Lab Exercises 20/05/2021: YUV4MPEG2

Esercizio 1

The YUV4MPEG2 format is a simple binary format designed to store uncompressed YCbCr frames formatted in a variety of ways. The extension is typically `.y4m`. The format begins with a STREAM-HEADER consisting of "YUV4MPEG2" characters followed by an arbitrary number of fields (TAGGED-FIELD) ending with the character `0x0A` (\n in C++). The fields have no prefixed order. A TAGGED-FIELD consists of a space character (`0x20`), an ASCII character (the TAG) that identifies its semantics and an arbitrary number (dependent on the tag) of other non-whitespace characters. All numeric values are encoded in decimal ASCII format.

The possible TAGGED-FIELDs for the STEAM-HEADER are:

1. `W` is the width of the image in pixels (integer). Mandatory.
2. `H` is the height of the image in pixels (integer). Mandatory.
3. `C` chroma subsampling/color plane format (string). It can be "420jpeg" (default if not present), "420mpeg2", "420paldv", "411", "422", "444", "444alpha", "mono".
4. `I` is a single character representing the interlacing. It can be "p" (progressive/no interlacing, default if not present), "t" (upper field first), "b" (lower field first).
5. `F` is the frame rate that is a fraction in the form numerator:denominator (integer:integer).
6. `A` pixel aspect ratio that is a fraction in the form numerator:denominator (integer:integer).
7. `X` application dependent field (string without whitespace).

The header is followed by the frames, each preceded by a FRAME-HEADER composed by the "FRAME" characters followed by an arbitrary number of TAGGED-FIELDs ending with the character `0x0A`. Also in this case, the fields have no prefixed order.

The possible TAGGED-FIELDs for the FRAME-HEADER are:

1. `I` frame interlacing (string without whitespace). The string is complex and unnecessary for this exercise.
2. `X` application dependent field (string without whitespace).

Frames are made up of the Y values of all pixels, followed by all Cb values, followed by all Cr values. Cb and Cr are suitably subsampled.

An example of a file might be (seen in a hex editor):

```
Offset(h) 00 01 02 03 04 05 06 07 08 09 0A 0B 0C 0D 0E 0F
00000000  59 55 56 34 4D 50 45 47 32 20 48 32 38 38 20 57  YUV4MPEG2 H288 W
00000010  33 35 32 20 43 34 32 30 6A 70 65 67 0A 46 52 41  352 C420jpeg.FRA
00000020  4D 45 0A 09 2B CC FF F7 FE FE FE FE FE FE FE FE  ME..+Ìÿ÷þþþþþþþþ
...
```

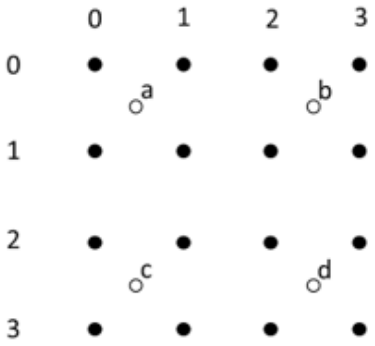
that is:

```
YUV4MPEG2 H288 W352 C420jpeg
FRAME
.+Ìÿ÷þþþþþþþ…
```

In this example, the file contains CIF format (352x288) frames encoded in YCbCr with Cb and Cr subsampled by 2 in width and height.

The 4:2:0 color coding in JPEG requires that every 4 Y pixels only one value of Cb and one value of Cr with coordinates centered with respect to the corresponding 4 Y are kept.

In the image below it is possible to see that for the pixels of position (0,0), (0,1), (1,0) and (1,1) only one value is kept (indicated with "a") is for plan Cb and for plan Cr.



In the file `y4m_gray.cpp` you have to implement the function corresponding to the following declaration:

```
bool y4m_extract_gray(const std::string& filename,
                      std::vector<mat<uint8_t>>& frames);
```

The function must load a file in YUV4MPEG2 format and save the Y planes of all frames in the grayscale image vector `frames`. The program should only support `C420jpeg` encoding with progressive interlacing (*i.e.* no interlacing) and ignore *frame rate*, *aspect ratio* or other parameters. In case of an error it must end with `false`. If the loading of the YUV4MPEG2 stream is successful the function must terminate by returning `true`.

Esercizio 2

The YUV4MPEG2 format is a simple binary format designed to store uncompressed YCbCr frames formatted in a variety of ways. The extension is typically `.y4m`. The format begins with a STREAM-HEADER consisting of "YUV4MPEG2" characters followed by an arbitrary number of fields (TAGGED-FIELD) ending with the character `0x0A` (\n in C++). The fields have no prefixed order. A TAGGED-FIELD consists of a space character (`0x20`), an ASCII character (the TAG) that identifies its semantics and an arbitrary number (dependent on the tag) of other non-whitespace characters. All numeric values are encoded in decimal ASCII format.

The possible TAGGED-FIELDs for the STEAM-HEADER are:

1. `W` is the width of the image in pixels (integer). Mandatory.
2. `H` is the height of the image in pixels (integer). Mandatory.

3. `C` chroma subsampling/color plane format (string). It can be "420jpeg" (default if not present), "420mpeg2", "420paldv", "411", "422", "444", "444alpha", "mono".
4. `I` is a single character representing the interlacing. It can be "p" (progressive/no interlacing, default if not present), "t" (upper field first), "b" (lower field first).
5. `F` is the frame rate that is a fraction in the form numerator:denominator (integer:integer).
6. `A` pixel aspect ratio that is a fraction in the form numerator:denominator (integer:integer).
7. `X` application dependent field (string without whitespace).

The header is followed by the frames, each preceded by a FRAME-HEADER composed by the "FRAME" characters followed by an arbitrary number of TAGGED-FIELDs ending with the character `0x0A`. Also in this case, the fields have no prefixed order.

The possible TAGGED-FIELDs for the FRAME-HEADER are:

1. `I` frame interlacing (string without whitespace). The string is complex and unnecessary for this exercise.
2. `X` application dependent field (string without whitespace).

Frames are made up of the Y values of all pixels, followed by all Cb values, followed by all Cr values. Cb and Cr are suitably subsampled.

An example of a file might be (seen in a hex editor):

```
Offset(h) 00 01 02 03 04 05 06 07 08 09 0A 0B 0C 0D 0E 0F
00000000  59 55 56 34 4D 50 45 47 32 20 48 32 38 38 20 57  YUV4MPEG2 H288 W
00000010  33 35 32 20 43 34 32 30 6A 70 65 67 0A 46 52 41  352 C420jpeg.FRA
00000020  4D 45 0A 09 2B CC FF F7 FE FE FE FE FE FE FE FE  ME..+Ìÿ÷þþþþþþþþ
…
```

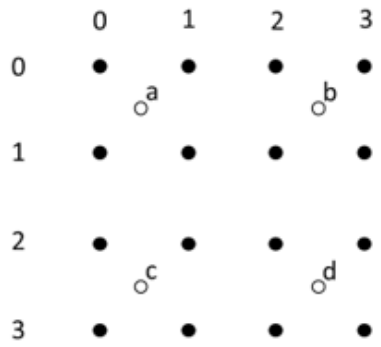
that is:

```
YUV4MPEG2 H288 W352 C420jpeg
FRAME
.+Ìÿ÷þþþþþþþ…
```

In this example, the file contains CIF format (352x288) frames encoded in YCbCr with Cb and Cr subsampled by 2 in width and height.

The 4:2:0 color coding in JPEG requires that every 4 Y pixels only one value of Cb and one value of Cr with coordinates centered with respect to the corresponding 4 Y are kept.

In the image below it is possible to see that for the pixels of position (0,0), (0,1), (1,0) and (1,1) only one value is kept (indicated with "a") is for plan Cb and for plan Cr.



In the file `y4m_color.cpp` you have to implement the function corresponding to the following declaration:

```
bool y4m_extract_color(const std::string& filename,
                       std::vector<mat<vec3b>>& frames);
```

The function must load a file in YUV4MPEG2 format, do the *upsampling* of the Cb and Cr planes, convert the YCbCr planes to RGB using the formulas below and save the image obtained in the `frames` vector. The program should only support `C420jpeg` encoding with progressive interlacing (*i.e.* no interlacing) and ignore *frame rate*, *aspect ratio* or other parameters. In case of an error it must end with `false`. If the loading of the YUV4MPEG2 stream is successful the function must terminate by returning `true`.

For the conversion from YCbCr to RGB use the following formulas:

$$\begin{bmatrix} R \\ G \\ B \end{bmatrix} = \begin{bmatrix} 1.164 & 0.000 & 1.596 \\ 1.164 & -0.392 & -0.813 \\ 1.164 & 2.017 & 0.000 \end{bmatrix} \begin{bmatrix} Y - 16 \\ Cb - 128 \\ Cr - 128 \end{bmatrix}$$

keeping in mind that the values of Y must be between 16 and 235 and those of Cb and Cr must be between 16 and 240. If they are not, they must be saturated in the valid range (for example a value of Y equal to 7 must become a 16, a value of Cb equal to 245 must become 240). Also R, G and B must always be saturated between 0 and 255 before putting them in a byte.