# Y4M_Gray

Submit Solution

Il formato YUV4MPEG2 è un semplice formato binario progettato per memorizzare fotogrammi non compressi YCbCr formattati in diversi modi. L'estensione è tipicamente `.y4m`. Il formato inizia con uno STREAM-HEADER composto dai caratteri "YUV4MPEG2" seguiti da un numero arbitrario di campi (TAGGED-FIELD) terminati con il carattere `0x0A` (il `\n` in *C++*) in ordine qualsiasi. Un TAGGED-FIELD è costituito da un carattere spazio (`0x20`), un carattere ASCII (il TAG) che ne identifica la semantica e un numero arbitrario (dipendente dal tag) di altri caratteri non `whitespace`. Tutti i valori numerici sono codificati in formato ASCII decimale.

I possibili TAGGED-FIELD per lo STEAM-HEADER sono:

- `W` larghezza dell'immagine in pixel: intero. Obbligatorio.
- `H` altezza dell'immagine in pixel: intero. Obbligatorio.
- `C` chroma subsampling/formato dei piani colore: stringa. Può essere "420jpeg" (default se non presente), "420mpeg2", "420paldv", "411", "422", "444", "444alpha", "mono".
- `I` interlacciamento: singolo carattere. Può essere "p" (progressivo/nessun interlacciamento, default se non presente), "t" (prima il campo superiore), "b" (prima il campo inferiore).
- `F` frame rate: rapporto espresso come intero:intero.
- `A` rapporto di forma dei pixel: rapporto espresso come intero:intero.
- `X` campo dipendente dall'applicazione: stringa (senza `whitespace`).

Dopo l'header vengono i fotogrammi, ognuno preceduto da un FRAME-HEADER composto dai caratteri "FRAME" di nuovo seguiti da un numero arbitrario di TAGGED-FIELD terminati con il carattere `0x0A` in ordine qualsiasi.

I possibili TAGGED-FIELD per il FRAME-HEADER sono:

- `I` interlacciamento del frame: stringa (senza `whitespace`). La stringa è complessa e non necessaria per questo esercizio.
- `X` campo dipendente dall'applicazione: stringa (senza `whitespace`).

I fotogrammi sono costituiti dai valori di Y di tutti i pixel, seguiti da tutti i valori Cb, seguiti da tutti i valori Cr. Cb e Cr sono sottocampionati opportunamente.

Un esempio di file potrebbe essere (visto in un editor esadecimale):

```
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
…
```

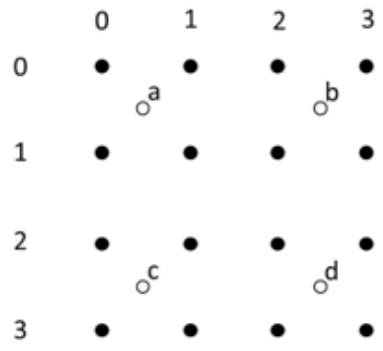
ovvero:

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

In questo esempio, il file contiene fotogrammi in formato CIF (352x288) codificati in YCbCr con Cb e Cr sottocampionati di 2 in larghezza e altezza.

La codifica dei colori 4:2:0 alla JPEG prevede che ogni 4 pixel Y venga tenuto un solo valore di Cb e un solo valore di Cr con coordinate centrate rispetto ai 4 Y corrispondenti.

Nell'immagine qui sotto è possibile vedere che per i pixel di posizione (0,0), (0,1), (1,0) e (1,1) viene mantenuto un solo valore (indicato con "a") sia per il piano Cb che per il piano Cr.




Nel file `y4m_gray.cpp` si implementi la funzione corrispondente alla seguente dichiarazione:

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

La funzione deve caricare un file in formato YUV4MPEG2 e salvare i piani Y di tutti i frame nel vettore di immagini in scala di grigio `frames`. Il programma dovrà supportare solo la codifica `C420jpeg` con interlacciamento progressivo (ovvero senza interlacciamento) e ignorare *frame rate*, *aspect ratio* o altri parametri. In caso di errore deve terminare ritornando `false`. Se il caricamento dello stream YUV4MPEG2 va a buon fine la funzione deve terminare ritornando `true`.