afternoon in the Detkin Lab. I am very fortunate to have such fantastic people welcome me to the Haskell community.