Lab 5. Final Report

WebCam

EE405 Electronic Design Lab

20154920 Wootae Song

17 May 2016

Table of Contents

Purpose

In this Lab, we capture images with a webcam that is connected to Beaglebone, transfer the captured images to PC using UDP, and display the images using SDL2.

Procedures & Results

A1. Test WebCam on PC Ubuntu

This step is to install software and drivers needed and initialize our laptop and Beaglebone for the webcam use. The commands lsusb, dmesg | tail, lsmod | grep uvc resulted in the exact same output as the lab text. After installing vlc, mplayer, and streamer, I was able to play videos, stream videos, and take a picture with Logitech C110 webcam.

A2. Capture Image with WebCam on PC

In this step, we take pictures using Logitech C110 webcam using a C program.

For uname –a, lsusb, ls /dev/vi*, and all the commands that follows, I got the same results.

```
root@beaglebone:/home/debian/nfs_client/5_WebCam/D/B3# uname -a
Linux beaglebone 3.8.13-bone79 #1 SMP Tue Oct 13 20:44:55 UTC 2015 armv7l GNU/Li
nux
root@beaglebone:/home/debian/nfs_client/5_WebCam/D/B3# lsusb
Bus 001 Device 003: ID 046d:0829 Logitech, Inc.
Bus 001 Device 001: ID 1d6b:0002 Linux Foundation 2.0 root hub
Bus 002 Device 001: ID 1d6b:0002 Linux Foundation 2.0 root hub
root@beaglebone:/home/debian/nfs_client/5_WebCam/D/B3# ls /dev/vi*
/dev/video0
```

After installing, v4l-utils and libv4l-dev, I was able to use v4l2-ctl commands and change the camera's properties.

24. Using the capture.c program

After modifying capture.c, I was able to take pictures with Logitech C110 webcam using a C program: capture2.c.

I first took YUYV format images and got the following results:

```
alan@alan-ThinkPad-X250:~/RoboCam/5_WebCam/d_ImageCapture/down$ ./Capture2_PC -Y
 -c 1 -o > capture2_pc_1.raw
Force Format 1
.
alan@alan-ThinkPad-X250:~/RoboCam/5_WebCam/d_ImageCapture/down$ ./Capture2_PC -Y
 -c 1 -o > capture2_pc_2.raw
Force Format 1
.
alan@alan-ThinkPad-X250:~/RoboCam/5_WebCam/d_ImageCapture/down$ ls -la capture2_
pc*.raw
-rw-rw-r-- 1 alan alan 614400  5월 17 14:37 capture2_pc_1.raw
-rw-rw-r-- 1 alan alan 614400  5월 17 14:37 capture2_pc_2.raw
```

YUYV format images are fixed in size. In YUYV format, each pixel contains 2 bytes. (1 byte for Y: luminance and 1 byte for U or V: Chroma) As a result, the size of an image that is 640 by 480 is 640 * 480 * 2 = 614400 bytes, which is pretty big. We can only read the raw file with hd command, which is shown below.

```
00000290  43 66 b3 79 47 a3 69 83  a5 9a 97 45 af 76 d5 d3  |Cf.yG.i....E.v..|
000002a0  51 66 2f 0c 5a e9 76 0d  f8 3a f8 0b 71 4f 31 ee  |Qf/.Z.v..:..qO1.|
000002b0  20 ec 85 b0 41 16 b4 79  d2 66 4a 3b 50 87 51 04  | ...A..y.fJ;P.Q.|
000002c0  6c 01 04 80 08 1c e0 f8  76 61 58 5f f9 d8 81 87  |l.......vaX_....|
000002d0  c6 16 49 88 26 d5 ed 00  ed 41 da c2 94 66 62 72  |..I.&....A...fbr|
000002e0  42 9b 26 5c c4 1c 08 bb  04 86 a4 5e db e9 5c 48  |B.&\.......^..\H|
000002f0  8f 3a 65 1a a2 4e ad 46  8c 8b e2 6c d2 52 f6 2c  |.:e..N.F...l.R.,|
00000300  9f 00 54 b9 30 47 d1 da  8e e9 26 0a 0d 89 22 4b  |..T.0G....&..."K|
00000310  30 2f 0f 18 60 90 06 41  0a 10 e0 59 4a 12 01 62  |0/..`..A...YJ..b|
00000320  00 eb 80 00 31 0e d5 8f  88 04 ae 22 7a 13 25 f2  |....1......"z.%.|
00000330  78 fb 8b 1b b8 16 dd 2e  c0 ca ba 00 b4 10 46 c9  |x.............F.|
00000340  dc 73 23 08 f8 26 b1 d1  02 d6 56 00 5b 24 9f cf  |.s#..&....V.[$..|
00000350  51 23 03 61 cb a6 9d 51  ad d8 53 9c 1a ad 29 bb  |Q#.a...Q..S...).|
00000360  6f a7 87 79 08 c5 87 bc  40 4d 69 10 15 9a 0e 6f  |o..y....@Mi....o|
00000370  f1 82 63 dc 83 04 75 0b  11 df 6c 90 f3 82 70 e3  |..c...u...l...p.|
--More--
```

We can take an image and save it as jpeg file, which compress the size of image according to the complexity of the image. However, the Logitech C110 webcam's image capturing format instantly. Therefore, to capture images with jpeg format, you need to run these commands before capturing an images:

./Capture2_PC –M –c 1 –o > capture2_pc_1.jpeg

./Capture2_PC –M –c 100 –o > capture2_pc.mjpg

After running the first command, I got the following result:

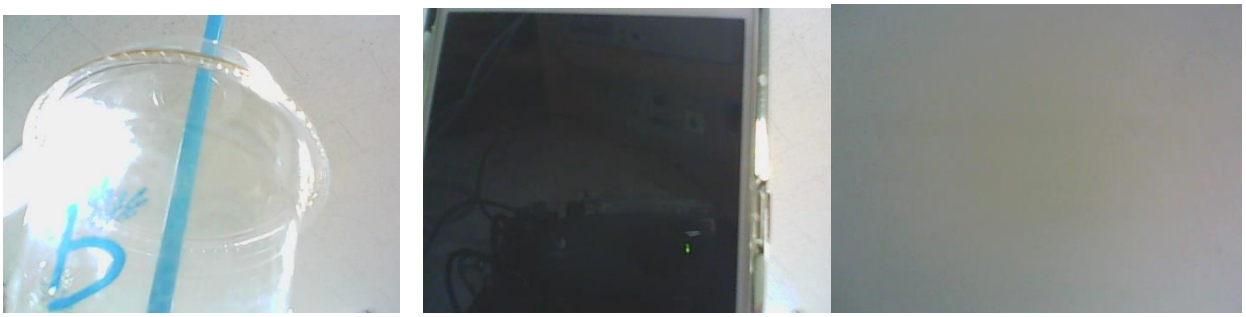
```
alan@alan-ThinkPad-X250:~/RoboCam/5_WebCam/d_ImageCapture/down$ ./Capture2_PC -M
 -c 1 -o > capture2_pc_1.jpeg
Force Format 2
.
```

```
-rw-rw-r-- 1 alan alan 614400  5월 17 14:42 capture2_pc_1.jpeg
-rw-rw-r-- 1 alan alan 614400  5월 17 14:41 capture2_pc_1.raw
-rw-rw-r-- 1 alan alan 614400  5월 17 14:41 capture2_pc_2.raw
```

As you can see in the above picture, capture2_pc_1.jpeg file has the size of 614400 bytes, which means that it is still in YUYV format.

The second command captures a video. After the second command, you will be able to capture in jpeg format.

```
alan@alan-ThinkPad-X250:~/RoboCam/5_WebCam/d_ImageCapture/down$ ./Capture2_PC -M
 -c 1 -o > capture2_pc_2.jpeg
Force Format 2
.
alan@alan-ThinkPad-X250:~/RoboCam/5_WebCam/d_ImageCapture/down$ ./Capture2_PC -M
 -c 1 -o > capture2_pc_3.jpeg
Force Format 2
.
alan@alan-ThinkPad-X250:~/RoboCam/5_WebCam/d_ImageCapture/down$ ./Capture2_PC -M
 -c 1 -o > capture2_pc_4.jpeg
Force Format 2
.
alan@alan-ThinkPad-X250:~/RoboCam/5_WebCam/d_ImageCapture/down$ ls -la capture2_
pc*.jpeg
-rw-rw-r-- 1 alan alan 614400  5월 17 14:42 capture2_pc_1.jpeg
-rw-rw-r-- 1 alan alan  43336  5월 17 14:45 capture2_pc_2.jpeg
-rw-rw-r-- 1 alan alan  34776  5월 17 14:45 capture2_pc_3.jpeg
-rw-rw-r-- 1 alan alan  31120  5월 17 14:46 capture2_pc_4.jpeg
```



Left: capture2_pc_2.jpeg, an image of a coffee cup, size: 43336 bytes

Middle: capture2_pc_3.jpeg, an image of a screen of a smartphone, size: 34776 bytes

Right: capture2_pc_4.jpeg, an image of a paper, size: 31120 bytes

We captured 3 images in jpeg format. We could confirm that the size of a jpeg image depends on the complexity of the image. The left image had the biggest size because it has the most variety of colors, curves, and lines; the right image had the least size because it is the simplest.

B. Capture WebCam Image on Beaglebone

The same exact procedure and output as A.

C. Learn SDL via Tutorial

Nothing to explain

D. Test Video View with SDL on PC

41. Test View_Images_PC (Source code attached)

Explanations:

This code is the modification of the sixth SDL tutorial. I added gray surface that is displayed at first before reading any images. However, the program reads the images very fast so you can barely see the gray box. Also, this code contains two loops: thread loop and main loop. The thread loop deals with reading images and displaying images using SDL and main loop deals with event polling like clicking x of a window to close the window. Each image is displayed for 1 second, which is implemented by using the usleep function. After calling the function SDL_CreateThread, a thread is created and it runs the function threadFunciton. ThreadFunction calls loadMedia Function in an infinite loop. The loadMedia function has a counter, reads an image according to the counter, and displays the image. The main function quits the program when a user clicks x of the window.

first picture: the gray screen displayed at first

the following three pictures are repeatedly displayed for 1 second.

42. Images_to_UDP and UDP_to_View

Images_to_UDP (Source code attached)

This program reads in 3 images, puts each images in to buffers, and send the images using UDP datagram.

This program requires command line argument which is the IP address of receiving host. With the IP address of receiving host, we gather network information using getaddrinfo. After, we create a socket so we can send message to the receiver. Then, we open an image file according to counter. After opening the

file, we send the size of the image first so that receiving host can prepare a buffer and specify its size to receive the image packet. After sending the size of an image, we create a buffer, store the image file into the buffer, and transmit the buffer to the receiver. This is repeated 3 times for the 3 pictures.

UDP_to_View (Source code attached)

This program is the combination of listener.c and SDL tutorials. We first initialize SDL, draw gray box, and create thread. The thread function is an infinite loop that creates UDP data socket and receives UDP packets. When this threaded loop receives an image packet, it opens a file called rcv.jpg with wb option, indicating that it will write to the file, and writes the image packet that is received to the opened file. After, the threaded loop runs loadMedia function, which opens the file (rcv.jpg) and displays the image. The main loop deals with event polling like clicking x of a window to close the window. To sum up, this program runs forever until you click x of the window, receives image packet, saves the image, displays the image to the SDL window.

Images_to_UDP sending three images to the host with IP address 192.168.100.3:

```
alan@alan-ThinkPad-X250:~/RoboCam/5_WebCam/D/B2/talker$ ./Image_to_UDP_PC 127.0.
0.1
sending picture size
img_size: 39914
sending picture size
img_size: 47758
sending picture size
img_size: 23832
```

The host with IP address 192.168.100.3 receiving the image

```
alan@alan-ThinkPad-X250:~/RoboCam/5_WebCam/D/B2$ make
g++ UDP_to_View_PC.cpp -w -lSDL2 -lSDL2_image -o UDP_to_View_PC
alan@alan-ThinkPad-X250:~/RoboCam/5_WebCam/D/B2$ ./UDP_to_View_PC
listener: waiting to recvfrom...
size: 39914
listener: waiting to recvfrom...
size: 47758
listener: waiting to recvfrom...
size: 23832
listener: waiting to recvfrom...
```

43, 44 Capture_to_UDP.c

Capture_to_UDP_PC and Capture_to_UDP_Bone resulted in the same exact result. Therefore, I will explain only Capture_to_UDP_Bone, which is the purpose of this lab.

Capture_to_UDP.c (Source code attached)

This program captures an image using Logitech C110 webcam, send the image using UDP, and quits.

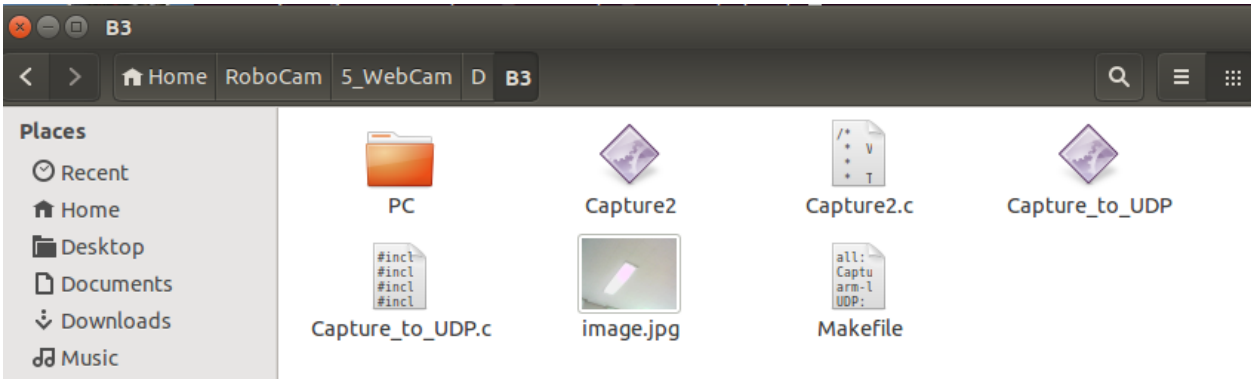
This program first calls Capture2.c program by using system() function. The exact line is: ("./Capture2 –M –c 1 –o > images.jpg"). This captures a jpeg file and save it as images.jpg. After, it checks for

command argument options. This program needs two command argument options –a and –p options. The –a option specify the IP address of the receiver and the –p option specify the port number of the receiver. After, it has the same network set up of UDP used in talker.c. Just like Images_to_UDP, this program stores the image into a buffer, sends the buffer to the receiver, and quits.

Capture_to_UDP sending an image that it captures to a receiving host that has IP address of 192.168.100.3 and port number of 4960.

```
debian@beaglebone:~/nfs_client/5_WebCam/D/B3$ ./Capture_to_UDP -a 192.168.100.3
-p 4960
Force Format 2
.
sending picture size
img_size: 6288
```

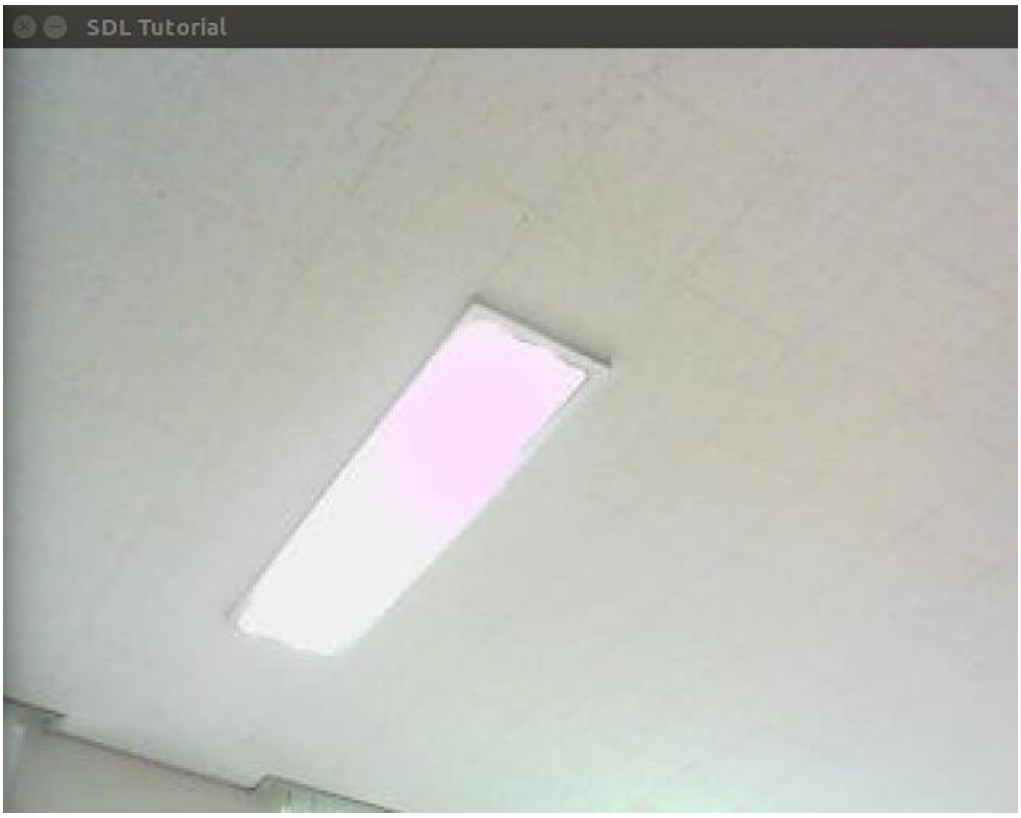
After running the command, image.jpeg is formed:



UDP_to_View receives the image packet

```
alan@alan-ThinkPad-X250:~/RoboCam/5_WebCam/D/B2$ ./UDP_to_View_PC
listener: waiting to recvfrom...
size: 6288
listener: waiting to recvfrom...
```

and displays the image


SDL Tutorial

Discussions

The Logitech C110 webcam does not change its image storing format instantly. As a result, after changing the image storing format, I had to capture few images to see the change in the image storing format.

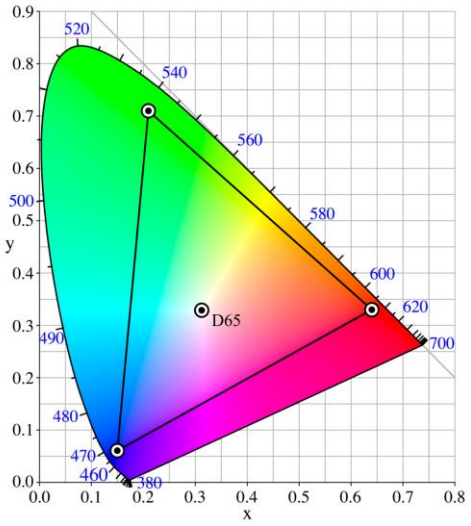
Also, when we captured images in 640 by 480 resolution, the size of image files was too large for datagram's maximum size, which is 2^16. As a result, we changed the resolution of the image captured to 320 by 240. The professor said in class that we need to implement the transmission of image in 640 by 480 resolution in the final lab 6. I am going to implement it by using fragmentation of a large UDP datagram.

Questions

1) Compare formats: YUYV and Jpeg

YUYV is YUV's Chroma subsampled (4:2:2) version. Let me explain what YUV is first.

The most well-known color space is RGB color space. The color gamut for RGB color space is shown below:

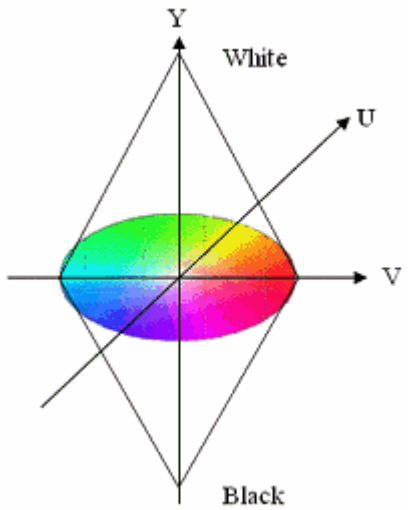


The colors inside and in-border of horseshoe are the colors that we can see. The colors inside of the triangle are the colors that can be represented on monitor. This color gamut is achieved by colorimeter experiment, which involved in human subjects to match RGB colors to purely saturated colors.

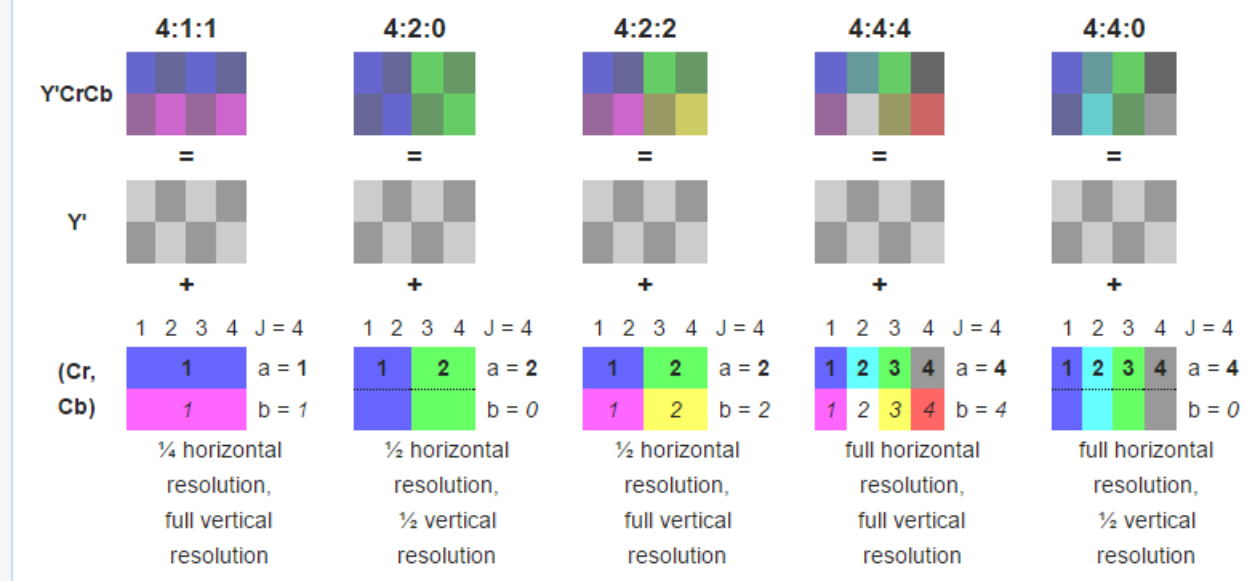
YUV is another color space that is more suitable for humans. Our eyes' retinas contain two types of photoreceptors: rods and cones, which help us to detect colors. Rods are responsible for black and white; cones are responsible for color perception. Humans are more sensitive to black and white than colors because our retina contains more rods than cones: 120 million cones to 6 million cones. Therefore, it is better to separate luminous information from color information.

YUV color space exactly does the work. In YUV, Y is responsible for luminance (brightness; black and white), U and V are responsible color information. As a result, YUV separates color information from luminance information, which is more suitable to humans than RGB color space.YUV is also used in video. In Europe, people use the PAL or SECAM codings for TV, which are based on YUV matrix transformation. Since YUV color shape separates color information from black and white, it is compatible with old black and white TV.

YUVYUV color space is shown below:

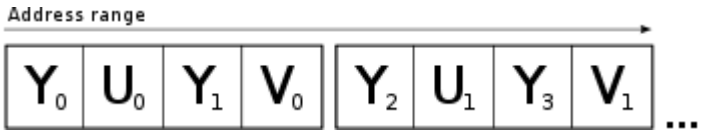


Now that I explained YUV color space, let me explain what Chroma subsampling is. The main reason that we use Chroma subsampling is to decrease the size of a video or an image. Chroma subsampling uses the fact that humans are less sensitive to colors than black and white, as I explained above. As a result, in Chroma subsampling, we do not use all of the color information on a pixel. The notation for Chroma subsampling is J:a:b. J is the horizontal sampling reference, a is the number of chrominance samples in the first row of J pixels, and b is the number of changes of chrominance samples in the second row of J pixels. You can see the examples below.



(YCbCr is the digital version of YUV)

YUYV is basically 4:2:2 Chroma sampled YUV like the picture below:



JPEG, joint photographic experts group, file is a file for digital image that implements lossy method. What I mean by lossy method is that, when you save a picture as a JPEG file, the saved JPEG file isn't 100% same as the original image and there is some loss of image. This is to save the size of image. (tradeoff between size and image quality) As a result, the size of JPEG file depends on the complexity of image. I experienced this in the lab. When I took different images the size of an image with least complexity (white paper) was smaller than the size of an image with complexity (Beaglebone Robot).

On the other hand, the YUYV files are fixed size because each pixel contains information (2 bytes). As a result, in Lab, the size of the images that we captured with YUYV format were the same and fixed. (640 * 480 * 2 = 614400) The size of the YUYV file depends on the resolution of an image.

2) Search image storage format in commercial digital camera

There are three types of image storing format in commercial digital camera: RAW, TIFF, and JPEG. Not every camera can capture an image with RAW and TIFF storing format, but most cameras can capture images with JPEG format. RAW and TIFF formatted files are not compressed in any manner. In other words, RAW and TIFF files include all of the information that your camera's lens capture. Because RAW and TIFF files are uncompressed, they are huge

in size. On the other hand, JPEG files are compressed in lossy manner. What I mean by lossy manner is that when you compress an image using lossy manner, you use some information of the original image. However, it is not likely that you will notice the loss with bare eyes unless you zoom in. Because JPEG images are compressed, they are relatively small in size compared to RAW and TIFF files. They are compressed in a manner that the more complex the image is bigger the size. As a result, you can save a lot of JPEG file images in a single memory card.

In our lab, we use Logitech C110 webcam. This webcam has capability of capturing an image as RAW and JPEG format. When I took a picture with RAW format, the size was fixed to 614400 bytes. (640 * 480 * 2) The size depends on the resolution. However, when I took pictures with JPEG format, the size of the images was smaller than RAW file. Also, the sizes depended on the complexity of images. This phenomenon is due to the compression of the JPEG images. Since there is no compression of RAW files, the size is fixed. We can change the level of compression of JPEG files. With high compression rate, we sacrifice image quality for small size of the file, and vice versa.

The image below compares RAW file image and JPEG file image. It is a little difficult to notice the difference. However, when you zoom in you may notice the difference.

Reference

DesignLab_RoboCam_Lecture5_E.pdf

DesignLab_RoboCam_Lab5_E.pdf

https://en.wikipedia.org/wiki/YUV

https://en.wikipedia.org/wiki/JPEG

https://en.wikipedia.org/wiki/Chroma_subsampling