Video I/O

2018.10.30

Seoungjun Oh(sjoh@kw.ac.kr)
Wooju Lee (krosea@kw.ac.kr)

Multimedia LAB

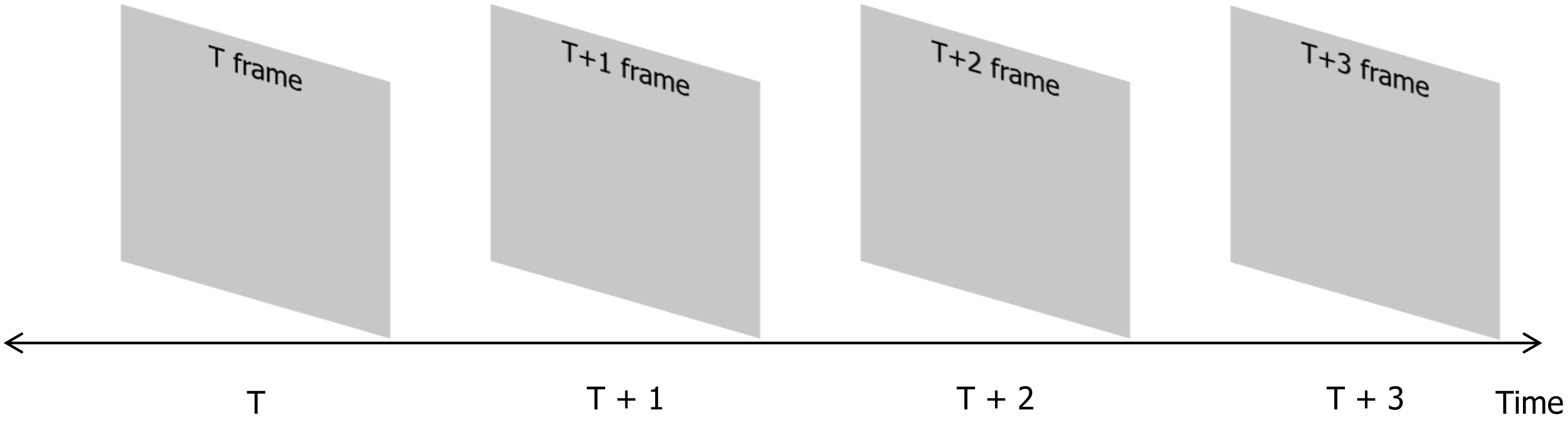
VIA-Multimedia Center, Kwangwoon University

Contents

- Video format
- Example
- Assignment

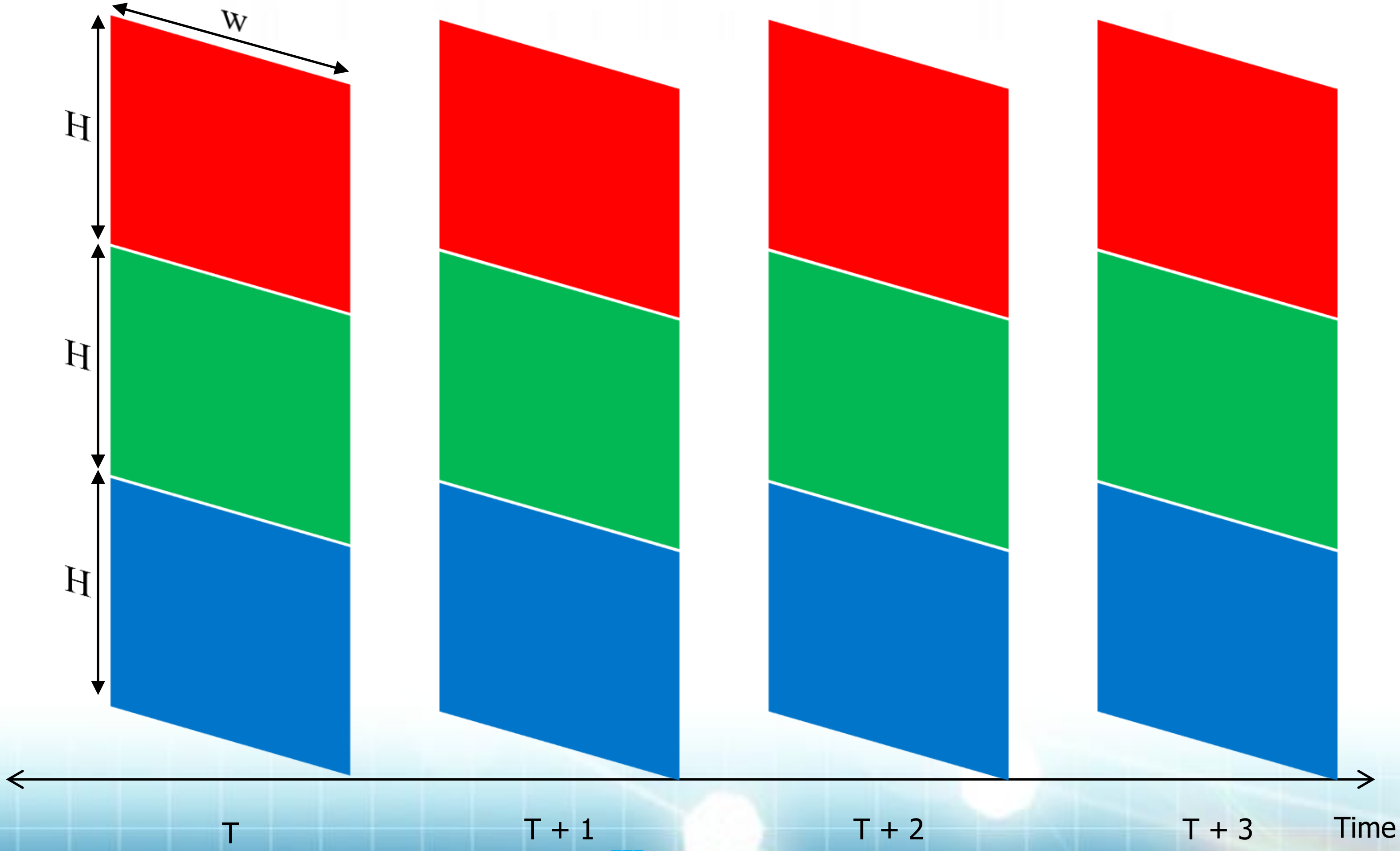
Video format

- Basic concept of the Video file

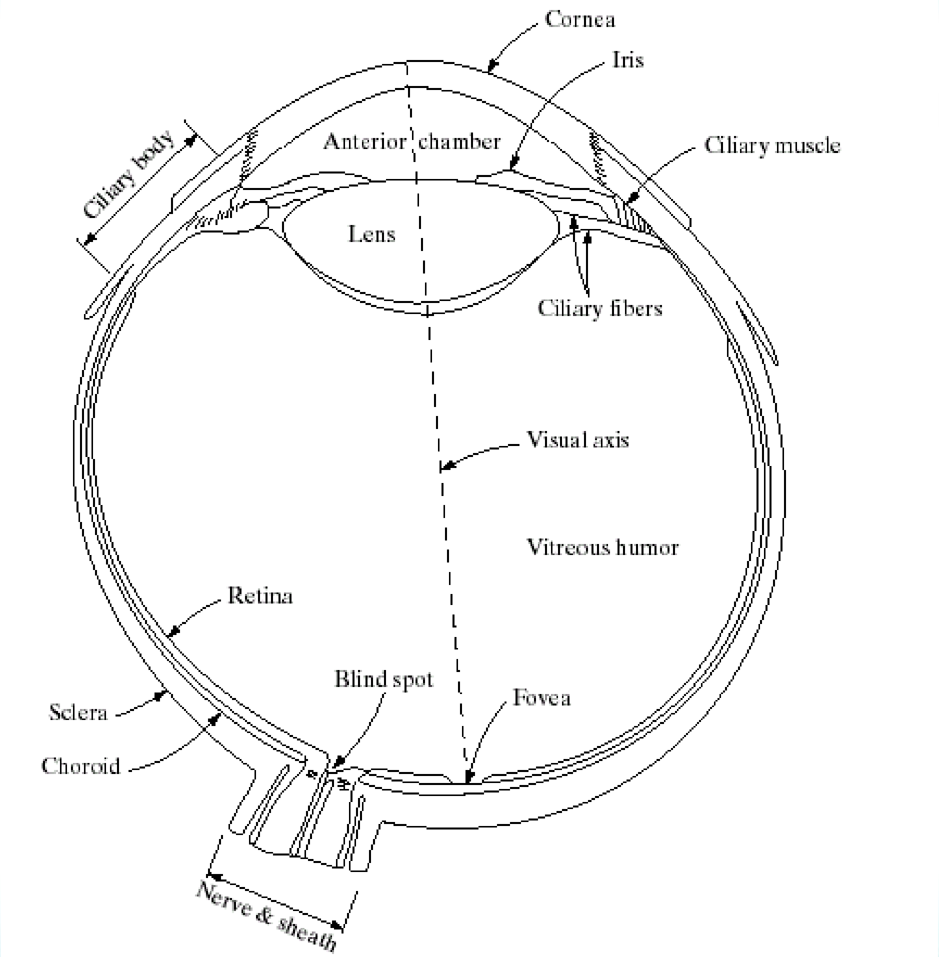


Video format

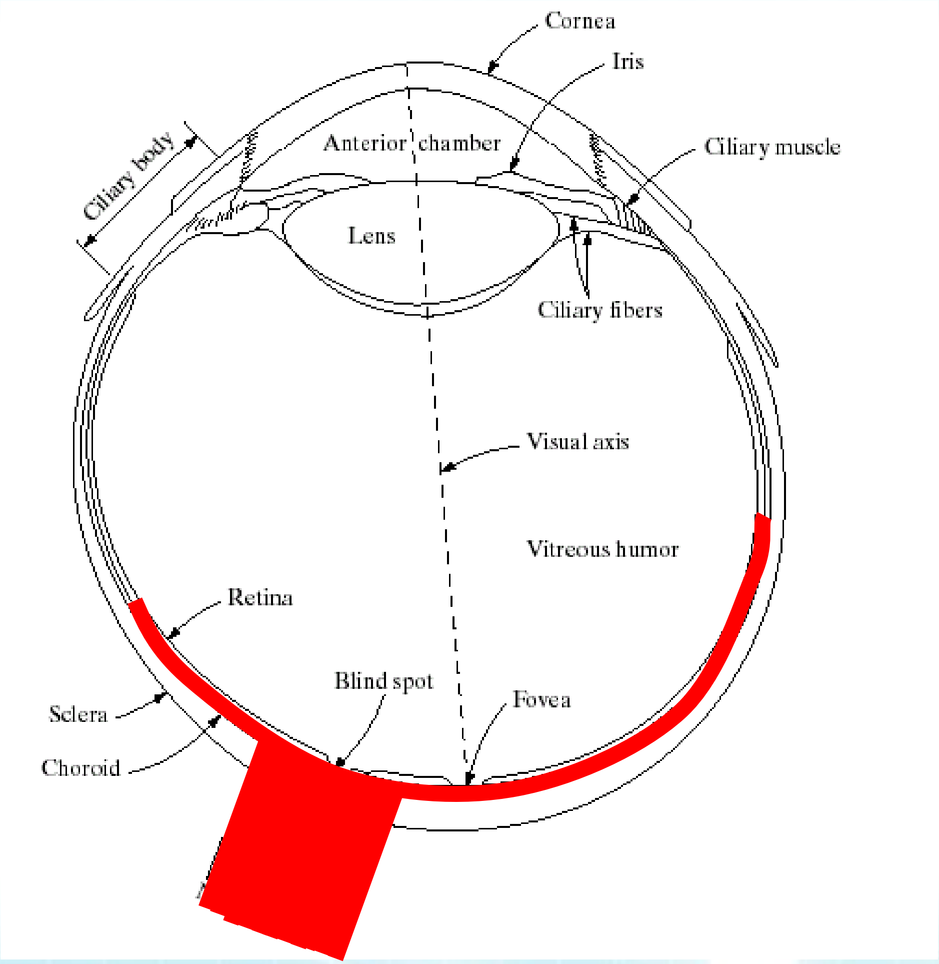
- Basic concept of the Video file



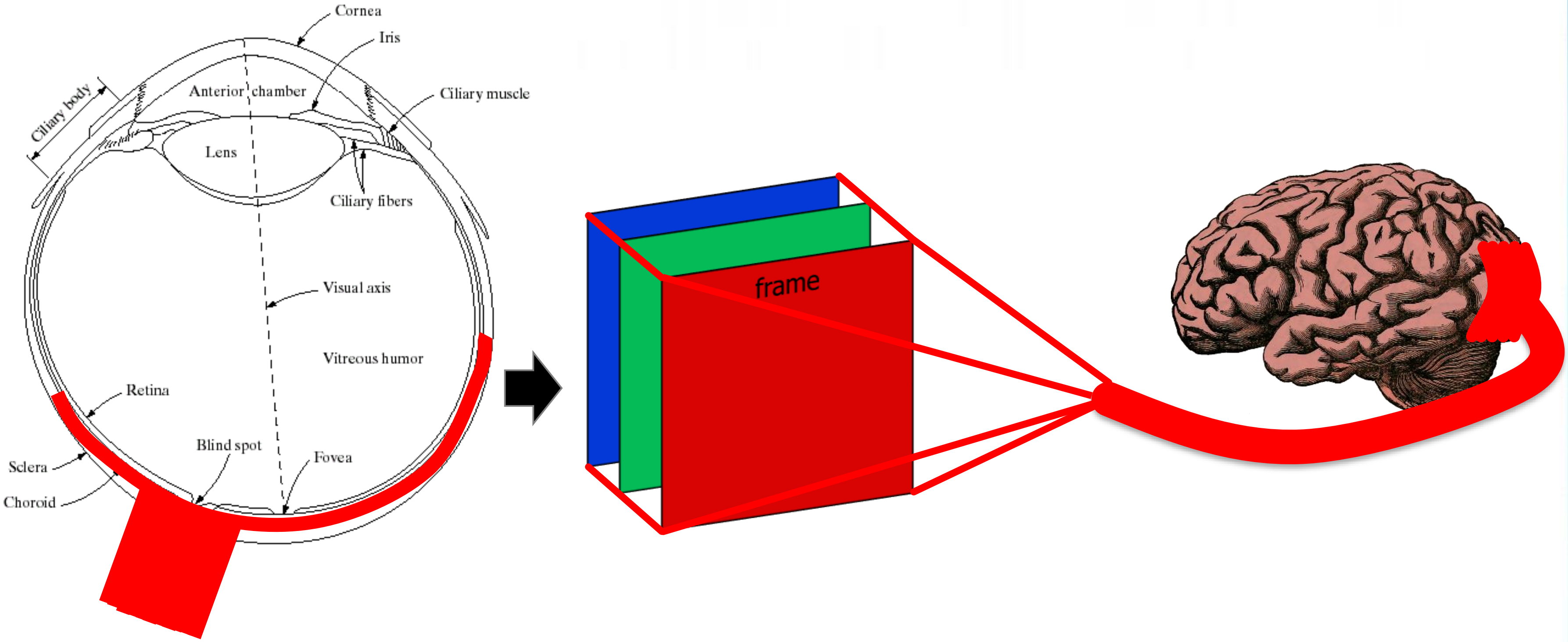
Elements of Visual Perception



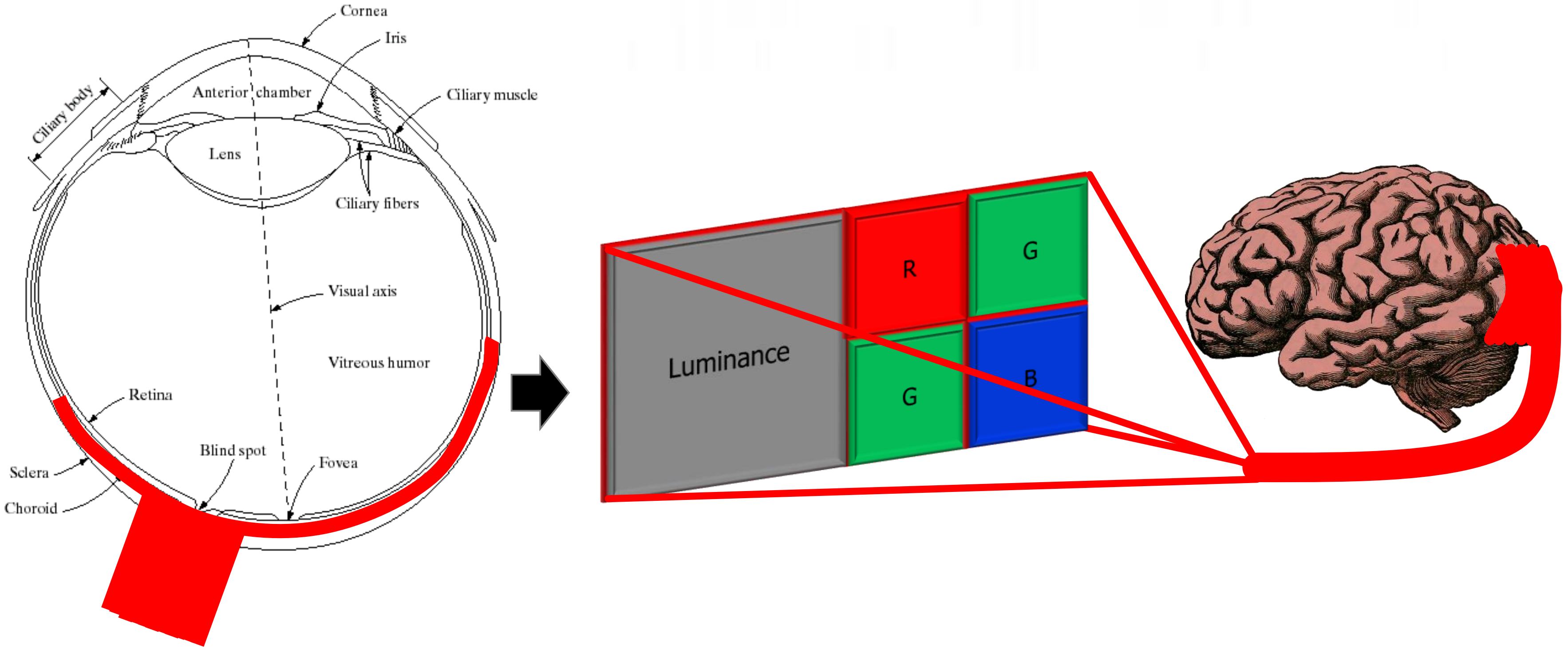
Elements of Visual Perception



Elements of Visual Perception



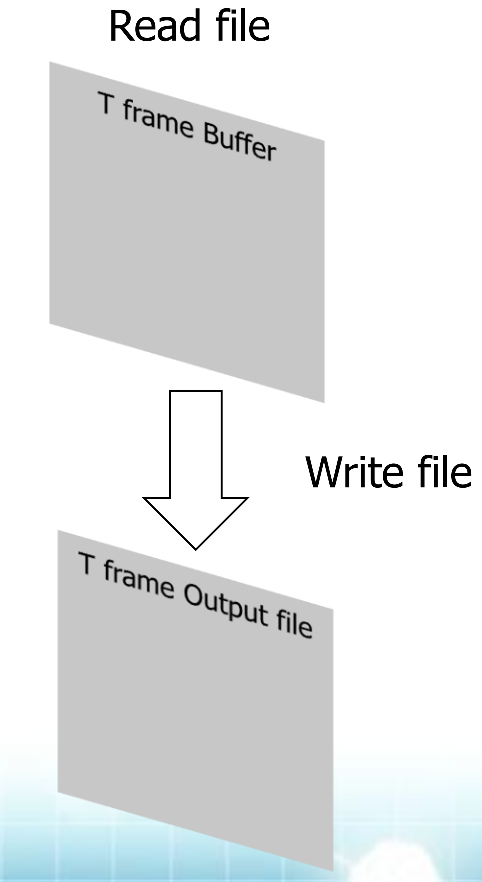
Elements of Visual Perception



Example

- Video copy

```
Loop(!End of file)
        {
```



Read file

T frame Buffer

Write file

T frame Output file

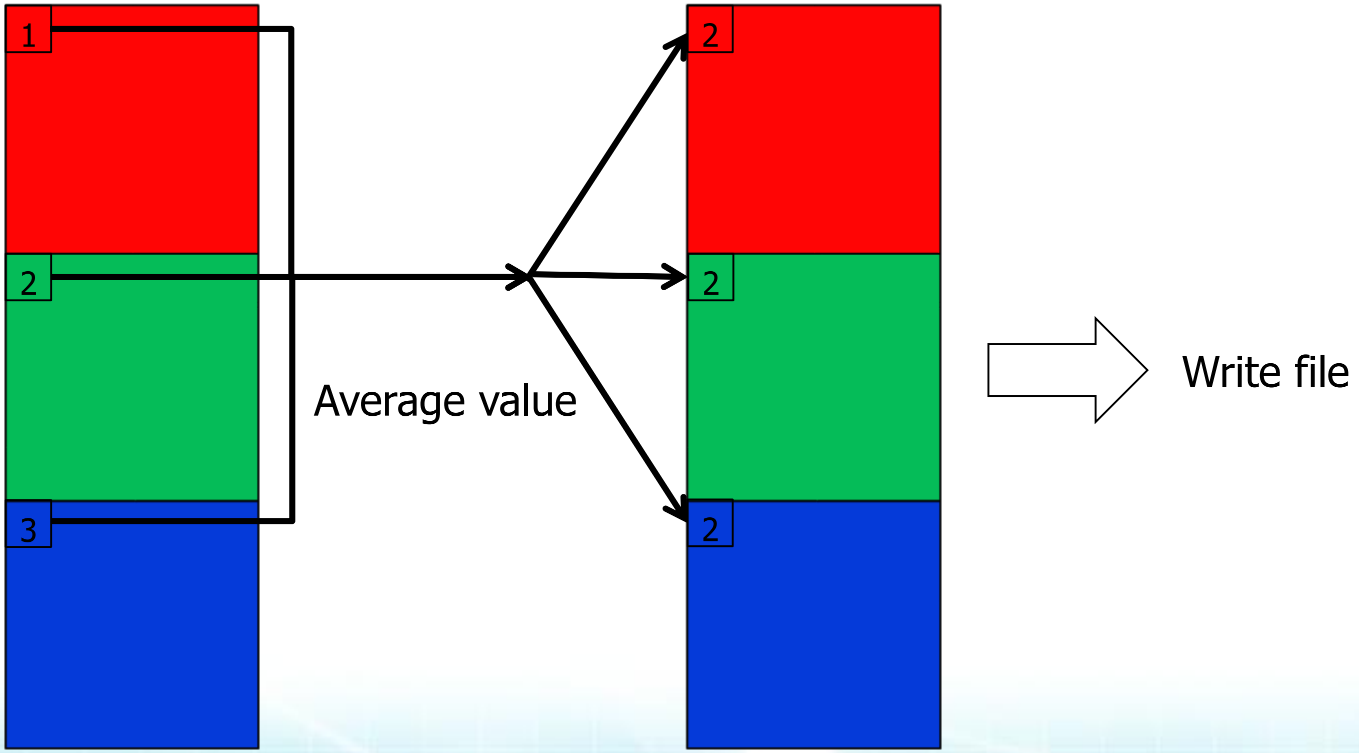
```
        }
```

Example

- Video average

```
Loop(!End of file)
        {
```

Read file



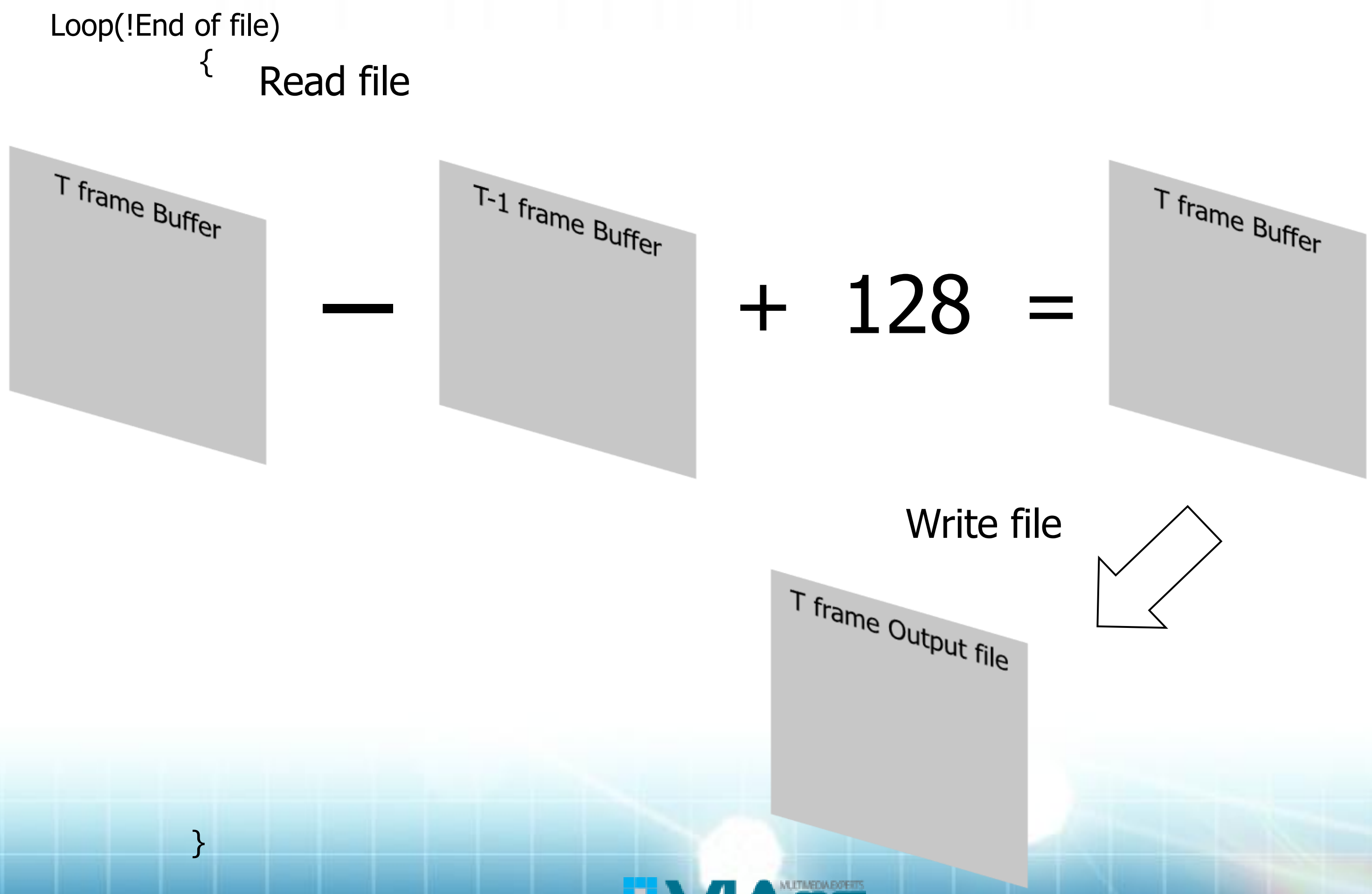
Average value

Write file

```
        }
```

Example

- Video subtraction



Loop(!End of file)

{ Read file

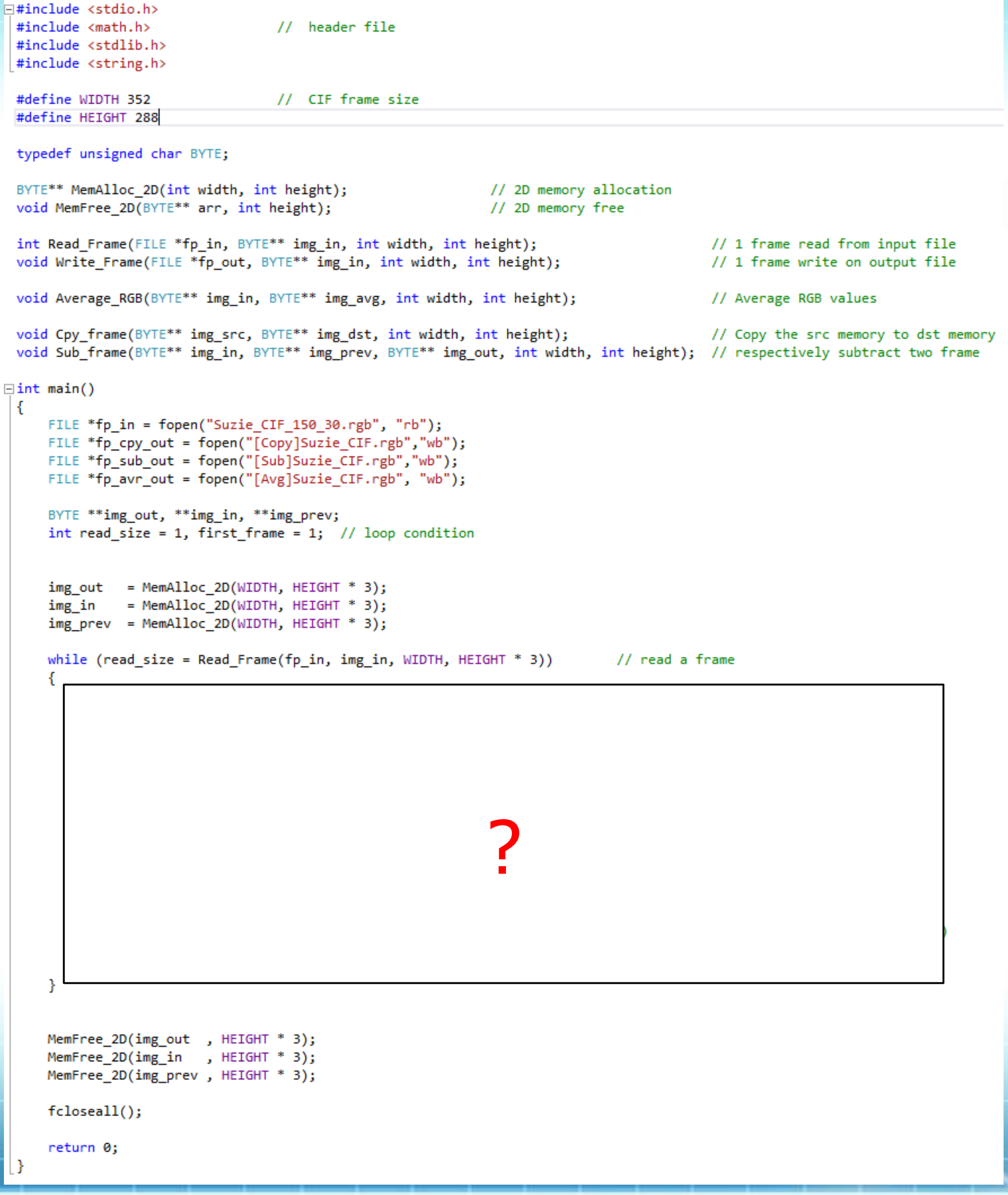
T frame Buffer − T-1 frame Buffer + 128 = T frame Buffer

Write file

T frame Output file

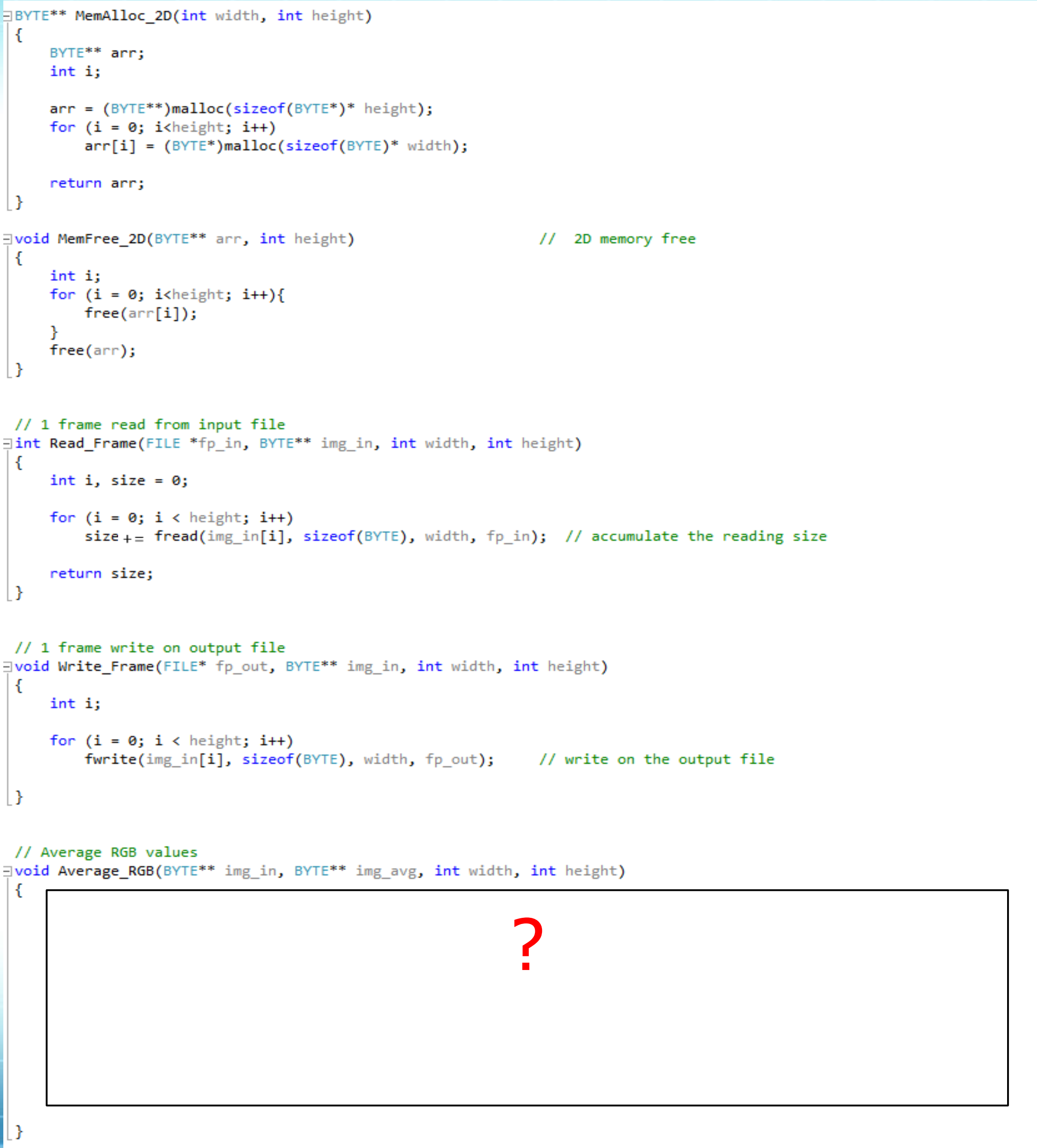
}

Example



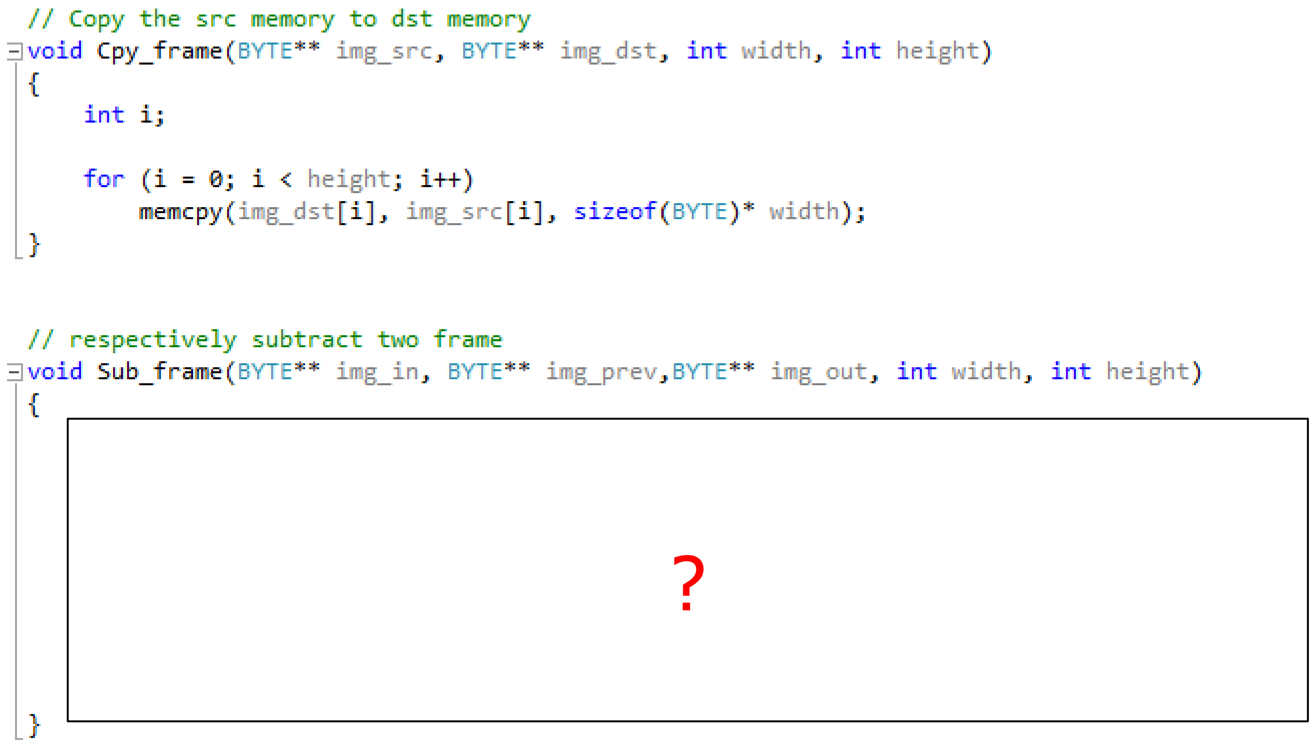
```c
#include <stdio.h>
#include <math.h>                //  header file
#include <stdlib.h>
#include <string.h>

#define WIDTH 352                //  CIF frame size
#define HEIGHT 288

typedef unsigned char BYTE;

BYTE** MemAlloc_2D(int width, int height);                 // 2D memory allocation
void MemFree_2D(BYTE** arr, int height);                   // 2D memory free

int Read_Frame(FILE *fp_in, BYTE** img_in, int width, int height);                   // 1 frame read from input file
void Write_Frame(FILE *fp_out, BYTE** img_in, int width, int height);                // 1 frame write on output file

void Average_RGB(BYTE** img_in, BYTE** img_avg, int width, int height);              // Average RGB values

void Cpy_frame(BYTE** img_src, BYTE** img_dst, int width, int height);               // Copy the src memory to dst memory
void Sub_frame(BYTE** img_in, BYTE** img_prev, BYTE** img_out, int width, int height);  // respectively subtract two frame

int main()
{
    FILE *fp_in = fopen("Suzie_CIF_150_30.rgb", "rb");
    FILE *fp_cpy_out = fopen("[Copy]Suzie_CIF.rgb","wb");
    FILE *fp_sub_out = fopen("[Sub]Suzie_CIF.rgb","wb");
    FILE *fp_avr_out = fopen("[Avg]Suzie_CIF.rgb", "wb");

    BYTE **img_out, **img_in, **img_prev;
    int read_size = 1, first_frame = 1;  // loop condition


    img_out   = MemAlloc_2D(WIDTH, HEIGHT * 3);
    img_in    = MemAlloc_2D(WIDTH, HEIGHT * 3);
    img_prev  = MemAlloc_2D(WIDTH, HEIGHT * 3);

    while (read_size = Read_Frame(fp_in, img_in, WIDTH, HEIGHT * 3))        // read a frame
    {

                                            ?

    }


    MemFree_2D(img_out  , HEIGHT * 3);
    MemFree_2D(img_in   , HEIGHT * 3);
    MemFree_2D(img_prev , HEIGHT * 3);

    fcloseall();

    return 0;
}
```

Example



```
BYTE** MemAlloc_2D(int width, int height)
{
    BYTE** arr;
    int i;

    arr = (BYTE**)malloc(sizeof(BYTE*)* height);
    for (i = 0; i<height; i++)
        arr[i] = (BYTE*)malloc(sizeof(BYTE)* width);

    return arr;
}

void MemFree_2D(BYTE** arr, int height)                    //  2D memory free
{
    int i;
    for (i = 0; i<height; i++){
        free(arr[i]);
    }
    free(arr);
}


// 1 frame read from input file
int Read_Frame(FILE *fp_in, BYTE** img_in, int width, int height)
{
    int i, size = 0;

    for (i = 0; i < height; i++)
        size += fread(img_in[i], sizeof(BYTE), width, fp_in);  // accumulate the reading size

    return size;
}


// 1 frame write on output file
void Write_Frame(FILE* fp_out, BYTE** img_in, int width, int height)
{
    int i;

    for (i = 0; i < height; i++)
        fwrite(img_in[i], sizeof(BYTE), width, fp_out);     // write on the output file

}


// Average RGB values
void Average_RGB(BYTE** img_in, BYTE** img_avg, int width, int height)
{
                                    ?








}
```

Example



```
// Copy the src memory to dst memory
void Cpy_frame(BYTE** img_src, BYTE** img_dst, int width, int height)
{
    int i;

    for (i = 0; i < height; i++)
        memcpy(img_dst[i], img_src[i], sizeof(BYTE)* width);
}


// respectively subtract two frame
void Sub_frame(BYTE** img_in, BYTE** img_prev,BYTE** img_out, int width, int height)
{



                                    ?



}
```