Term Project #1: YUV format 이해

2017312041 정종호

목차

본격적인 과제에 앞서 YUV에 구체적으로 이해 해보도록 하자. YUV format은 기존 영상의 RGB format에서 더욱 압축시킬 목적으로 생겨난 형태이다. YUV의 개념 설명자체는 교수님의 pdf에 잘 설명이 되어있으므로 개념적인 부분은 생략하고 조금 더 구체적인 예시를 들어가며 이해를 해보도록 하겠다.

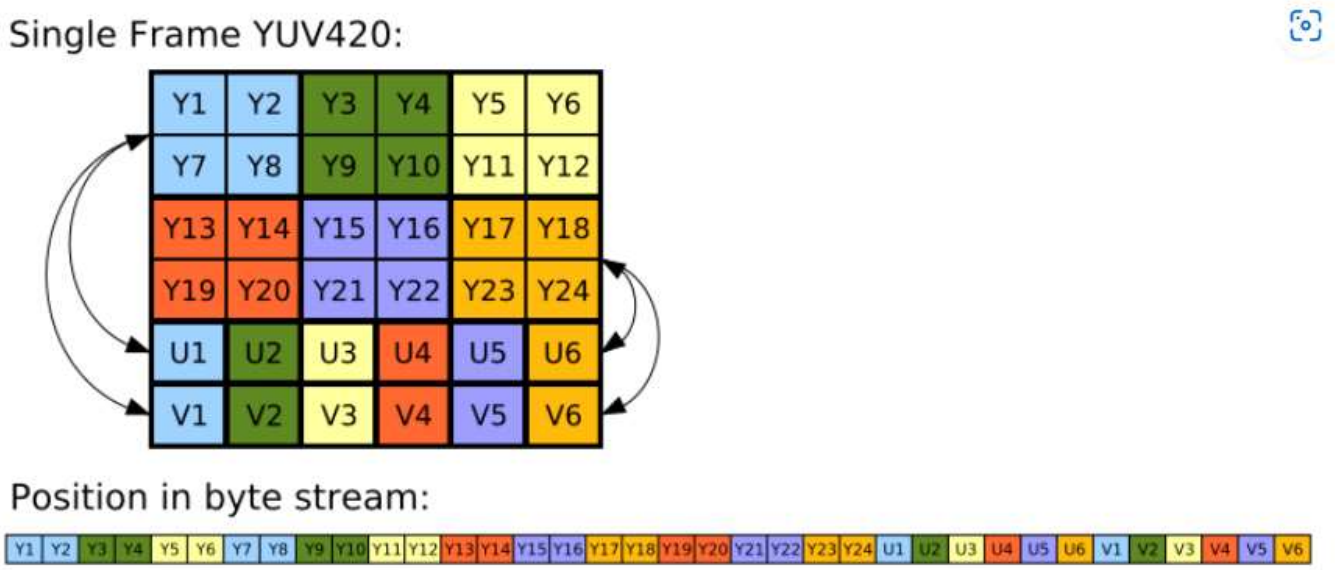
우리가 흔히 아는 RGB의 경우는 [R,G,B]형태로 이루어져있다. 흔히 0~255사이의 숫자를 부여하는데 파이썬의 리스트의 경우 16*16의 이미지라면 [0~255,0~255,0~255] 리스트가 하나의 원소가 되어 16*16이미지에 들어가게 되는 것이다. 총 원소의 개수는 w*h*3이 될 것이다. 그런데 컴퓨터의 경우는 0과1밖에 사용하지 못하기 때문에 0~255를 표현하기 위해서는 2^8=256 즉 8bit가 필요하다. 즉 하나의 빛을 표현하기 위해 필요한 메모리는 24(8*3)bit인 것이다. 아래 예시를 보면 좀 더 직관적으로 이해될 것 이다.

RGB	RGB	RGB	RGB	RGB	RGB	RGB	RGB
RGB	RGB	RGB	RGB	RGB	RGB	RGB	RGB
RGB	RGB	RGB	RGB	RGB	RGB	RGB	RGB
RGB	RGB	RGB	RGB	RGB	RGB	RGB	RGB
RGB	RGB	RGB	RGB	RGB	RGB	RGB	RGB
RGB	RGB	RGB	RGB	RGB	RGB	RGB	RGB
RGB	RGB	RGB	RGB	RGB	RGB	RGB	RGB
RGB	RGB	RGB	RGB	RGB	RGB	RGB	RGB

그렇다면 이렇게 생긴 format을 어떻게 압축한다는 것일까? YUV를 조금 더 천천히 생각해보자.

YUV란 사람의 눈이 밝기(Y)에 민감하고 색상(U,V)에 둔감하다는 것을 이용하였다. 픽셀마다 사람 눈에 민감한 밝기(Y)데이터를 넣어주고 비교적 둔감한 색상(U,V)데이터는 2*2픽셀마다 넣어주는 것이다. 직관적인 이해를 위해 아래 그림을 보자. (실제로 이렇지는 않지만 이해를 위한 그림이다.)
2*2픽셀당 YUV가 하나씩 들어가 있다 여기서 UV가 모두 동일한 색상을 나타낸다는 것이다.
위에서 말했듯이 인간은 색상에 둔감하므로 해상도를 낮춰도 인지하는 데 문제가 없기에 이런 방식으로 압축을 하는 것이다. 여기서 RGB의 경우는 4개의 픽셀을 사용할 시 총 12개의 문자가 들어갔다. 하지만 YUV의 경우 4개의 픽셀 사용 시 YYYYUV 총 6개만 사용하므로 RGB에 비해서 압축할 수 있다는 것을 알 수 있다.

YUV	Y	YUV	Y	YUV	Y	YUV	Y
Y	Y	Y	Y	Y	Y	Y	Y
YUV	Y	YUV	Y	YUV	Y	YUV	Y
Y	Y	Y	Y	Y	Y	Y	Y
YUV	Y	YUV	Y	YUV	Y	YUV	Y
Y	Y	Y	Y	Y	Y	Y	Y
YUV	Y	YUV	Y	YUV	Y	YUV	Y
Y	Y	Y	Y	Y	Y	Y	Y



그림참고: [YUV 영상 처리 실습 - 날도의 잡다한 기술 블로그 | NaLDo's IT Blog (naldo627.github.io)]

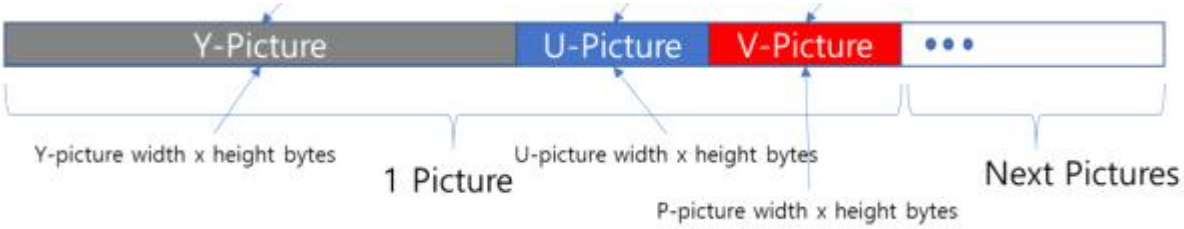
그렇다면 여기에서 YUV 뒤에붙은 420은 무엇인지 궁금하지 않을 수 없다. 이는 YUV의 비율이다.
예를 들어 YUV444, YUV422, YUV420, YUV411을 보자. 이는 Y에 대한 비율이다. 444의 경우는 Y가 4바이트가 필요할 때 U와V도 4바이트씩 필요한 format이다. 422,411의 경우도 마찬가지로 Y가 4바이트가 필요할 때 U와V가 각각 2, 1이 필요한 format이다. (411은 422보다 색상 표현기능이 떨어질 것이다.)

2. 2번문제 해결과정

해당 파일의 크기는 41,472,000byte이다. 이 파일이 1280*720 크기의 이미지 30장이 YUV420 format으로 있다는 것을 생각해보면

1280*720*1byte(Y)*30, 1280*720*0.25byte(U)*30, 1280*720*0.25byte(V)*30로 이루어졌다는 것을 알 수 있다.
여기서 이미지는 연속적으로 이루어져 있을 테니 파일을 한 장의 이미지 크기(1280*720*(1+0.25+0.25))씩 자른 뒤 홀 수 번째 데이터들만 가져오겠다. (YUV format은 헤더가 없어서 작업이 편리했다.)

3. 3-A번문제 해결과정



의 형태이기에 2가지 방식을 사용해보려 한다.
a. UV 파트에 모두 0을 넣는 방식
b. UV 파트를 스킵하고 Y파트를 연속적으로 쓰는 것

a의 결과는 기존의 형태 중에 Y파트만 출력되는 것을 볼 수 있다. 하지만 영상의 용량은 바뀌지 않는다.
b는 용량이 줄어들 것이다. YUV는 raw데이터 이므로 크기는 정확하게 $\frac{2}{3}$만큼 줄 것이다.

3-B번 문제 해결과정

본 문제를 해결하기 위해 아래의 픽셀에서 색을 칠한 부분만 남겨놓으려 한다.

1	2	3	4
5	6	7	8
9	10	11	12
13	14	15	16

그런데 본인의 프로그래밍 실력이 부족하기에 아래 영상을 압축하려면 두 가지 과정을 거쳐야한다. 세로를 압축하고 가로를 압축하는 것이다. 예를 들어 보면

a.

1	2	3	4
5	6	7	8
9	10	11	12
13	14	15	16

->

1	3
5	7
9	11
13	15

->

1	3
9	11

b.

1	2	3	4
5	6	7	8
9	10	11	12
13	14	15	16

->

1	2	3	4
9	10	11	12

->

1	3
9	11

의 두 가지 케이스가 있을 것이다. a의 케이스의 경우에는 홀수 번째 픽셀 정보만 골라서 온 뒤에 width/2번의 메모리씩 가져오는 것

b의 케이스 경우에는 width번씩 메모리를 가져온 뒤에 남은 것 중 홀수 번째 픽셀 정보만 골라서 오는 것이다. 여기서 홀수 번째 memory에 접근하는 방식을 반복적으로 써야 하기에 꽤 많은 시간이 걸린다. 그렇기에 a의 케이스보다 b의 케이스로 압축하는 것이 더 빠른 시간이 걸릴 것이라 예상된다. (호출되는 함수의 횟수가 많이 차이 남) 이를 확인해보도록 하자

4.

1번 문제 해결

YUV player를 틀게되면 Size와 Color라는 툴바가 등장하는데 Size에서 720p, 1080p 같은 해상도 사이즈를 선택 할 수 있으며 Color에서 YUV444, YUV422, YUV420 같은 영상 format에 대하여 선택할수 있다. YUV는 raw데이터로 결국 어떻게 해석하냐에 따라 결과물이 나오므로 Size와 Color에서 알맞는 format을 설정하여 플레이하는것이 중요하다.

2. 번문제 (홀수 picture)

```c
#include <stdio.h>
#include <stdlib.h>
#include <string.h>
int main() {

   FILE *pFile = NULL;
   FILE *pWriteFile = NULL;


   pFile=
fopen("./PeopleOnStreet_1280x720_30_Original
.yuv", "rb");
   pWriteFile =
fopen("./result/YUV_Y_0.yuv", "wb");

   if(pFile == NULL) {
     printf("file is not exist");
     return 0;
   }


   int frame_size = 1280 * 720
*(1+0.25+0.25) ;
   unsigned char* frame = (unsigned
char*)malloc(frame_size);


   for (int i = 0; i <30; i++) {
     if(i % 2 == 0) {
       memset(frame, 0, frame_size);
       fread(frame, sizeof(unsigned char),
frame_size, pFile);
       fwrite(frame, sizeof(unsigned char),
frame_size, pWriteFile);
     } else {
       fread(frame, sizeof(unsigned char),
frame_size, pFile);
       memset(frame, 0, frame_size);
     }
   }

   fclose(pFile);
   fclose(pWriteFile);
   printf("finish");
   return 0;
}
```

<- 기존 파일 열기

<- 작성할 파일 열기

<- 기존 파일 존재유무 확인

<- 프레임사이즈 = 해상도 *(Y(1)+U(0.25)+V(0.25))

<- 홀수 번째만 선택 (0부터 시작이므로)

3-A번 문제 해결

YUV 플레이어에는 Size와 Color을 고정시켰기에 2가지 방식으로 보겠다. (1280*720, YUV420)

a. UV 파트에 모두 0,128을 넣는 방식
b. UV 파트를 스킵하고 Y파트를 연속적으로 쓰는 것

a.

```
   for (int i = 0; i <30; i++) {


     memset(frame, 0, frame_size);
      fread(frame,  sizeof(unsigned  char), frame_size, pFile);
      fwrite(frame,  sizeof(unsigned  char), frame_size, pWriteFile);

     memset(frame, 0, frame_size);
      fwrite(frame,  sizeof(unsigned  char), frame_size * 1/2, pWriteFile);
      fread(frame,  sizeof(unsigned  char), frame_size * 1/2, pFile);

   }
```

b.

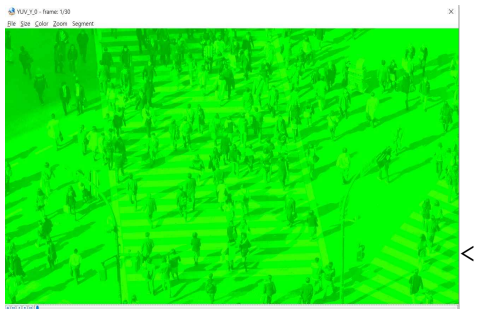
```
   for (int i = 0; i <30; i++) {


     memset(frame, 0, frame_size);
      fread(frame,  sizeof(unsigned  char), frame_size, pFile);
      fwrite(frame,  sizeof(unsigned  char), frame_size, pWriteFile);


     memset(frame, 0, frame_size);
      fread(frame,  sizeof(unsigned  char), frame_size/2, pFile);

   }
```

a. Size: (1280*720) Color: YUV420



<- 0대입

<- 128대입

b. Size:(1280*720), color: Y

YUV_Y.yuv	2022-10-12 오전 9:31	YUV 파일	27,000KB
YUV_Y_0.yuv	2022-10-13 오후 6:25	YUV 파일	40,500KB
YUV_Y_128.yuv	2022-10-13 오후 6:26	YUV 파일	40,500KB

$$\left(\frac{1280\times720\times\left(\frac{3}{2}\right)\times30\ -\ 1280\times720\times\left(\frac{1}{2}\right)\times30}{1024}\right)=27000$$

3-B번 문제해결

(1) 너비 -> 높이 압축(a)

```
for (int i = 0; i <30; i++) {

     memset(frame, 0, frame_size/2);
     for (int j = 0; j<frame_size/2; j++)
     {
        fread(frame,  sizeof(unsigned  char),
1, pFile);
       fwrite(frame,  sizeof(unsigned  char),
1, pWriteFile);
       fseek(pFile, 1, SEEK_CUR);
     }
       fseek(pFile,    frame_size    *    1/2,
SEEK_CUR);
   }
```

```
   for (int i = 0; i <30; i++) {

     for  (int  j  =  0;  j<frame_size2/1280;
j++)
     {
       memset(frame2, 0, 640);
       fread(frame2,  sizeof(unsigned  char),
640, pFile2);
       fwrite(frame2, sizeof(unsigned char),
640, pWriteFile2);
       fseek(pFile2, 640, SEEK_CUR);
     }
   }
```

(2) 높이 -> 너비 압축(b)

```
for (int i = 0; i <30; i++) {

     memset(frame, 0, frame_size/2);
     for  (int  j  =  0;  j<frame_size/(1280*2);
j++)
     {
       memset(frame, 0, 1280);
        fread(frame,  sizeof(unsigned  char),
1280, pFile);
       fwrite(frame,  sizeof(unsigned  char),
1280, pWriteFile);
       fseek(pFile, 1280, SEEK_CUR);
     }
       fseek(pFile,frame_size    *    1/2    ,
SEEK_CUR);
   }
```

```
for (int i = 0; i <30; i++) {

     memset(frame2, 0, frame_size2/2);
     for (int j = 0; j<frame_size2/2; j++)
     {
        fread(frame2,  sizeof(unsigned  char),
1, pFile2);
       fwrite(frame2, sizeof(unsigned char),
1, pWriteFile2);
       fseek(pFile2, 1, SEEK_CUR);
     }
   }
```

사이즈: (640 * 360), color: Y

```
root@DESKTOP-0N4R8ND:/mnt/c/Use
finish
a 방법 소요시간: 9.000000
root@DESKTOP-0N4R8ND:/mnt/c/Use
finish
b 방법 소요시간: 4.000000
```

YUV_Y.yuv	2022-10-12 오전 9:31	YUV 파일	27,000KB
YUV_Y_0.yuv	2022-10-13 오후 6:25	YUV 파일	40,500KB
YUV_Y_128.yuv	2022-10-13 오후 6:26	YUV 파일	40,500KB
YUV_Y_sampling.yuv	2022-10-13 오후 9:00	YUV 파일	6,750KB
YUV_Y_sampling_a1.yuv	2022-10-19 오후 1:39	YUV 파일	13,500KB
YUV_Y_sampling_a2.yuv	2022-10-19 오후 1:40	YUV 파일	6,750KB
YUV_Y_sampling_b1.yuv	2022-10-19 오후 1:40	YUV 파일	13,500KB
YUV_Y_sampling_b2.yuv	2022-10-19 오후 1:40	YUV 파일	6,750KB

참고

1. 그림참고: https://naldo627.github.io/2019/04/14/yuv-handling/
2. https://velog.io/@ruthetum/YUV