Multimedia Services and Applications (SAM)
2021/2022

Third Lab Assignment Report
Image Processing in Matlab

João Carlos Machado Rocha Pires (UP201806079)

U.PORTO
FEUP FACULDADE DE ENGENHARIA
UNIVERSIDADE DO PORTO

Contents

1. Blue background extraction

1.1 Basic Segmentation

For this first exercise, I've used the following script (based on the 'segmentBB2019.m'):

```
function basicsegmentation()

close all;
disp('Select a picture');
[filename, pathname] = uigetfile('*.*', 'open the picture');
fullname=fullfile(pathname,filename);
image=imread(fullname);

figure(1), imshow(image), title('Original image');

[height, width, planes] = size(image);

if(planes==3)
    r = image(:, :, 1);
    g = image(:, :, 2);
    b = image(:, :, 3);

figure(2), imshow([r g b]), title('RGB components');

figure(3),imhist(uint8(b)), title('Blue channel histogram');

threshold=input('Which threshold?');

BWforeground=zeros(height, width);
for i=1:height
    for j=1:width
        if(b(i,j)<threshold) BWforeground(i,j)=255;
        end
    end
end

figure(4), imshow(BWforeground),title('B&W segmented image');

imwrite(BWforeground, 'black_white.jpg');
```

```
end
```

This script allows the user to select a picture, then shows the original and the RGB components. After that, before prompting the user to select the threshold value, shows the histogram for the B(lue) component and, finally, creates a picture with the same size as the original, having in white the pixels whose B component is less than the threshold and in black all the others, saving the final result in a new picture.

For this exercise, I tested the script above for the pictures 'birdBB.jpg' and 'jump.jpg', as follows:

1.1.1 'jump.jpg'

The original picture is the following:

Figure 1.1: Original picture

The RGB components are the following ones:

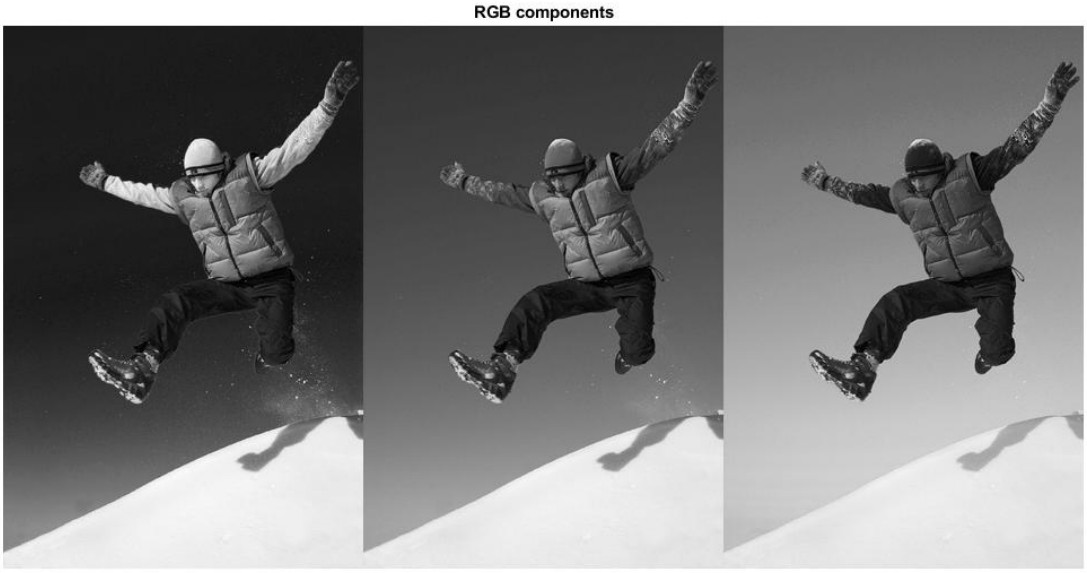


Figure 1.2: RGB components

The histogram for the B component is the following one:

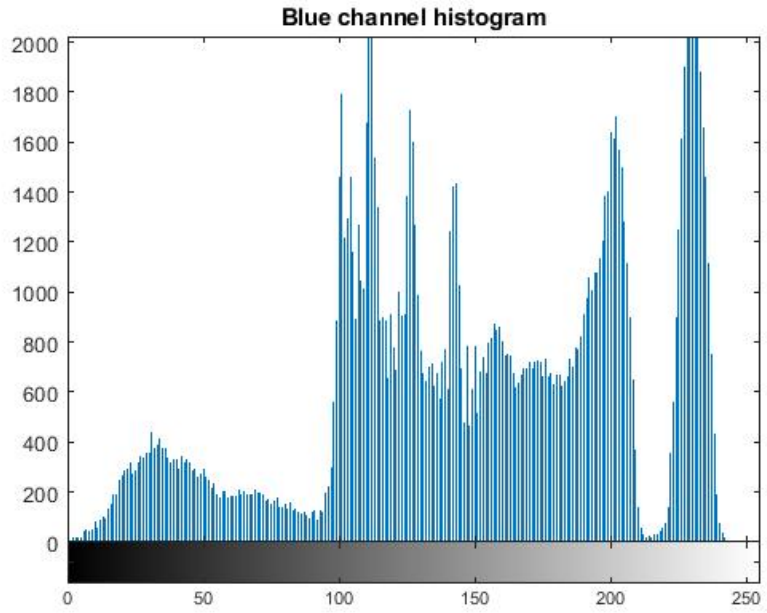


Figure 1.3: Histogram for B component

By analysing the histogram, we can get to know that the higher values come more or less above 94 (x axis), which means this would be a good threshold. However, it does not produce perfect results, as we can see bellow, since the man jumping is not completely white. Some pixels of him are still black. Nevertheless, it's the value with best results (from the ones I tested), as values under 94 (like 58) make most of the man black and above 94 (100 and 110), the higher the value, the more sky is considered as foreground, which is not supposed to happen.

(a) Threshold = 58

(b) Threshold = 94

(a) Threshold = 100

(b) Threshold = 110

Figure 1.5: Different results for the BW picture for different threshold values.

1.1.2 'birdBB.jpg'

The original picture is the following:

Figure 1.6: Original picture

Comparing with the one above, this time, the blue background is more uniform, while in the 'jump.jpg' the background moved from darker blue (top) to white blue (bottom). However, the shape

of the man jumping was more visually delimited that the bird in this picture, specially its wings, which make difficult to separate the foreground from the background.

The RGB components are the following ones:

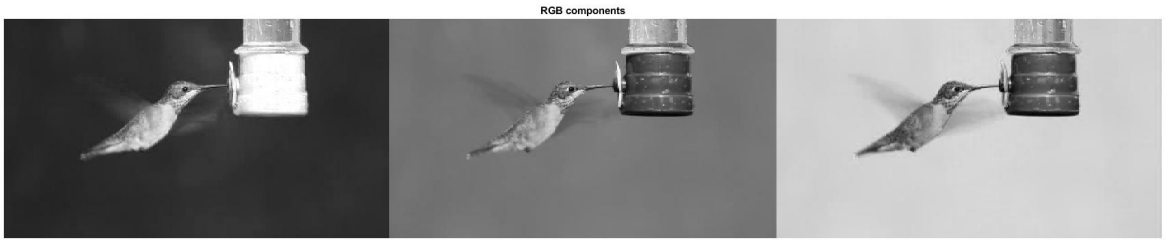


Figure 1.7: RGB components

The histogram for the B component is the following one:

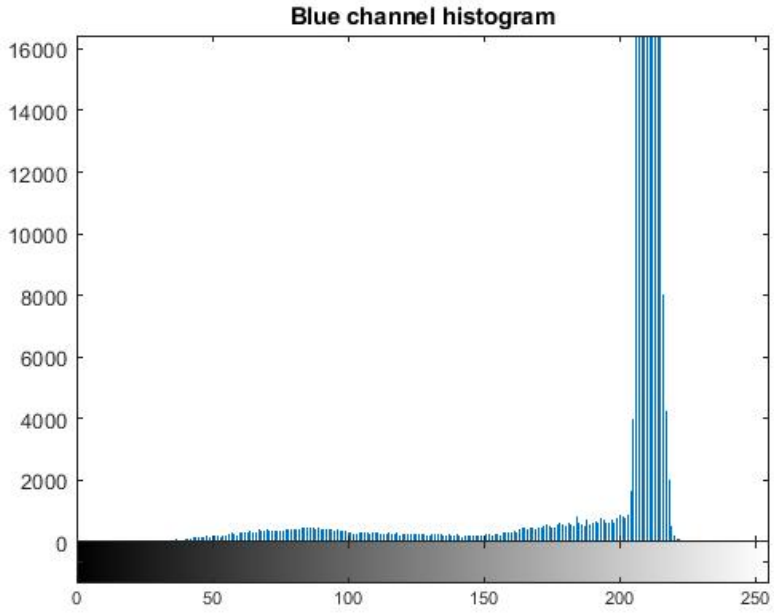
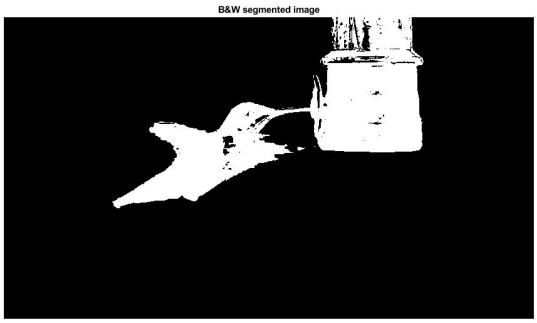
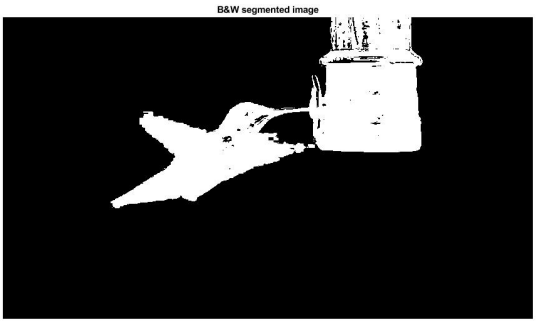


Figure 1.8: Histogram for B component

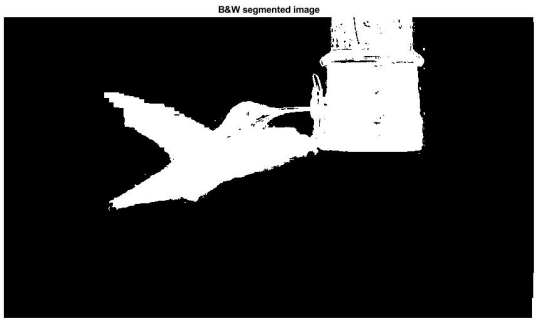
By analysing the histogram, we can get to know that the higher values come more or less above 203 (x axis), which means this would be a good threshold. However, it does not produce perfect results, as we can see bellow, since the bird assumes a strange shape, specially its wings. Nevertheless, it's the value with best results (from the ones I tested), as values under 203 (like 184 and 190) make the bird delimiter constantly smaller and the water supplier containing more black (undesired) parts and above 203 (215) the foreground detection is not accurate at all.



(a) Threshold = 184



(b) Threshold = 190



(a) Threshold = 203

(b) Threshold = 215

Figure 1.10: Different results for the BW picture for different threshold values.

1.1.3 Final thoughts

For the pictures used, one advantage was the fact that the foreground we wanted to separate from the background didn't have any blue parts. However, both pictures had some problems (the man jumping had different blue tones in the background and the bird wings made difficult the delimitation task).

Another interesting aspect was the fact that in the first picture, the snow and the shadow of the man don't appear in the foreground, which might indicate that there might be a low blue value in zones of the background and still we can obtain good results. The same happens for high blue values in some parts of the foreground objects. If they are not so many, they'll probably be not noticed on the integration of the foreground with the new background.

Answering the question "Is it always true that a pixel with a high value in the B component is always blue?", the answer is not, because we saw that in the first picture we had the snow part with high values in the B component histogram, and the snow is white, not blue.

1.2 Alternative Segmentation

In this experiment, similar to the one made before, but this time considering that a pixel is really blue if it has high values in the B component and low values in the other components ("blueness" of a pixel = B - max(R,G)), I've used the following script:

```
function alternativesegmentation()

close all;
disp('Select a picture');
[filename, pathname] = uigetfile('*.*', 'open the picture');
fullname=fullfile(pathname,filename);
image=imread(fullname);

figure(1), imshow(image), title('Original image');

[height, width, planes] = size(image);

if(planes==3)
    r = image(:, :, 1);
    g = image(:, :, 2);
    b = image(:, :, 3);

figure(2), imshow([r g b]), title('RGB components');

blueness = double(b) - max(double(r), double(g));

figure(3), imshow(uint8(blueness)), title('blueness channel');

figure(4), imhist(uint8(blueness)), title('blueness channel histogram');

threshold=input('Which threshold?');

BWforegroundBlueness=ones(height,width);

BWforegroundBlueness=blueness<threshold;

figure(5), imshow(BWforegroundBlueness), title('B&W segmented image using blueness');

imwrite(BWforegroundBlueness, 'bw_blueness.jpg');

end
```

The script does the same as the one above until the RGB components separation. Then, it

calculates the blueness and shows both the blueness result as the corresponding histogram, asking then the user to insert the threshold value and, finally, creates and shows the BW segmented image using blueness.

For this experiment, I've used the picture 'christmasBB.jpg':

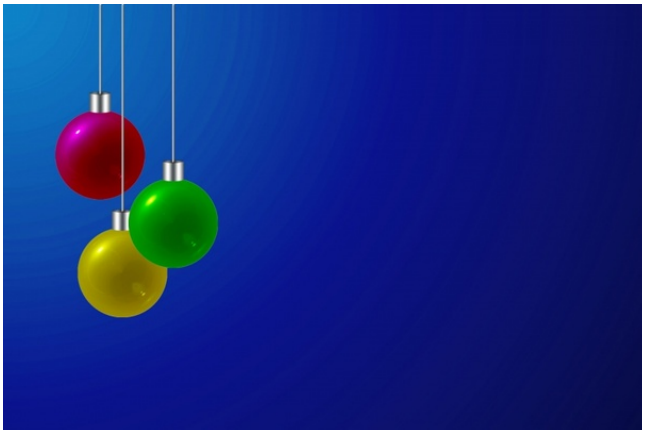


Figure 1.11: Original picture

The RGB components are the following ones:

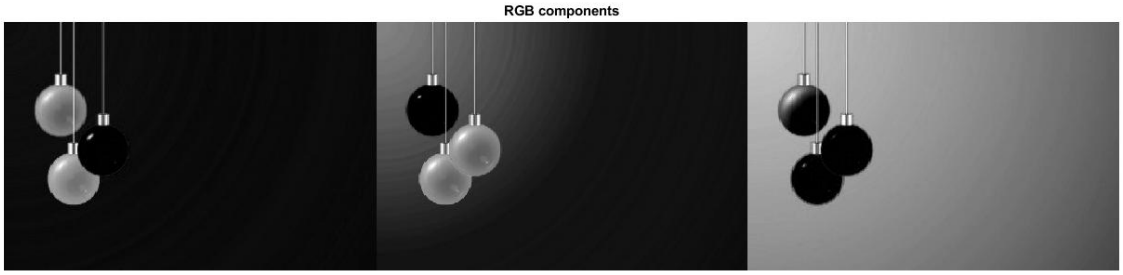
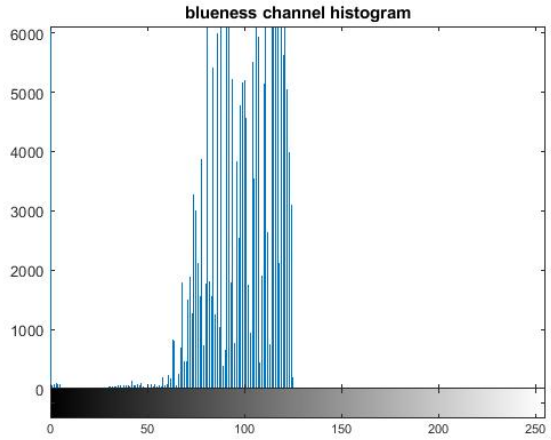
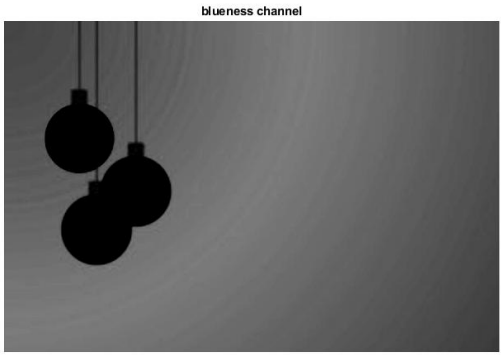


Figure 1.12: RGB components

The blueness channel histogram and result are the following:



(a) Blueness Channel Histogram

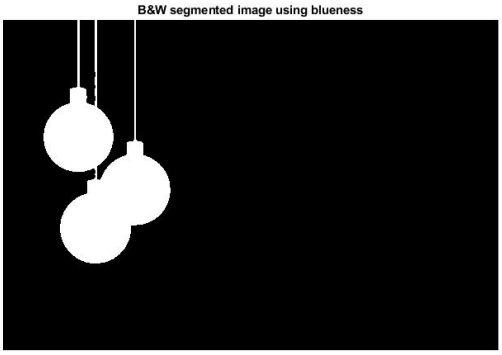


(b) Blueness Channel

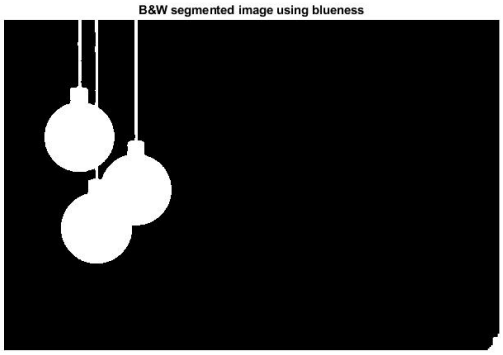
Figure 1.13: Blueness

After proceeding with an analysis of the histogram, I've tested the script with 4 threshold values (50, 61, 67 and 81).

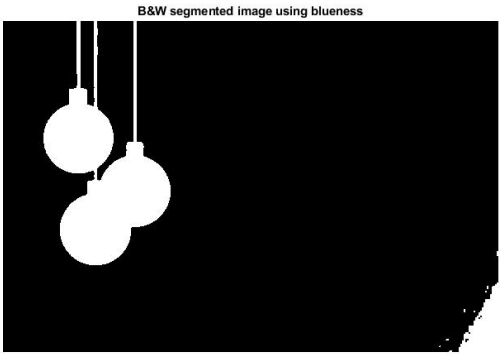
The best results were obtained for the threshold value 50.



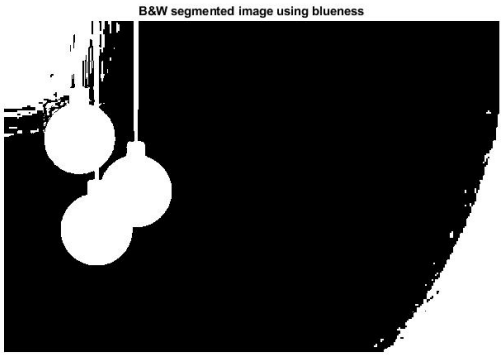
(a) Threshold = 50



(b) Threshold = 61



(a) Threshold = 67



(b) Threshold = 81

Figure 1.15: Different results for the BW segmented image using blueness for different threshold values.

2. Adding objects to another image

For this last experiment, I've used the following script. It does the same as above, but, in the end, first generates the colored foreground using blueness and then prompts the user to select a background to merge with the obtained foreground.

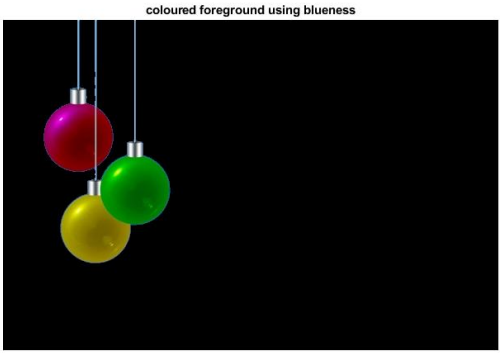
```
function mergefgbg()

close all;
disp('Select a picture');
[filename, pathname] = uigetfile('*.*', 'open the picture');
fullname=fullfile(pathname,filename);
image=imread(fullname);

[height, width, planes] = size(image);

if(planes==3)
    r = image(:, :, 1);
    g = image(:, :, 2);
    b = image(:, :, 3);

blueness = double(b) - max(double(r), double(g));

threshold=input('Which threshold?');
```

```
BWforegroundBlueness=ones(height,width);

BWforegroundBlueness=blueness<threshold;

foregroundR=zeros(height, width);
foregroundG=zeros(height, width);
foregroundB=zeros(height, width);

for i=1:height
    for j=1:width
        if(blueness(i,j)<threshold)
            foregroundR(i,j)=r(i,j);
            foregroundG(i,j)=g(i,j);
            foregroundB(i,j)=b(i,j);
        end
    end
end

foregroundRGB=cat(3,uint8(foregroundR),uint8(foregroundG),uint8(foregroundB));
figure(1), imshow(foregroundRGB),title('coloured foreground using blueness');

close all;
disp('Select a background image');
[filename2, pathname2] = uigetfile('*.*', 'open image');
fullname2=fullfile(pathname2,filename2);
image2=imread(fullname2);
outputImage=imfuse(image2,foregroundRGB);
figure(2), imshow(uint8(outputImage)),title('fused image');

end
```

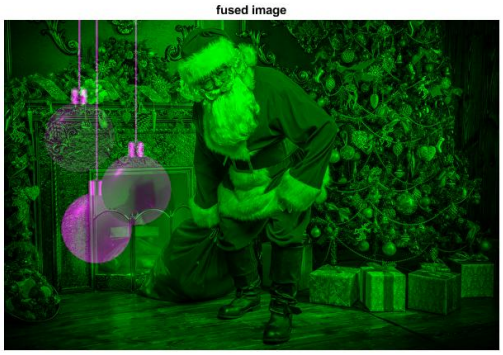
I've tested this script with the 'christmasBB.jpg' (threshold = 50) as foreground and the following picture as background:

Figure 2.1: Background image

The result was:



(a) Colored Foreground using Blueness



(b) Fused image

Figure 2.2: Result of the merge of the foreground with a new background.