Capture Synchronization for Multiple C120 Cameras with Multiple Pointers

Navid Mohaghegh
Dept. of Computer Science and Engineering, York University
navid@cse.yorku.ca

Abstract

USB communication protocol of NaturalPoint OptiTrack FLEX C120 camera is reverse engineered and corresponding Linux driver is developed. Multiple C120s are connected and synced to a real-time Linux system driving blinking wireless laser pointers. The result is a low latency real-time system that is capable of tracking multiple pointers in a large interactive surface.

1. Introduction

Large interactive surfaces enable novel forms of computing (e.g. Smartboard[1], Microsoft Surface[2], etc.). One of the research problems in this domain is how to handle user input on large surfaces[i]. Current solutions do not scale well beyond displays of approximately one square meter[ii]. This project realizes a novel form of interaction device based on wireless pens. These pens are based on laser diodes, which enables them to be used both on the display as well as off the display. This distant pointing is a feature that no other system can offer. Moreover, detecting the distance of a pen from the surface enables other forms of novel interaction. This project involves writing a real-time Linux driver for a 120 Hz Camera System. Communication protocol of the camera needs to be reverse engineered. Moreover, the project also involves frame accurate synchronization of the cameras to a computer controlled blinking laser.

2. Milestones

Project Initiation
- Research on different project scenarios (2009/09/15 - 2009/10/18)

Definition
- Defining scope of the project (2009/10/20 - 2009/10/26)
- Determine deliverable, constraints (2009/10/26 - 2009/11/02)

Planning
- Brainstorming and project plans (2009/11/02 - 2009/11/09)
- Contract proposal and detailed work breakdown (2009/11/09 - 2009/11/13)

Execution
- Obtain Dragon12 and Sequoia boards (2009/11/16 - 2009/11/23)
- Obtain OptiTrack FLEX C120 camera (2009/11/16 - 2009/11/23)
- Compile and install Linux on Sequoia board (2009/11/23 - 2009/11/30)
- Compile and install Xenomai real-time patch (2009/12/01 - 2009/12/08)
- Reverse engineer the OptiTrack FLEX C120 camera (2009/12/07 - 2009/12/14)
- Write the device driver for OptiTrack FLEX C120 camera (2010/01/04 - 2010/01/11)
- Interim Report (2010/01/11)
- Testing the device driver for OptiTrack FLEX C120 camera (2010/01/11 - 2010/01/18)
- Use Dragon12 to control multiple laser pointer (2010/01/18 - 2010/02/01)
- Use RF transceivers with Dragon12 to sync multiple laser pointers (2010/02/01 - 2010/02/15)
- Sync Dragon12 and Sequoia boards (2010/02/15 - 2010/03/15)
- Final Testings (2010/03/15 - 2010/04/02)
- Project Presentation (2010/04/05)
- Project Final Report (2010/04/30)

3. OptiTrack C120 Camera

NaturalPoint OptiTrack FLEX C120[3] Camera (shown in Figure 1) is one of the cheapest high speed and pre-processed USB video cameras in the market capable of tracking multiple bright spots. Below are the details:

- 120 FPS

1 http://www.smarttech.com
2 http://www.microsoft.com/surface
3 http://www.naturalpoint.com/optitrack/products/flex-c120

- Resolution: 355 x 288
- Imaging modes: Pre-processed, Greyscale
- Latency: 9ms
- Operating Range: 3cm - 4 meters, depending on marker size
- Lens: 46 degrees, IR 800nm bandpass coated
- Lens mount: M12 Lens Holder
- Power: 5 V @ 490 mA, including IR LED Ring
- Multi-camera synchronization
- IR LED intensity control: IR @ 850 nm, 12 LEDs 45 degree FOV, adjustable brightness, removable
- Frame Decimation control (transmit every Nth frame)
- Frame Rate control (4 - 120 FPS)
- Exposure control (electronic)
- Image size windowing
- Numeric LED readout
- Free Baseline SDK
- Dimensions: 1.78"(W) x 2.74"(H) x (0.75"(D) + 0.53"(D LED))

Figure 1 - NaturalPoint OptiTrack FLEX C120 Camera

4. Obstacles

- Reverse engineering of C120
- Building a hard real-time system
- Syncing multiple laser pointers with multiple camera
- Make the laser pointers wireless

4.1. Reverse engineering of C120

C120 has three end points:

- Vendor ID is 0x131d and Production ID is 0x126 (Rev=00.00)
- EndPoint1 which has the address of 0x02 (USB bulk out)
- EndPoint2 which has the address of 0x84 (USB bulk in)
- EndPoint3 which has the address of 0x86 (USB bulk in)

C120 starts sending raw greyscale data as soon as it receives the first initialization packet. This makes the protocol analysis extremely hard as most of the USB packet sniffers try to capture the entire traffic on all the end points. And obviously with the rate of 120 FPS, the allocated memory buffers become full very soon. As a result, some of the packets may not be captured since packet sniffer simply discards the extra data which is just arrived to buffer while emptying the buffer to the disk.

After examining the protocol many times, it was observed that EP0x86 is purely used to send out data packets from the camera. We have modified the packet sniffer[4] to omit all the traffic on EP0x86 during the initialization phase so that at least we can capture the useful traffic on the other end points reliably.

4.2. Building a hard real-time system

Building a hard real-time system
AMCC PowerPC 440EPx[5] single board computer running Linux and Xenomia[6] hard-real-time framework is used to host the driver for C120 camera. Details for setting up Linux and Xenomia on this board can be studied on Appendix A.

Freescale Motorola HCS12[7] micro controller has been utilized to deal with any low level timers. Codes that have been developed for this platform can be studied on Appendix B.

4.3. Syncing multiple laser pointers with multiple camera

There is no guarantee of deterministic latency for USB bulk transfers as bus will only be utilized after all the other transactions have been allocated and also re-transmission can occur at any time due to CRC mismatch. This makes the synchronization of the cameras and blinking laser pointers almost impossible.

Luckily, each camera has a sync signals that allows multiple cameras to be synced together at the camera level. Also after studding the C120 USB protocol, it has been observed that each camera frame has a counter value (increased linearly from 0 to 0xff which loops).

Camera performs the following once it receives the sync signal from the master camera:

4http://sourceforge.net/projects/usbsnoop
5http://www.appliedmicro.com
6http://www.xenomai.org
7http://gcc-hcs12.com

- Opens the shutter
- Exposure delay
- Closes the shutter
- Processes the taken picture to find the bright spots
- Serializes the coordinates of the bright spots
- Assembles the data in a packet along with the frame counter
- Sends the packet
- Increases the frame counter (if the value is 0xff, frame counter will be 0x00 again)
- Sleeps and listen for sync signal activation event

This is very helpful as a hard-real time system can be developed to monitor the cameras' sync signal and blink the laser pointers accordingly. Moreover, frame counter of the cameras can be used to further sync a specific frame to a specific laser pointer.

The following is summary of the events that happens on the developed hard-real-time system which is tapping on the sync signal of the master camera (on sync signal activated):

- Increase the frame counter (this is the clone of the camera counter)
- if (counter%10 == 1) Blinks only laser 1
- if (counter%10 == 2) Blinks only laser 2
- …

The following is summary of the events that happens on the developed driver on the Linux side (on USB data packet arrival):

- Extract the packet data and frame counter
- Decode the coordinate
- if (counter%10 == 1) Print laser1 coordinate
- if (counter%10 == 2) Print laser2 coordinate
- ...

Please refer to Appendix B and Appendix C for detailed source codes.

4.4. Make the laser pointers wireless

Low latency wireless transceivers needed (less than 9 ms of latency). It was decided to use off the shelf RF transceivers with no error correction capabilities for proof of concept. Link 315 MHz, 4800 bps transceiver[8] was utilized. However more enhanced transceivers (i.e. XBee Pro 115200 bps with error correction capabilities[9]) need to be studied.

8http://www.sparkfun.com/ommerce/product_info.php?products_id=8947
9http://www.sparkfun.com/commerce/product_info.php?products_id=8690

5. Decoding a sample raw output of the camera

Below is a sample raw output of the camera:

- ...
- 36 00 f7 6b b7 b3 b6 03 36 00 52 6b 00 00 00 0c
- 36 00 f1 6d b7 b3 b6 03 36 00 54 6d 00 00 00 0c
- 36 00 f3 6f b6 b4 b5 03 b7 b3 b6 03 36 00 56 6f 00 00 00 10
- ….

Each packet consist of the followings:

- As you can see above, each packet starts with “36 00 ??”
- After the start token, we will have the frame counter. In above we have: 6b, 6d, 6f as our frame counters
- After the frame counter we have quadrupole of a coordinates data. For instance for the packet with frame counter 6f, we have two bright spots with the following encoded coordinate:
    - b6 b4 b5 03
    - b7 b3 b6 03
- After the encoded coordinate, we will have 36 00 which indicates the end of packet.

C120 reports coordinate of X and Y for each bright spots. Below is an example of decoding a quadrupole:

- Imagine a bright spot with “b6 b4 b5 03” data
- y = 0xb6; //Y coordinate
- x_1 = 0xb4; //X coordinate - start
- x_2 = 0xb5; // X coordinate - end
- high_bits = 0x03;
- x_1 += (((high_bits & 0x01) >> 0) << 8) + (((high_bits & 0x08) >> 3) << 9);
- x_2 += (((high_bits & 0x02) >> 1) << 8) + (((high_bits & 0x10) >> 4) << 9);
- y += (((high_bits & 0x04) >> 2) << 8) + (((high_bits & 0x20) >> 5) << 9);

Please note the length of the bright spot can be calculated using $|x_2-x_1|$

6. Change the camera settings

Please send the followings commands to end point located at the address 0x02 to change different settings of the camera (?? stand for the hex value of the setting):

- Threshold value of C120 can vary from 1 to 253. Send the following command to change the value:
  - 15 ?? 01 00
  - for instance: 15 fd 01 00 changes the threshold to 253
  - please note 0xfd = 253
- Exposure value of C120 can vary from 0 to 399. Send the following commands to change the value:
  - 23 40 1c ?? 00 00 //low bits
  - 23 40 1d ?? 00 00 //high bits
  - 23 40 00 84 00 00
  - for instance to changes the exposure to 399 the following commands should be sent:
    - 23 40 1c 8f 00 00 //low bits
    - 23 40 1d 01 00 00 //high bits
    - 23 40 00 84 00 00
    - please note 0x018f = 399
- C120 has a hardware feature that can limit the length of the bright spots.
  - For instance, C120 can be instructed to only report the bright spots that has a minimum length of 399 by sending (0 to 1024 are valid values):
    - 19 0e 0f 01 8f
    - please note 0x018f = 399
  - Or as another example, C120 can be instructed to only report the bright spots that has a maximum length of 399 by sending (0 to 1024 are valid values):
    - 19 0f 0f 01 8f
    - please note 0x018f = 399
- To put the camera in greyscale mode the following commands should be sent:
  - 14 01
  - 19 03 0f 00 01
  - 12
- To put the camera out of greyscale mode the following commands should be sent:
  - 14 01
  - 19 03 0f 00 00
  - 12
- C120 can be instructed to change the frame rate to less than 120 FPS. Frame rate of the camera can be changed using the following patterns:
  - 3 to 6 frames: 23 40 00 c8 00 00
  - 7 to 10 frames: 23 40 00 c0 00 00
  - 11 to 13 frames: 23 40 00 b8 00 00
  - 14 to 16 frames: 23 40 00 b0 00 00
  - 17 to 20 frames: 23 40 00 a8 00 00
  - 20 to 33 frames: 23 40 00 a0 00 00
  - 34 to 40 frames: 23 40 00 98 00 00
  - 41 to 50 frames: 23 40 00 90 00 00
  - 51 to 66 frames: 23 40 00 88 00 00
  - 67 to 100 frames: 23 40 00 80 00 00
  - Please note if you like C120 only send 3 frames per second, you need to use 3 to 6 frames setting. You will receive 6 packets, and you have to drop 3 of them at software side. And as you can say this is much better than discarding 117 packets at software side.
- C120 decimation feature can be changed as well. This means camera can be instructed to actually send the data packets every Nth frame. For instance to have the data every 399 frames, once can send the following:
  - 19 0c 0f 01 8f
  - please note 0x018f = 399
- Each C120 can have a unique ID. The ID can be set using:
  - 19 07 0f 00 ??
  - for instance to have an ID of 1: 19 07 0f 00 01
- C120 can be instructed to only report the bright spots that fall into a specific interval of X and Y
  - To put a constrain on X values:
    - 19 08 0f ?? ?? //X_{start}
    - 19 09 0f ?? ?? //X_{end}
    - For instance to report only the bright spots that their X value folds in 0-399 the following should be sent:
      - 19 08 0f 00 00 //$X_{start} = 0$
      - 19 09 0f 01 8f //$X_{end} = 399$
  - To put a constrain on Y values:
    - 19 0a 0f ?? ?? //Y_{start}
    - 19 0b 0f ?? ?? //Y_{end}
    - For instance to report only the bright spots that their X value folds in 0-399 the following should be sent:
      - 19 0a 0f 00 00 //$Y_{start} = 0$
      - 19 0b 0f 01 8f //$Y_{end} = 399$

7. Start and stop C120

Please note you should stop the camera before sending any new settings. Below commands will start and stop the camera (need to be sent to the endpoint located at 0x02):

- Stop the camera:
  - 14 01
  - 12
  - 10 00 20
  - 10 00 80
  - 10 00 10

    - 10 00 40
    - 13
- Start the camera:
    - 14 01
    - 12
    - 10 00 80
    - 10 00 20
    - 13
    - Perform a read from end point located at 0x84
    - Perform a read from end point located at 0x84
    - 17
    - Perform a read from end point located at 0x84
    - 1D
    - Perform a read from end point located at 0x84
    - 19 14 0f 00 00
    - 14 01
    - 12
    - 15 87 01 00
    - 14 01
    - 19 03 0f 00 00
    - 12
    - 14 00
    - 19 07 0f 00 36
    - 14 00
    - 12
    - 10 20 20
    - 10 80 80

8. Tricks to have longer “on” time for lasers

We have mentioned that HCS12 micro controller is used to monitor the sync active signal of the master camera to drive the lasers according to the frame counter. The whole idea is to have only one laser active at a given time that we can identify its location.

If we can keep lasers “on” more often, we will be able to have less obvious blinking which will visually improve our feeling from the blinking system.

To achieve this goal, we will decrease the exposure time of the camera. This enables us to turn on all the lasers as soon as the exposure period is done. Also we may reduce the number of lasers that system can track (i.e. 6 lasers concurrently).

9. Future work

The current system is fully functional and fulfilled all the requirements. However it will be nice to add a GUI for calibration of multiple cameras and change the settings.

Wireless subsystem can be improved much more. For instance, if reliable transceivers with embedded error correction is used, one can disable/enable lasers on the fly to even further enhances the blinking effect. Currently 6 laser pointers are utilized and which are blinking 10 times in a second.

i W. Stuerzlinger, L. Zaman, A. Pavlovych, J.-Y. Oh, The Design and Realization of CoViD, A System for Collaborative Virtual 3D Design, Virtual Reality, ISSN 1359-4338, 10(2), 135-147, Oct 2006

ii Pavlovych, A., Stuerzlinger W. Laser Pointers as Interaction Devices for Collaborative Pervasive Computing, Advances in Pervasive Computing, Eds. Ferscha, Hoertner, Kotsis, OCG, ISBN 385403176-9, 315—320 , Feb 2004