MO446 – Introduction to Computer Vision
Project 0

Breno Leite
Guilherme Leite

10/08/2017

Question 1 - Input images

The Figure 1 was used as an input to all the exercises, as specified it is a colored image, rectangular and its type is *PNG*. The image dimensions are 400x300, which satisfies the required size.

Figure 1: Input image used in the questions (**p0-1-0**)

Note: The image was converted from *JPG* to *PNG* format.

Important note: *The borders seen in the figures are not part of the image, they are figurative information about the starting and ending points of the image. Moreover, all the image scales in this report were changed in order to make the text more readable.*

Question 2 - Color planes

2-a) A PNG image is represented in RGB format, which means that there are three channels for each pixel. These channels represents the intensity of each color, in a determined pixel, RGB stands for: Red, Green, and Blue. The Figure 2 represents an image obtained by swapping the red and blue channel in the input image.

Figure 2: Input image with red and blue channels swapped (**p0-2-a-0**)

We can note a heavy intensity of green in the image after the swap, the reason for that is explained looking at the original image. Most of the pixels in the original image has a low intensity of red, in opposite, the intensity of blue is quite high mainly because of the water and sky. When the swap occurs, all the blue intensity goes to red highlighting things that had a little bit of red intensity on the original image, like the fences.

In other hand, all the intensity from the red (really low on the original image) goes to blue. So, that makes the intensity of blue goes down in places that blue was too high (e.g. water and sky). This way, the green color becomes more intense on places that blue was predominant on the original image, explaining the green lake.

2-b) A monochrome or grayscale images is composed by a single channel, which ranges from 0 to 255 as RGB channels. Thus, create a monochrome image from the green channel is a straightforward procedure, it is just a copy of the green channel into a new image. The result obtained using this procedure on the input image is seen on image (**p0-2-b-0**).

Figure 3: Monochrome image created from green channel (**p0-2-b-0**)

As shown in the figure, the representation of the image stills really good. The reason for this is the high intensity of the green channel in the original image, which makes this channels, by its own, a good representation for image.

2-c) Using the same structure as in **2-c**, we this time use the red channel to create the new monochrome image, which is shown on image (**p0-2-c-0**).

Figure 4: Monochrome image created from red channel (**p0-2-c-0**)

In this image, we are able to see more blurred areas. The reason for this result is the low intensity of the red channel over the image pixels, which makes just the channel red not as good representation as the green channel.

2-d) The image created from the green channel (**p0-2-b-0**) looks more like what we expected, this is explained by three channels of the image. As said before, the original image is more intense on the green channel than in the red channel[1]. We normally would expect that the green approach better express the image, mainly because the nature of the green color in the real world. However, we can not expect this result for every image.

In contrast, we would expect a computer to work better in an image extracted from the red channel, the intuition is that the color red gives a better contrast to the image. In this way, the information like shapes of objects on the image are highlighted in the monochrome image, one example is the contrast between the sky and the clouds on image (**p0-2-c-0**).

Question 3 - Replacements of pixels

In this question we merge the results shown in Figures 3 (**p0-2-b-0**) and 4 (**p0-2-c-0**). In order to do that, we inserted the centered 100x100 pixels from image **p0-2-b-0** into the image **p0-2-c-0**, the result is shown in Figure 5.

Figure 5: Pixels centered (100x100) from Figure 3 into Figure 4 (**p0-3-0**)

After that, we inserted the generated image (**p0-3-0**) into its respective channel (green) in the original image. The results is shown in Figure 6.

Figure 6: Original image with green channel replaced from Figure 5 (**p0-3-1**)

[1]Mean values for each channel in the original image: Red: 127.40, Green: 149.85, Blue: 173.02

As we can see, the image is almost the same as the original input, except by the center part. If we recall, the Figure 3 is formed by two different images. And, each of them are monochrome images created from the green and red channels of the original input. The Figure 7 shows how the channels representations is represented in the Figure 6, which was created by the insertion.

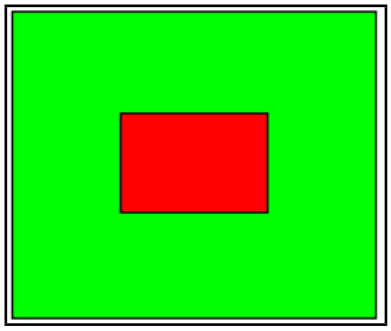


Figure 7: Channels representation layout in the Figure 5

When we replace the image **p0-3-0** inside the green channel of the original input, all the pixels outsize the box at the center are equal, which keeps the same pixels for all the outside-box part. In contrast, the pixels inside the box, which represents the red color, overwrites the green pixels in that region forming a different image.

Question 4 - Arithmetic and geometric operations

4-a) The minimum and maximum values from the image **p0-2-b-0** (*img-green*) are 0 and 255, respectively. The mean value is 149,85 and the deviation is 57,94. The computation for all these values were made using the python library *numpy*, which is a library design for scientific computing.

All these function are already implemented into the *numpy* library, so, instead of code snippets we will be showing the function names and the basis formulas that each function is implemented. The functions *numpy.min()* and *numpy.max()* find the highest and lowest element in a given array, this procedure implemented by *numpy* is pretty straightforward, so we will not show any formulas.

Moreover, in order to obtain the mean, we used the *numpy.mean()* that returns the average value given an array. This function implements the following formula:

$$\bar{X} = \frac{\sum_1^n X_i}{N}$$

where X_i is each pixel in the image and N is the total number of pixels in the image. The standard deviation is implemented by the function called *numpy.std()*, this function is based in the following equation:

$$S = \frac{\sqrt{\sum (x_i - \bar{x})^2}}{N}$$

In which, X_i is each pixel in the image, $\bar{X}$ is the mean value and N is the total number of pixels in the image.

4-b) In this question the image **p0-2-b-0** (*image-green*) was used, we subtracted the mean value from all of the pixels, divided them by the standard deviation, multiplied them by 10 and added back the mean value. Figure 8 is the output image after these operations.

Figure 8: Normalized image from image-green (**p0-4-b-0**)

The image lost a great portion of it's contrast, which could help removing noise from the image or maybe adjust the overall contrast of it. The idea behind it is that, the operations applied reduces the presence of pixels with values too far apart from the others, meaning, pixels that diverge too much are normalized within a certain range of values.

4-c) The image **p0-2-b-0** was also used in this question, so, we shifted twice all pixels to their left. In here, we could either fill the new values in the right size with zero, or we could move the shifted-off part to the right part of the image. The first one was selected because of simplicity to work. The Figure 9 shows the result image after the shift.

Figure 9: 2 pixels shifted image from image-green (**p0-4-c-0**)

After the previous steps, we subtracted the shifted image to the original (not shifted), making

sure all the values were valid ones. After this process, the image shown in Figure 10 was created.

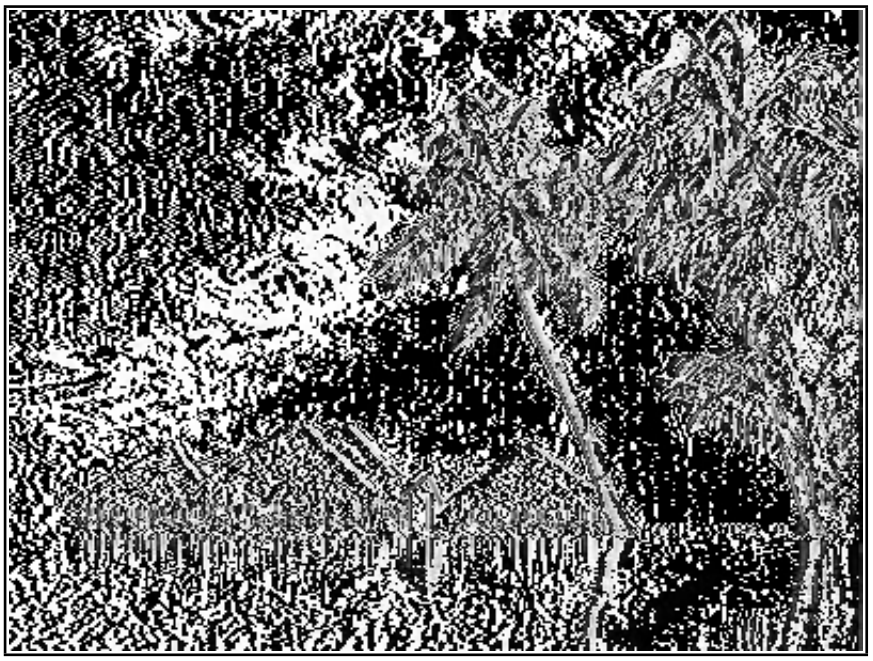


Figure 10: Subtracted image (p0-4-c-0) to (p0-1-0) (**p0-4-c-1**)

The negative values means that the right adjacent pixel was a brighter than the actual pixel, in other words, the intensity of that specific color at that specific pixel is higher than the right pixel next to it.

Question 5 - Noise

In this question we added some noise to the input image, using a Gaussian function. The Gaussian function normally depends of three variables, however, we are going to vary just the σ, which is responsible for the range of the generated numbers.

5-a) In this item, we use the generated number to add a noise into the green channel of the image. The Figure 11 shows different images obtained with different values to σ in the Gaussian function.

(a) $\sigma = 0.5$ (b) $\sigma = 1$ (c) $\sigma = 5$ (d) $\sigma = 10$

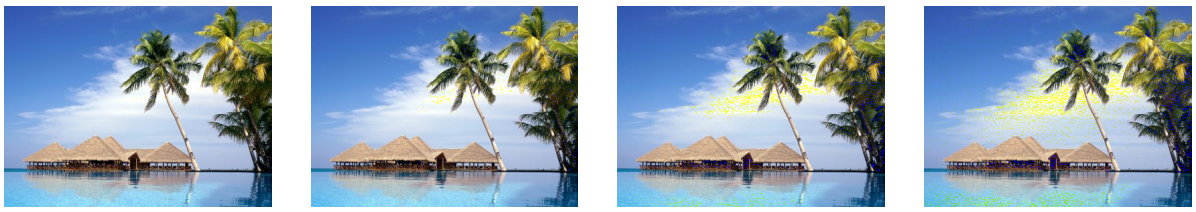
Figure 11: Noise in the green channel from the Gaussian function with different σ values

As we can see, the number of noise in the image increases proportionally with the σ. The noise is noticeable since $\sigma = 0.5$, the reason for that is the color of the pixels added. As we are changing the values to the green channel, it makes colors that were white become something similar to pink, which is pretty contrasting between white pixels. We can also see some green points, those come obviously because sometimes the Gaussian adds some intensity to some dark green pixel points.

The final result reported was obtained using a $\sigma = 10$, and the Figure 12 shows the result.

Figure 12: p0-5-a-0

5-b) In this question, we used the same function as before, but now we add the values to the blue channel instead of green. The Figure 13 shows the result when using different σ values.



(a) $\sigma = 0.5$ (b) $\sigma = 1$ (c) $\sigma = 5$ (d) $\sigma = 10$

Figure 13: Noise in the blue channel from the Gaussian function with different σ values

As the other example, the number of noise increases with the σ value. However, as we change the blue channel white pixels tend to become a yellow color, which makes the contrast smoother than the pink as seen before.

The Figure 14 shows the result submitted in the project, which is using $\sigma = 10$.

Figure 14: p0-5-b-0

5-c) The image **p0-5-b-0** looks better, the reason is the low level of the contrast that happens between the white and yellow pixels. In the image **p0-5-a-0**, the high contrast between white and pink makes the image looks too different from the original version.