Report Labwork 10

Heiko VIEL

November 4, 2023

1 Report

For the project, I first used the function that we had coded for converting an image from RGB to HSV. Then I made the first function which uses the CPU, this method is based on the fact that we go through all the pixels of the image 1 by 1 with 2 For loop (you have to be careful not to exceed the image , hence starting at "half_window" and ending at height (width) - half_window), then as there are 4 windows, we do a loop again to go through them and get their standard deviation. We find the minimum standard deviations and we apply the change in RGB values to the pixels of the original image.

For the GPU part, I had a lot of problems because I couldn't manipulate the Arrays with Cuda. In addition, numpy does not work so we had to proceed differently. First of all we recover the pixels of the image with "tidy" and "tidx". We first check that they are indeed in the interval to avoid leaving the image, otherwise we assign the value 0. Then to calculate the standard deviation of each window, I went through the values twice. pixels to first calculate the mean and then calculate the variance to end up with the standard deviation. Then I found the minimum of the standard deviations and iterated through the pixels again to calculate the new RGB values.

The speed up between CPU and without-shared memory is huge. For the CPU, it took 110 seconds instead of less than 1 second for GPU.

I think my version with GPU could be better, I don't know if it's possible to remove the For loop but I didn't find how. Maybe it is also possible that my CPU programm is not working very well because the result is not the same as GPU.