# Implementation of Non Local Means Algorithm

Dinesh Subburaaj, Eram Arfa, Priyashi Yadav

Institut for Parallele und Verteile Systeme, IPVS, University of Stuttgart

**Abstract.** Non Local Means is a commonly used algorithm for image processing. Local Means filter calculates the mean of the group of pixels surrounding the target pixel to smoothen the image whereas Non Local Means finds the mean of all the pixels in the image weighted by how similar are these to the target pixel. While there are various techniques employed for image processing, Non Local Means in particular has greater post-filtering clarity and less loss of details compared to other algorithms available. In this report, we would like to introduce the concept of Non Local Means. Further we would discuss the implementation of the algorithm on CPU and GPU. With the sole aim to improve the time complexity, the GPU is used, hence, we would compare the performances between the two implementations and share our results.

## 1 Introduction

In the area of image processing, image denoising is one of the major problems. To main objective of image denoising is to remove interference due to noise that occurs during transmission recording etc. Buades proposed Non Local Means denoising algorithm in 2005. This method uses the non-local pixel for denoising in the image in contrast to the conventional local neighbourhood scheme which is employed in most algorithms for image processing.
In Local Neighbourhood filtering technique, a noisy pixel is replaced by the average value of the surrounding pixels. Non Local Means, henceforth addressed as NLM, uses the weighted average of all the pixels in the image to replace the noisy pixel. NLM has a self-similarity property due to which similar pixels contribute to the denoising of the noise pixel and the results are therefore better. NLM adds method noise which is similar to white noise. It is desirable as it is less disturbing in the denoised product.
Recently, NLM has been extended to other image processing algorithms such as deinterlacing, view interpolation, and depth maps regularizations. However, the computational complexity of NLM is quadratic in the number of pixels of the image, making it expesive to apply directly.
A popular way to reduce the time complexity of an algorithm is parallelization. We could parallelize an algorithm using multiple CPUs (cores) or GPUs.

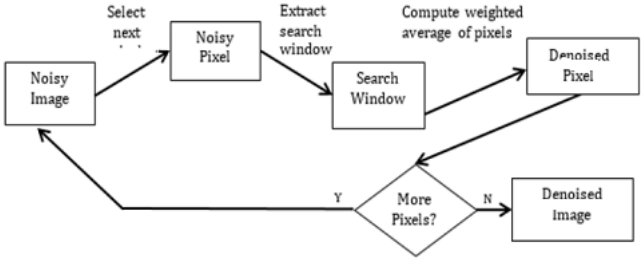


**Fig. 1.** Block Diagram of NLM Filtering Algorithm [3]

## 2 Definition of Non Local Means

Suppose the area of the image is $\Omega$ and p and q are two points within the image. Then the algorithm is as follows:

$$u(p) = \frac{1}{C(p)} \int_{\Omega} v(q) f(p,q)\, dx \tag{1}$$

where u(p) is the filtered value of the image at point p, v(q) is the unfiltered value of the image at point q and f(p,q) is the weighting function, and the integral is evaluated for all q $\epsilon$ $\Omega$.

$$C(p) = \int_{\Omega} f(p,q)\, dq \tag{2}$$

The principle of NLM is as shown in the figure 2. The figure has three pixels denoted as points P, Y1, Y2 [1] and their neighbourhoods. Its can be clearly observed that the pixels P and Y1 have similar neighbourhoods whereas P and Y2 don't. The similarity weights are calculated and it is found that $\omega$(P,Y2) of Y2 is larger than $\omega$(P, Y1) of Y1 while filtering. The final result is computed by finding all the point pixels in the image that have a similarity with P and then calculate the weights of the similarities obtained.

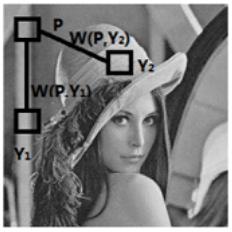


**Fig. 2.** Principle of NLM Filtering Algorithm [1]

## 3 Project Planning

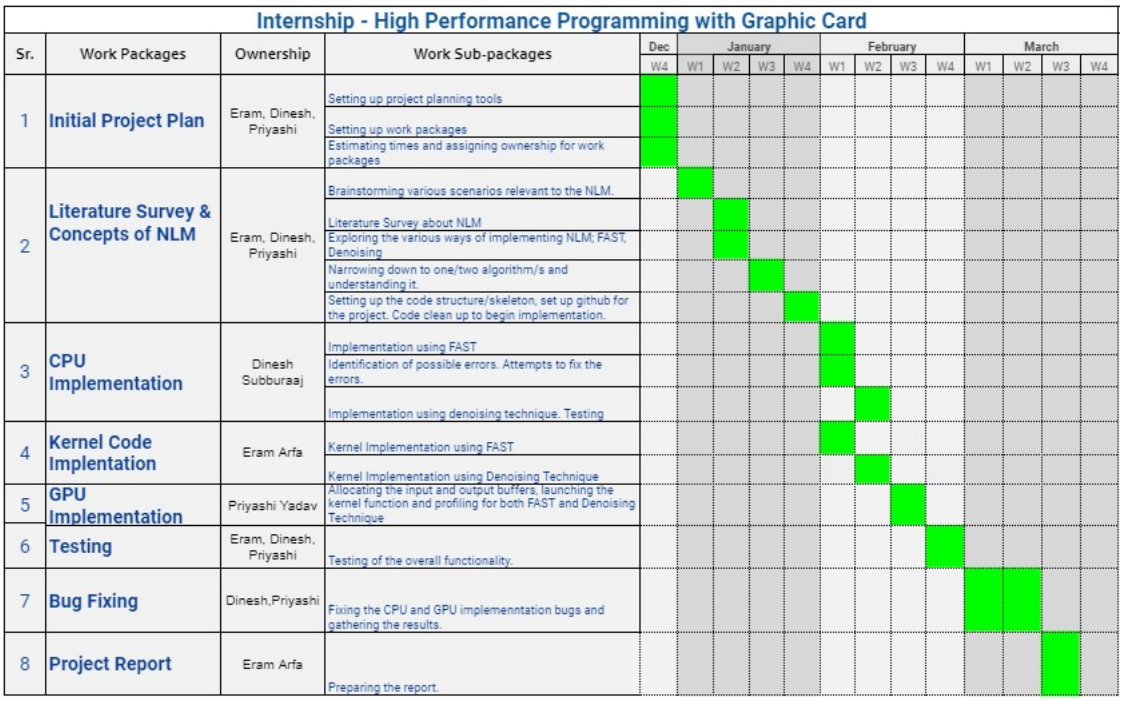


**Fig. 3.** Planning and Effort Estimation

## 4 CPU vs GPU

CPU and GPU are microprocessors and both are silicon-based. CPU consists of less cores compared to GPU and handle sequential signal processing while GPUs have a paralled architecture and allow multiple functions to run at the same time.The hundred plus cores of GPU supports execution of way faster than CPU at a cheaper rate. GPU executes same operation on work items such as pixels in an image and we can think of it as a Single Instruction Multiple Data processor array which supports application having parallel data.
GPUs are high latency tolerant and provide a higher throughput compared to CPUs. While general purpose computing is still performed by the CPU, GPUs on the other hand are more power efficient. It is therefore suited for implementing NLM.

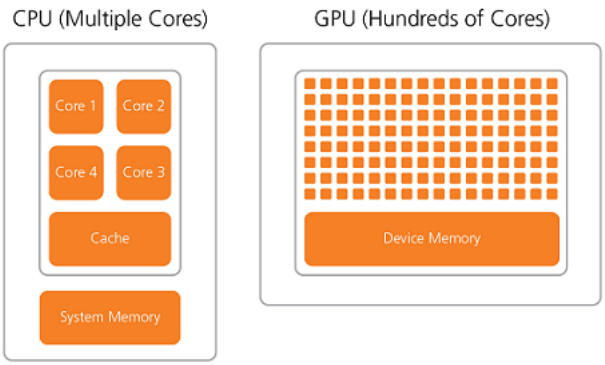


**Fig. 4.** CPU vs GPU [2]

## 5 Implementation

### 5.1 Implementation on CPU

The implementation of the algorithm on the CPU is done in C++. The input and output buffers are defined for the CPU where the noisy image and the resultant denoised image will be stored. The width and height of the image and also the search window is taken as input.Then the Euclidean distance is calculated within the similar window and sum of all the distances is computed. Thereafter the weight matrix calculation is done using the formula

$$w = e^{\frac{-v}{h^2}} \tag{3}$$

The pixels are then multiplied with the weight matrix and the sum is computed within the search window. The denoising result for each pixel is then computed for the whole image and stored in the output buffer.The time taken for the computation on the CPU is profiled for performance comparison.

### 5.2 Implementation on GPU

OpenCL is a cross-platform programming language for parallel computing programming on GPU and other microprocessors. OpenCL provides an independent platform for parallel computing. It creates efficient programming interfaces. Work-group is a collection of related work items that execute on a single computing unit, the work-items within the work-group are executed on the same thread. When launching the kernel, the host code specifies the work size. The following steps are performed to implement the algorithm on GPU:

- Define the number of work items in the work group both in the X and Y direction.
- Then we calculate the total number of elements in the X and Y direction to get the overall size of data in bytes. The height and width of image which is going to be provided as input data to the GPU function is also defined in the host code.
- The input and output buffers are allocated on the device based on the size of the work groups calculated above.
- The memory is initialized to the starting point. It is useful for debugging because otherwise GPU memory will contain information from last execution.
- The read and write buffers are defined and then the kernel object is created and finally the kernel is launched.
- The kernel takes the noisy image, size of the image as input. Then the Euclidean distance is calculated within the similar window and sum of all the distances is computed.
- Then the weight of each pixel is calculated within the search window.
- The denoising result for each pixel is then computed in the whole image.
- The time taken for the computation on the GPU is profiled for performance comparison.

## 6 Results

### 6.1 Output GPU

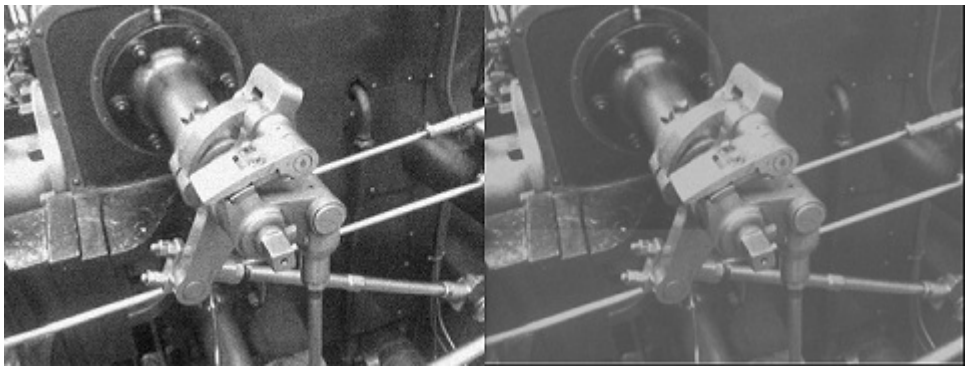

**Fig. 5.** Noisy Image (left) GPU Result (right)

### 6.2 Output CPU

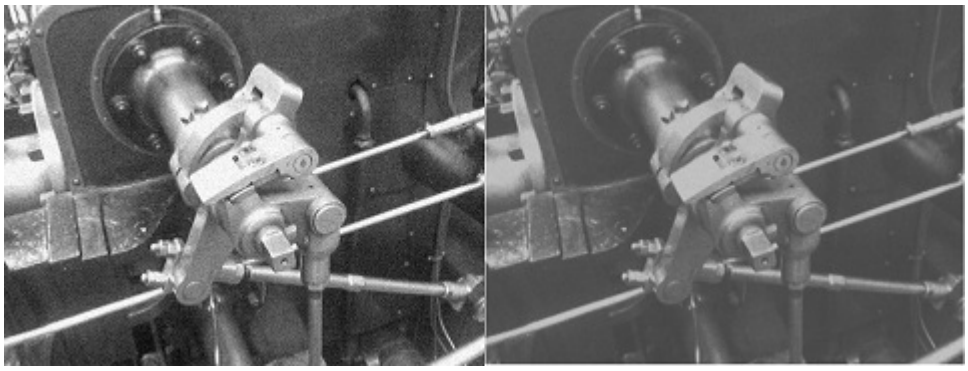

**Fig. 6.** Noisy Image (left) CPU Result (right)

## 7 Performance Evaluation

The performance evaluation of the algorithm on the CPU and GPU is done on the basis of metric execution time.The execution times on both devices is computed and the speedup is measured. The following figure is the output of the algorithm shows the computation times of running the code on GPU and CPU. While profiling the time taken on both GPU and CPU are captured and the speed up is calculated.
Ideally, the GPU should be faster than the CPU but in the results from our implementation it is visible that CPU time is less than GPU time. We have tried various approaches like multiple kernel launches and local memory but somehow it didn't fit for NLM algorithm. Hence the performance couldn't be improved

```
Using platform 'NVIDIA CUDA' from 'NVIDIA Corporation'
Working
Using device 1 / 1
Running on GeForce GT 610 (2.1)
inputWidth:: 640inputHeight:: 480Starting to run nlm algorithm on cpu
Debug:: I VALUE: 0
Debug:: I VALUE: 1
Debug:: I VALUE: 2
Debug:: I VALUE: 3
Debug:: I VALUE: 4
Debug:: I VALUE: 5
Debug:: I VALUE: 6
.....
.....
.....
Debug:: I VALUE: 477
Finishing host calculation
Image written
Creating kernel object

GPU Time: 2032.451837s
CPU Time: 1638.424021s
Speedup: 0.806132
Creating Output Image GPU
```

**Fig. 7.** Performance Comparison

## References

1. Overview of image noise reduction based on non-local mean algorithm Baozhong LIU1,2,a and Jianbin LIU1,2 1Computer School, Beijing Information Science Technology University ,Beijing 100101 ,China 2Software Engineering Research Center, Beijing Information Science Technology University ,Beijing 100101 ,China
2. https://www.a2zgyaan.com/cpu-vs-gpu/
3. CUDA IMPLEMENTATION OF NON-LOCAL MEANS ALGORITHM FOR GPU PROCESSORS Farha Fatina Wahid Department of Information Technology, Kannur University, Kerala, India farhawahid@gmail.com Sugandhi K Department of Information Technology, Kannur University, Kerala, India sugandhikgs@gmail.com Raju G Department of Computer Science and Engineering, Christ (Deemed to be University), Bengaluru, India kurupgraju@gmail.com