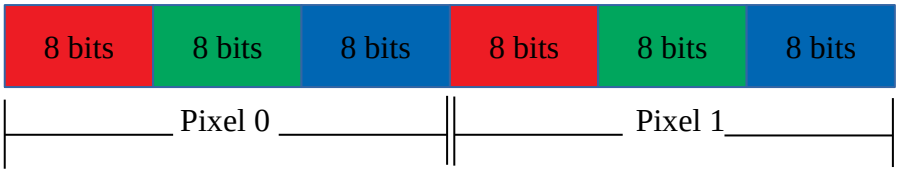
Results RGB to YUV using C++

✦ Image formats required

- Input is RGB888 (RGB24):

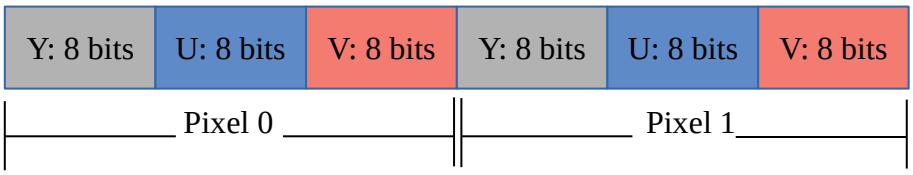
  - Pixels of 24 bits: 8 bits of red, 8 bits of green and 8 bits of blue.
  - Packed format.
  - Without alpha component.
  - Image size of 640x480 pixels.



RGB888 color format

- Output is YUV444:

  - Pixels of 24 bits: 8 bits of Y, 8 bits of U(C'b) and 8 bits of V(C'r).
  - Packed format.
  - Without alpha component.
  - Image size of 640x480 pixels.



YUV444 color format

✦ Algorithm description

The algorithm implemented first take all bits of the input file (The name of the file is stored in the variable input_image) and store all of them into an array of chars. Next, a casting is applied over that array to get the same values, but stored into an array of floats.

That array has the three components (RGB) of each pixel of the image.

The next step is to apply the conversion for each pixel from RGB to YUV, to do that, only is needed to apply the following matricial operation.

$$Y = (0.257 * R) + (0.504 * G) + (0.098 * B) + 16$$

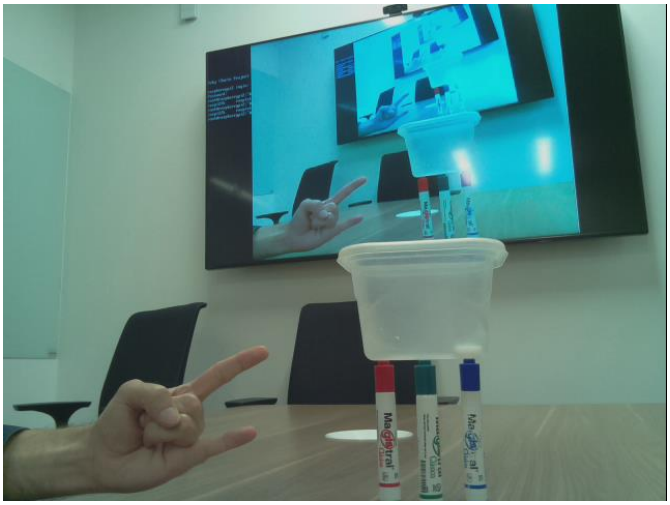
$$U = Cb = -(0.148 * R) - (0.291 * G) + (0.439 * B) + 128$$

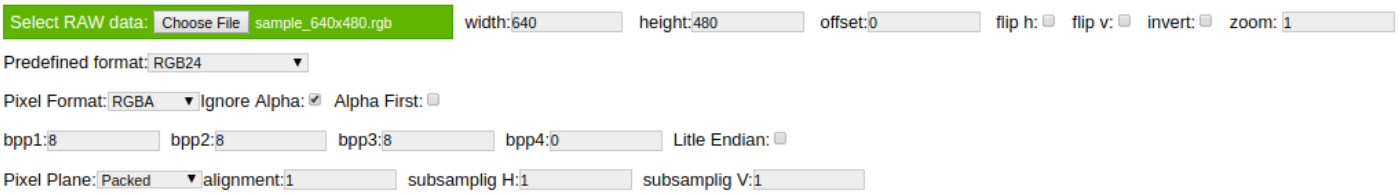
$$V = Cr = (0.439 * R) - (0.368 * G) - (0.071 * B) + 128$$

Each new YUV pixel is stored in a new array that then is written to the new image in YUV format.

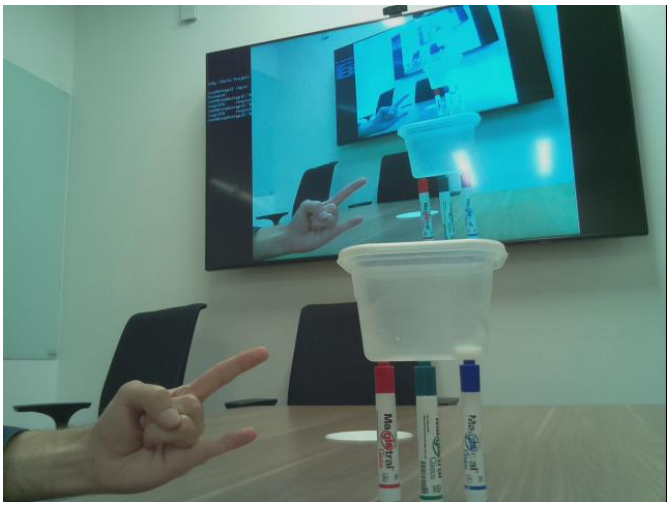
✦ Sample image

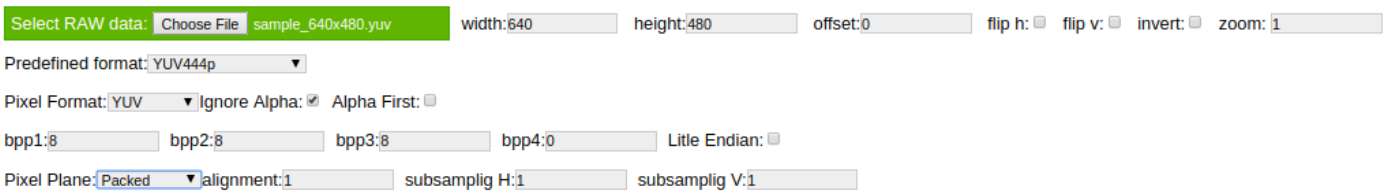
- Input image (RGB888):





- Output image (YUV444):





✦ Processing time

Rgb2yuv-c was executed 5 times to get the average processing time. The following table summarize the results.

	C++	
1	151359	us
2	151215	us
3	151318	us
4	151108	us
5	151345	us
Average	151269	us