Parallel and Distributed Systems

Marios Pakas Kosmas Meta

[mariospakas@ece.auth.gr](mailto:mariospakas@ece.auth.gr) -@ece.auth.gr

9498 9390

https://github.com/Mavioux/Parallel-And-Distributed-Systems-Exercise-3

# Exercise 2

## Helpful functions

In order to manipulate images in c, a library found on github was used that enabled image loading and storing in 1D arrays. For the addition of white gaussian noise to the original image, the box-muller algorithm was used. There were two awgn\_generator functions created, one was our own implementation, and the second one was a function found on a site[2]. Since the second function seemed to provide better noisy images, we proceeded with that implementation.

## NonLocal Image Algorithm

### Serial

The serial implementation of the algorithm looped through each individual image pixel and calculated its final value. To achieve that, it created an array(pixel\_patch) with a size of patchsize2, that would be then compared with the comparison\_patch array of each individual pixel of the image. For that a second loop was used to go through each pixel of the image, and the comparison\_patch was stored and then subtracted from the pixel\_patch. This, divided by sigma2,given a negative sign and then passed as a power of e would be multiplied with the value of the comparison pixel and added to the value of the original pixel. All those weights would be added to a variable zeta, with which we will divide the value of the original pixel, so as to make sure that in the end the value is between 0 and 1. When the initial loop is finished (O(n4)) the image produced would have been smoothed out and the gaussian noise should not be visible anymore.

### Cuda

So as to speed up the calculation of the value of each pixel, parallel programming with the help of a vector architecture, such as Cuda, was used. The general idea here was to call a width\*height kernel and assign each pixel to a different gpu thread. This enabled us to calculate the final value the same way as serially, but simultaneously for each pixel. This was the complexity of the code was reduced down to O(n2) with dramatic differences to the time it took to finish the job, especially as the size of the image or the patchsize increases.

Sources:

[1] Image Manipulation Library: <https://github.com/nothings/stb/>

[2] AWGN: <https://www.embeddedrelated.com/showcode/311.php>

Extra PDF on NonLocal Image Algorithm: <https://hal.archives-ouvertes.fr/hal-00271147/PDF/ijcvrevised.pdf>