Université de Genève

Imagerie Numérique
13X004

TP 4: Histograms and point operations

Author: Joao Filipe Costa da Quinta

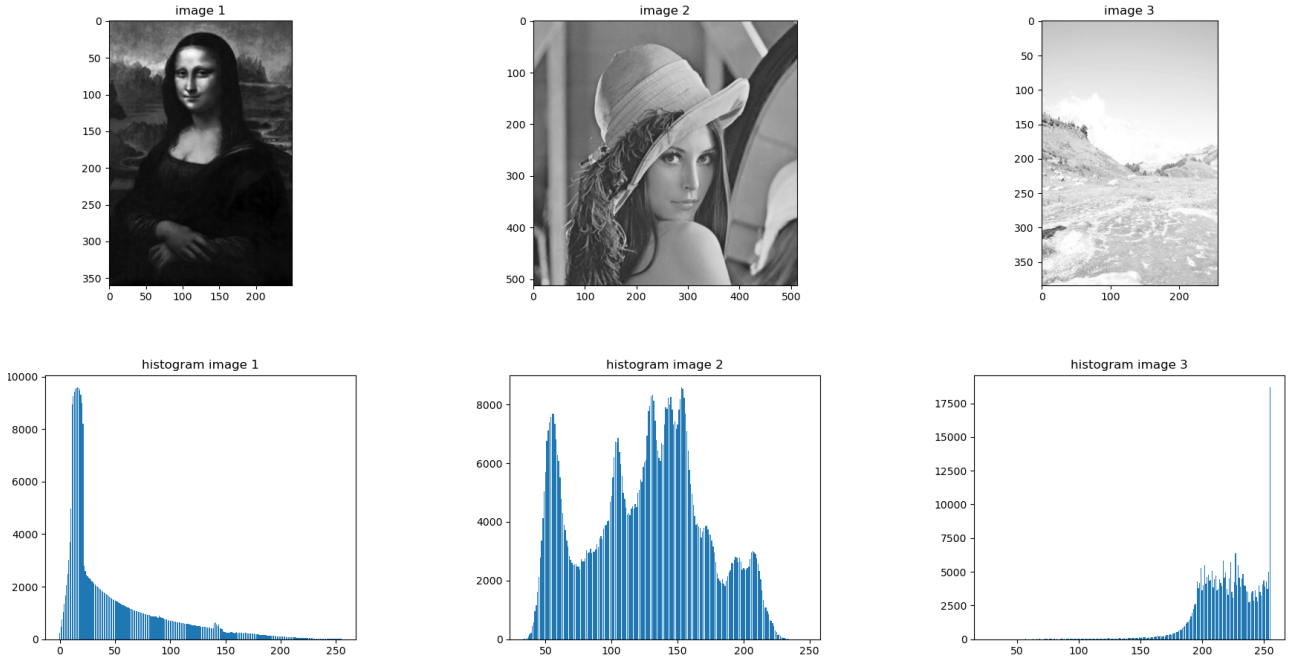
E-mail: Joao.Costa@etu.unige.ch

November 19, 2020

UNIVERSITÉ
DE GENÈVE
FACULTÉ DES SCIENCES
Département d'informatique

Exercise 1

(a)

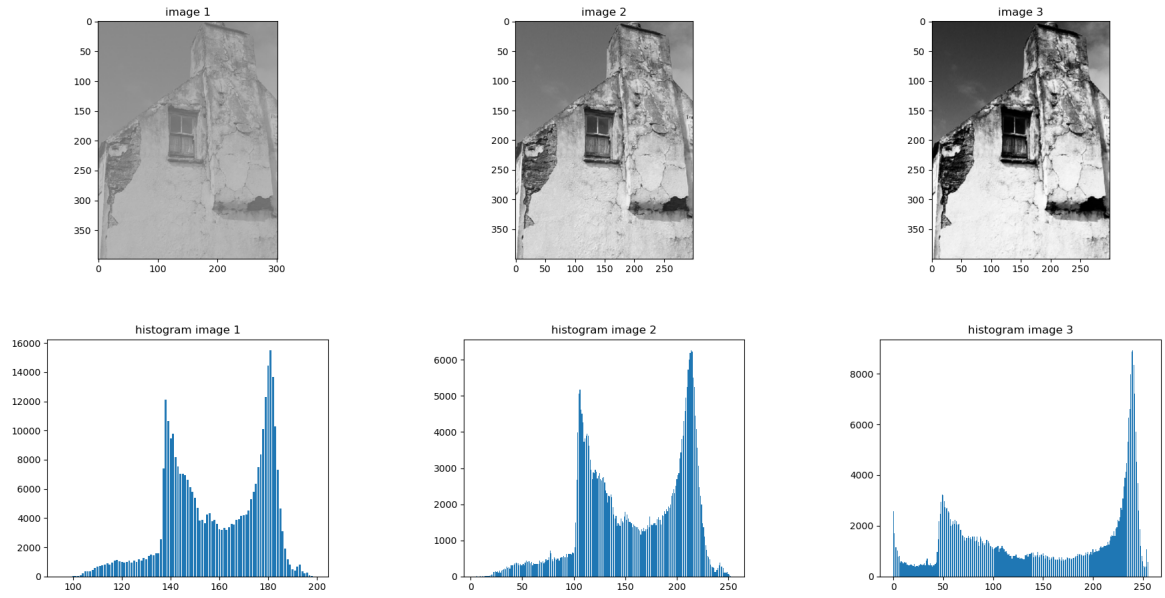


(b) Image 1 has a problem with under exposure, everything is very dark, and details are hard to notice. The opposite can be said for image 3, as we witness over exposure, we can hardly notice the mountain in the background. Image 2 has no exposure problems.

(c) To quickly see if an image has exposure problems, we can quickly see if the histogram is has a centered distribution, rather than a polarized one like in image 1 and 3.

Exercise 2

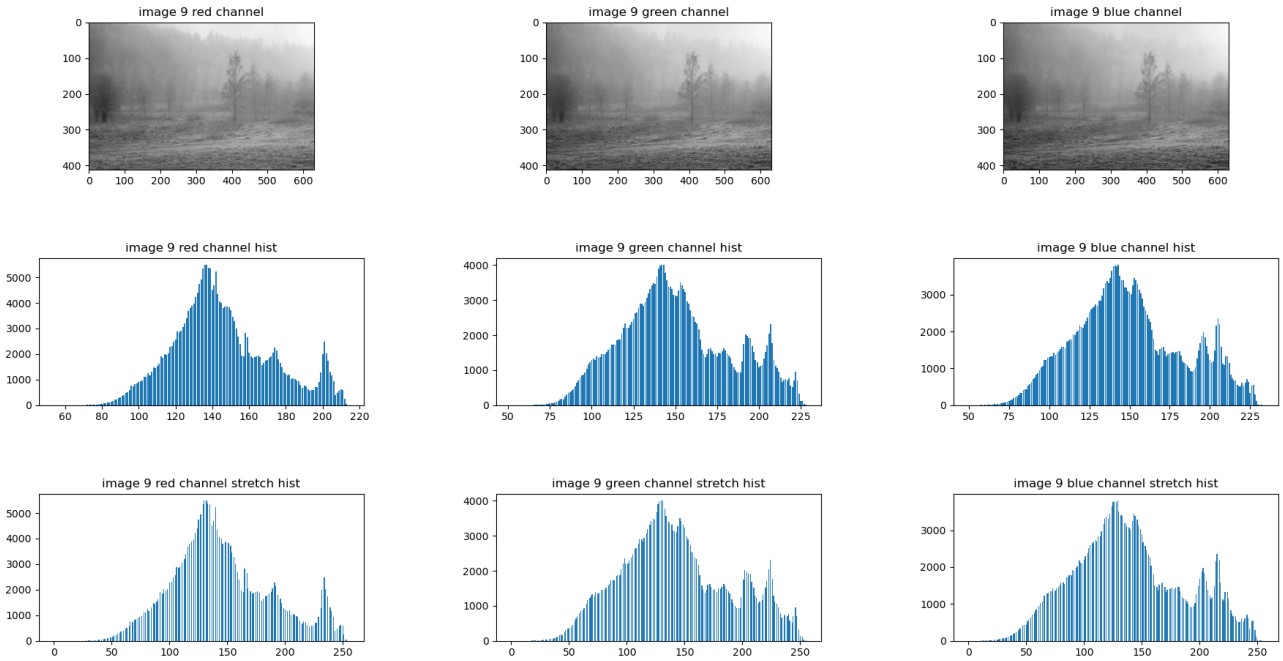
(a) Contrast problems can be noticed in the histogram by looking at H and L values, if they are far apart then there are no problems with contrast, if they are closer together then all the pixels have similar colors, and the image will lose quality.



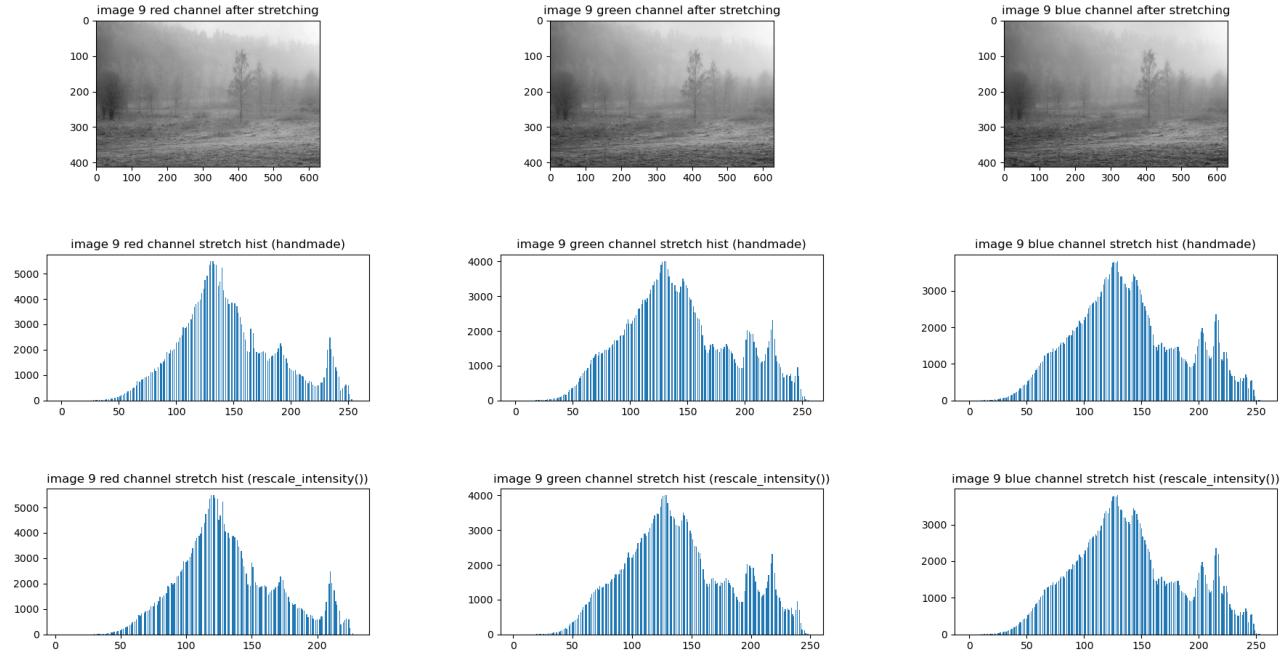
Exercise 3

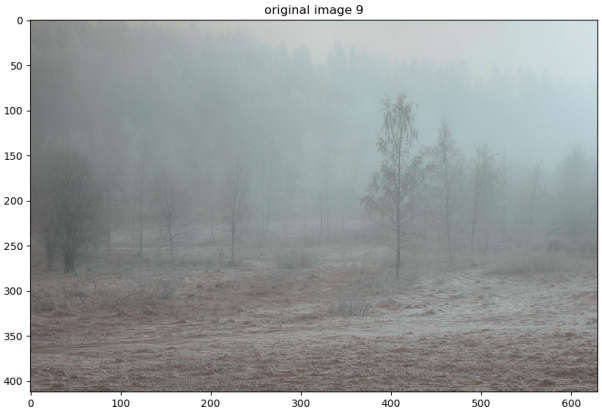
(a) This function can be found in the lib.py file, it's name is: contrast_ajustment_one_channel

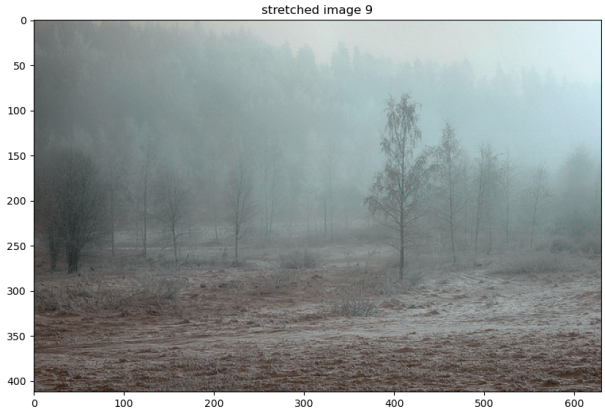
(b) In exercice_2, I explained what contrast problems are, and to fix them we can stretch the histogram to make sure all available shades of [black - white] array. Looking at the histograms, we clearly see that the H and L values change and are further apart.



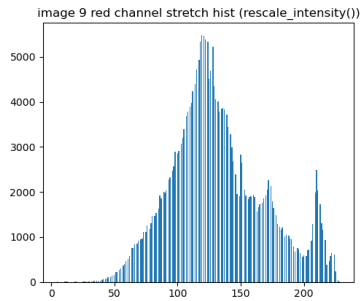
(c) We can see that both the function done in point (a) and the rescale_intesity() result in the same histograms. Moreover, to compare the original to the stretched one, I ploted them side by side, and in the stretched one we can clearly notice it has more details, especially in the foreground, the dirt is much higher quality. (see images below)

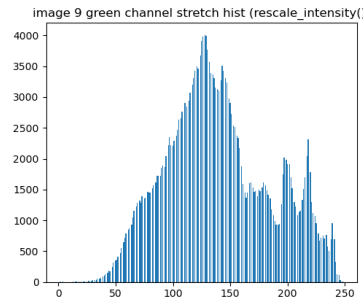


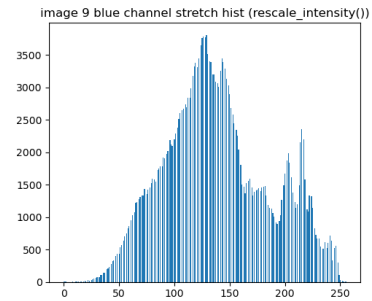


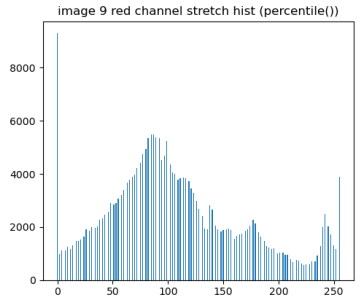


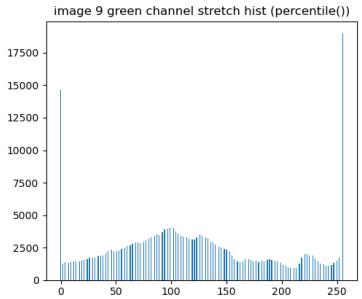
(d) Finally using the percentile function we see that the histograms have many pixels that are 0 and 255 that weren't there before. Comparing the images we see that the dirt in the foreground is even better quality, however the first tree is too dark, we would probably need to adjust the parameters.

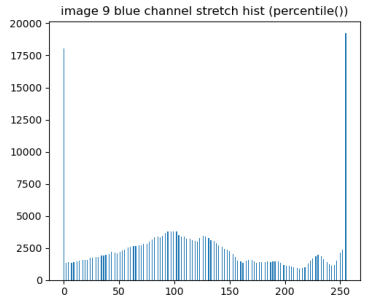


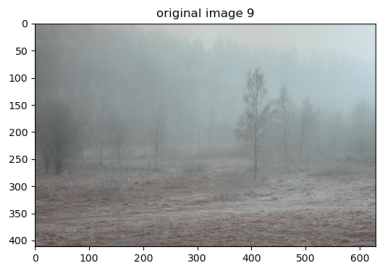


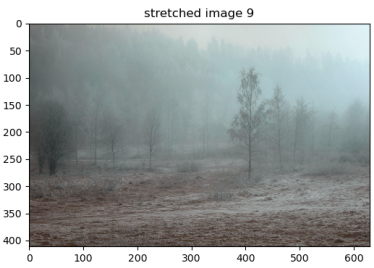


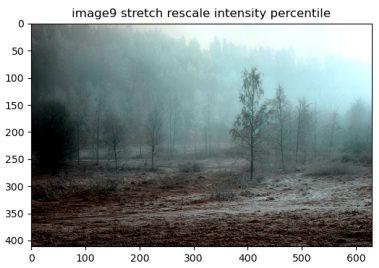








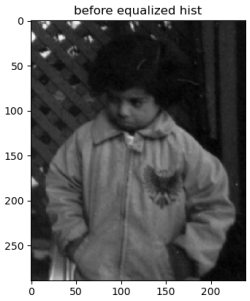


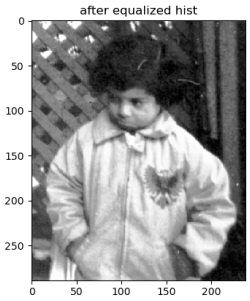


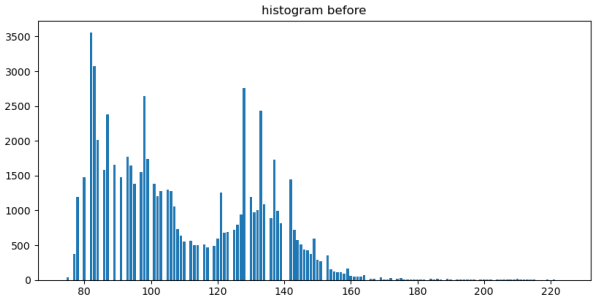
Exercise 4

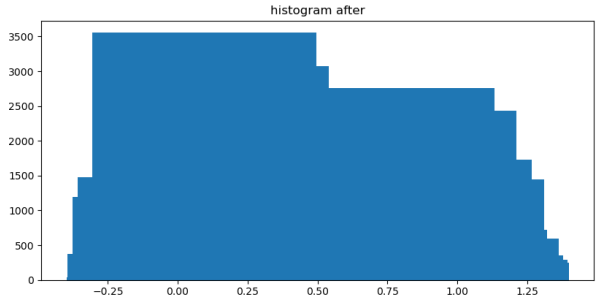
(a) The equalize_histogram function is another technique to enhance contrast problems, it tries to make intensities more equal all over the image, by doing this the result histogram follows an uniform distribution,

rather than a normal one that is usually witnessed in a normal image. In the result image we clearly see better, however, the boy's jacked is all of a sudden white, before it was grayish, is this it's true color ?



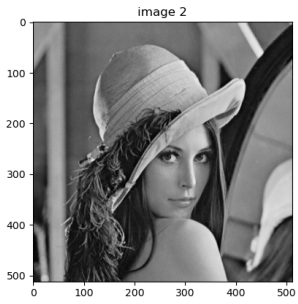


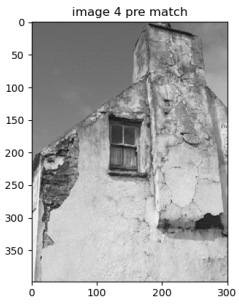


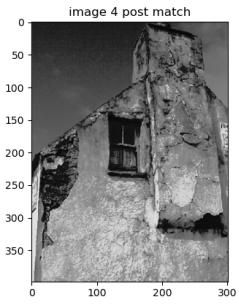


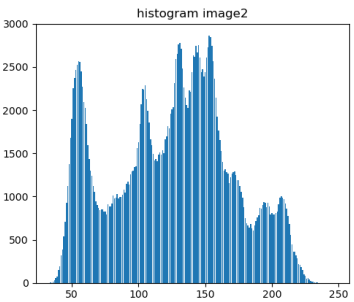
Exercise 5

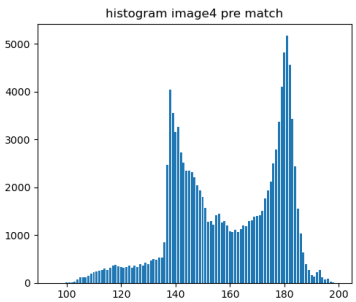
(a) This function receives two images has parameter, it creates an new one that has an histogram that looks more like the one given has reference (histogram of the second parameter). This can be used to fix contrast problems, if we have an image like image 2 that has no contrast problems, we can turn an image that has H and L values close to each other (image 4) and fix them to make them further apart. This is similar to what we would get with the fucking stretch, but the shades are not evenly separated.

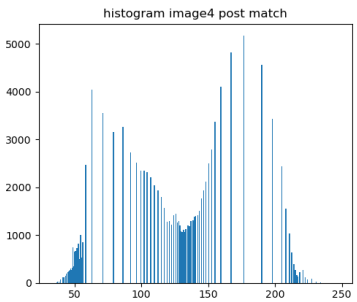








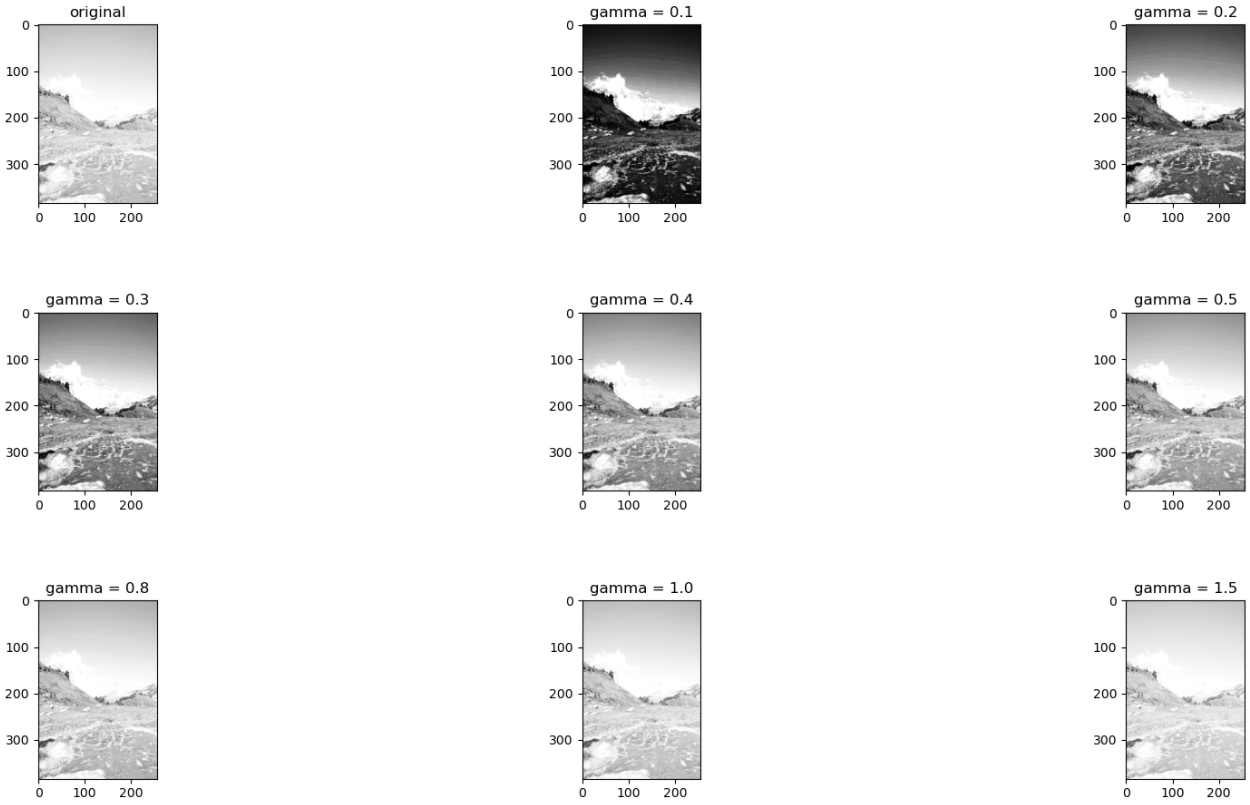




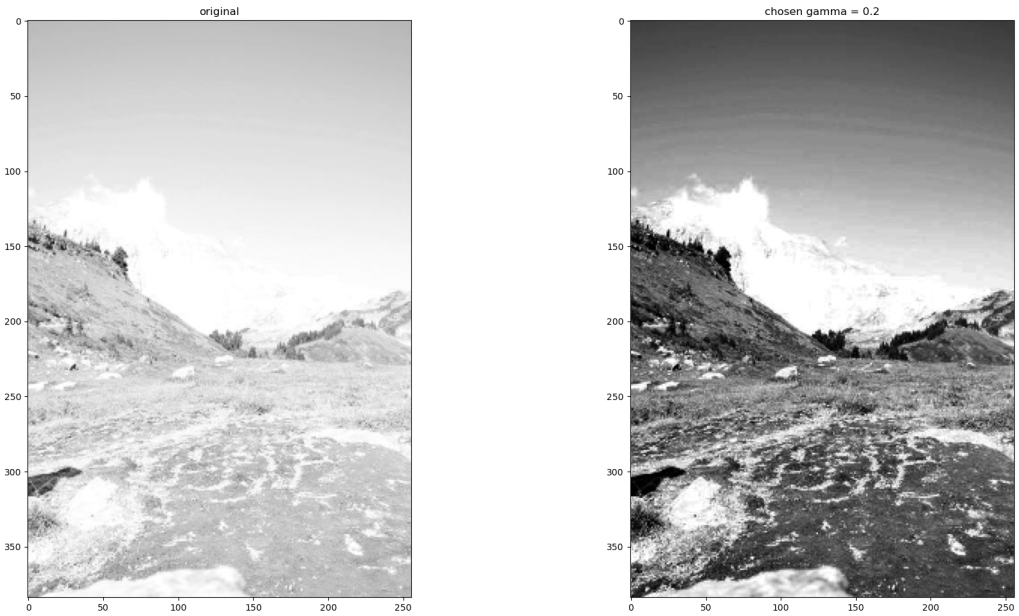
Exercise 6

(a) This function can be found in the lib.py file, it's name is: gamma_correction

(b) Here we can see gamma correction with different gamma values.

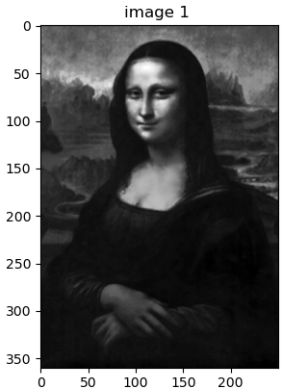


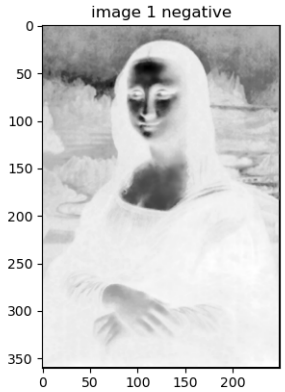
(c) I would argue that the best looking image is the one with gamma correction = to 0.2, this is much better than the original image, as the contrast between the ski and the background mountain is much clear (we can barely see it in the original image)

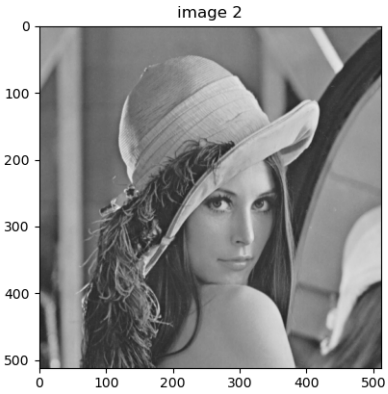


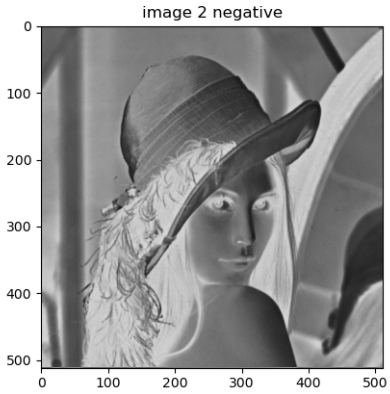
Exercise 7

(a) This transformation can be useful with images that have a darker background as new details might appear in the digital negative transformation.



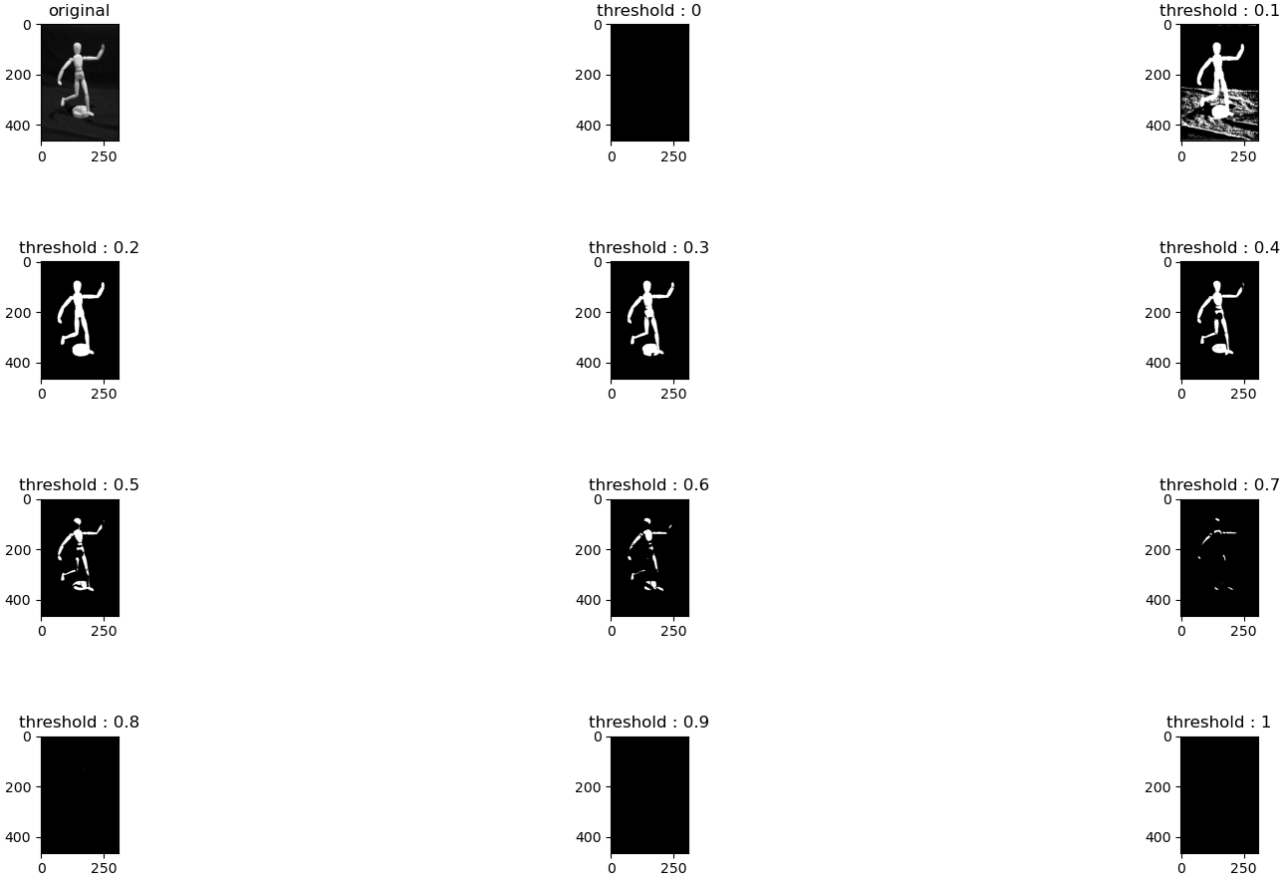




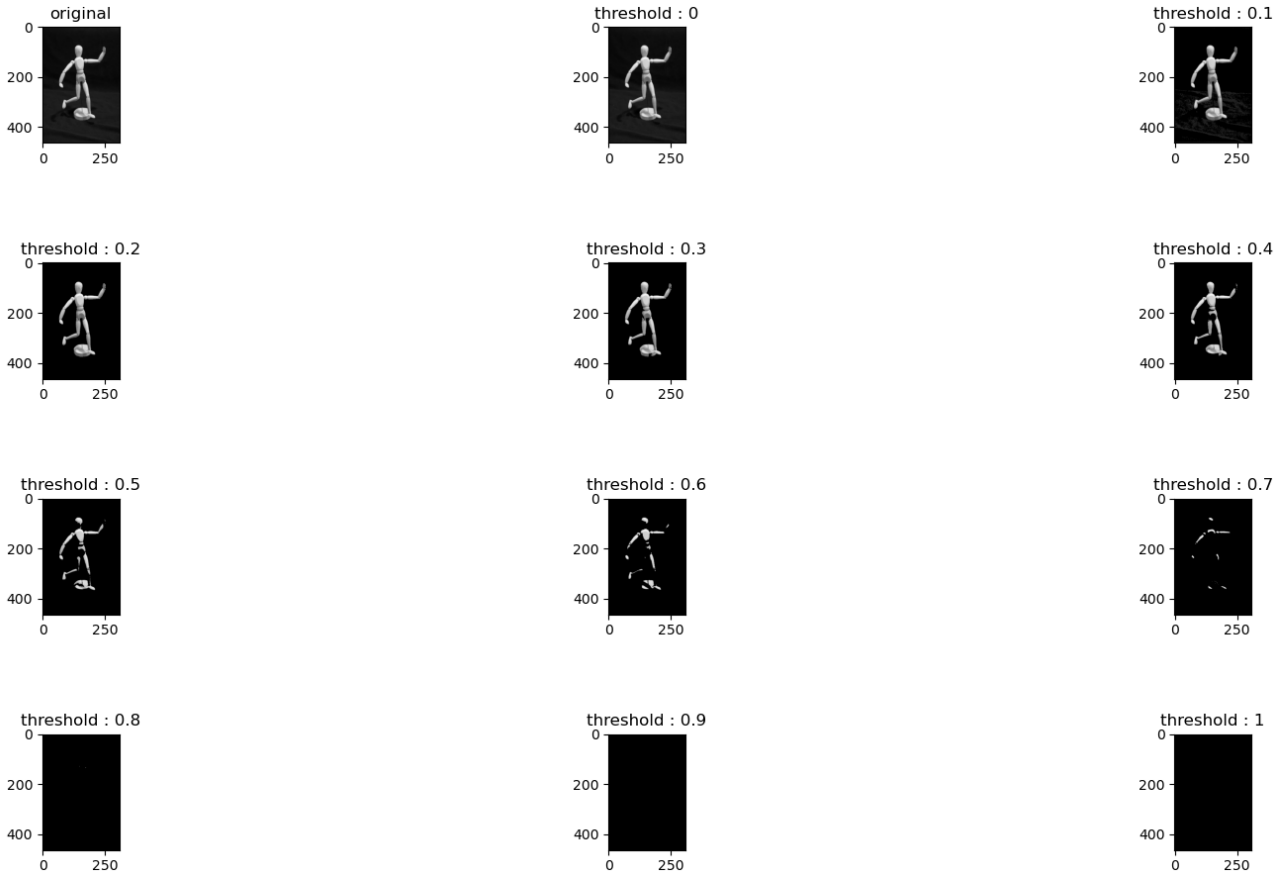


Exercise 8

(a) I don't know why the image for threshold = 0 is black, but it should be white, as every value in the array is equal to 1. since every pixel in the image is bigger or equal to 0.



(b) The multiplication gives us expected results, as the array of threshold = 0 is full of ones, then the multiplication pixel-wise will result in the original image.



(c) The best threshold depends on what we want to see, if the goal is to see nothing but the stick man in the middle without any shadows or anything else, then I'd say threshold = 0.2.