Computer Vision: Project 0

Due on August, 2017

(191079) Nathana Facion | (192803) Rafael Mariottini Tomazela

Contents

Question 1

Input images. Find an interesting image to use (img). It should be color, rectangular in shape (that is not the same length and width). Make sure that either dimension (height or width) do not exceeds 512 pixels. Store the image in the input folder using the name convention. (Note that it should be named p0-1-0.png. The names on the questions are for referencing them in the project's text.)

p0-1-0

Question 2

(a) Swap the red and blue channels of the input image, img. Store it in the output folder. (Note that this should be named p0-2-a-0.png, and in contrast to the following items it is a color image.)

(b) Create a monochrome image (img-green) by selecting the green channel of the input image. Store it in the output folder. (Note that this should be named p0-2-b-0.png, and so on.)

(c) Create a monochrome image (img-red) by selecting the red channel of the first input image. Store it in the output folder.

(d) Which image looks more like what you would expect a monochrome image to look like? Would you expect a computer vision algorithm to work on one better than the other? Why? (Note that since this is a question you should answer it in the report.)

Question 2 continued on next page...

(a)

p0-2-a-0

(b)

p0-2-b-0

(c)

p0-2-c-0

(d) We would expect a monochrome image to look more like the one in (b), since the other one is too bright. And it depends on what the algorithm wants to do. Since we think that the image in (b) is closer to a monochrome version of the image, it makes sense to consider it a good one for a computer vision algorithm (a worse monochrome version of the image could lead to wrong results). We could also argue that the algorithm can differentiate better between regions in (b).

Question 3

Replacements of pixels. Lets call the monochrome image from your answer to the last item of the previous question A, and the other one B. Take the center square region of 100 x 100 pixels of A and insert it into the center of B. Store the generated image in the output folder. Replace the respective channel of B into the original image. Store it in the output folder. What do you see? Explain.

p0-3-a-0

p0-3-b-0

The channel of B is red, and the channel of A is green. Since in the center of the image there is almost no green, then when we put this square into the red channel, we will have almost no red there. And that's the case here: the red parts that should make the heart and background purple, disappear, and there is mostly the blue left from the purple.

Question 4

Arithmetic and geometric operations

(a) What is the min and max of the values of img-green? What is the mean? What is the standard deviation? How do you compute them? Provide code snippets in your report for these questions.

(b) Normalize the values of img-green. That is, subtract the mean from all pixels, then divide them by the standard deviation. Then multiply by 10 (if you are using an image representation from 0 to 255) or by 0.05 (if you are using a range from 0 to 1). Now add the mean back to each pixel. Store the resulting image in the output folder. How the image looks like? Do you have an idea of what happened?

Question 4 continued on next page...

(c) Shift img-green to the left by 2 pixels. Store the resulting image in the output folder. Subtract the shifted version to the original, and save the difference image. Make sure that all the values you are saving are legal so you can see all the differences. What do the negative values mean? What do you see in the result?

(a)
Min of green channel: 0.00
Max of green channel: 226.00
Mean of green channel: 112.77
Std of green channel: 53.57

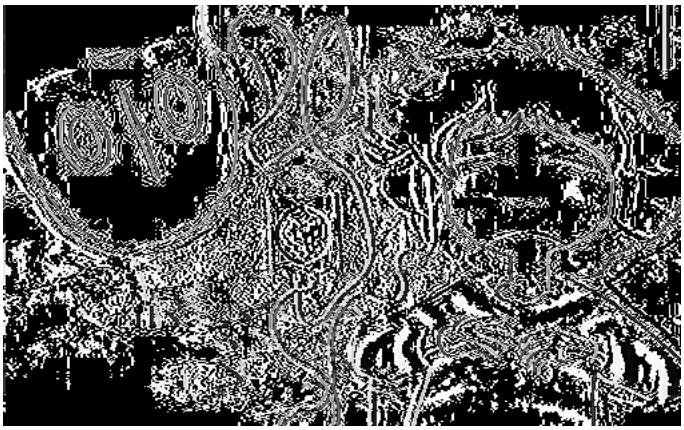
Code snippet:

```
# letter a
print("Min of green channel: %.2f" % g.min())
print("Max of green channel: %.2f" % g.max())
print("Mean of green channel: %.2f" % g.mean())
print("Std of green channel: %.2f" % g.std())
```

g being the green channel from the image
(b)

p0-4-b-0

The image was smoothed, there is less contrast. This happened because the standard deviation got lower, and that means the intensity of the pixels got closer to the mean value, leading to less contrast.
(c)



p0-4-c-0

A negative value means simply that there was a difference between the intensity of the pixel, and the one two pixels in its left (the value could also be positive in some cases, but they are negative only when there is a intensity difference). Since when the intensities are equal the difference is zero, then these areas are black. So what happens is that we can detect borders in a simple way.

Question 5

5. Noise

(a) Take the original colored image (img) and add Gaussian noise to the pixels in the green channel. Increase the sigma until the noise is visible in the image. Store the image into the output folder. What is the value of sigma you used? How did the noise in the resulting images behave? What did you observe?

(b) Add the noise using the same sigma to the blue channel instead. Store the output.

(c) Which image looks better? Why?

(a)

p0-5-a-0

The chosen sigma value was 30. The noise simply made a lot of contrasting pixels in the image.

(b)

p0-5-b-0

(c) In the second one the noise is not as visible. Maybe the reason is that we have less blue receptors in our eyes, so changes in that channel may have a less perceptible effect on the image.