

# Implementation of Non Local Means Algorithm

Dinesh Subburaj, 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. It uses all the pixels' mean value in the image weighted by how similar these are to the target pixel to smoothen the image. While there are various techniques employed for image processing, Non-Local Means has greater post-filtering clarity and less loss of details than other algorithms available. In this work, we are using the Non-Local Means algorithm to denoise the image. The algorithm is implemented on both CPU and GPU. With the sole aim to improve the time complexity, the GPU is used. Hence, we will compare the performances between the two implementations and share our results.

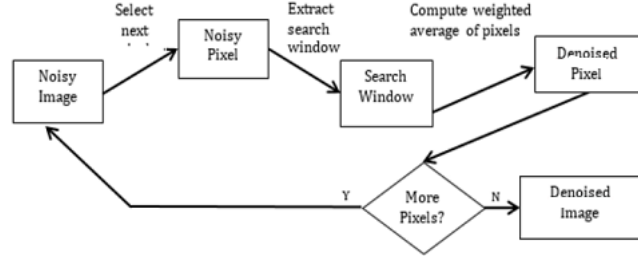
## 1 Introduction

In the area of image processing, image denoising is one of the major challenges. The main objective of image denoising is to remove interference due to noise that occurs during transmission recording etc. Buades proposed a Non-Local Means denoising algorithm in 2005. This method uses the non-local pixel for denoising the image in contrast to the conventional local neighbourhood scheme, which is employed in most algorithms for image processing.

In the 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 expensive 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:

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

where  $u(p)$  is image's filtered value at location  $p$ ,  $v(q)$  is image's unfiltered value at location  $q$  and  $f(p, q)$  is the weighting function, and the integral is evaluated for all  $q \in \Omega$ .

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

The principle of NLM is, as shown in figure 2. The figure has three pixels denoted as points  $P$ ,  $Y_1$ ,  $Y_2$  [1] and their neighbourhoods. It can be clearly observed that the pixels  $P$  and  $Y_1$  have similar neighbourhoods, whereas  $P$  and  $Y_2$  don't. The similarity weights are calculated, and it is found that  $\omega(P, Y_2)$  of  $Y_2$  is larger than  $\omega(P, Y_1)$  of  $Y_1$  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

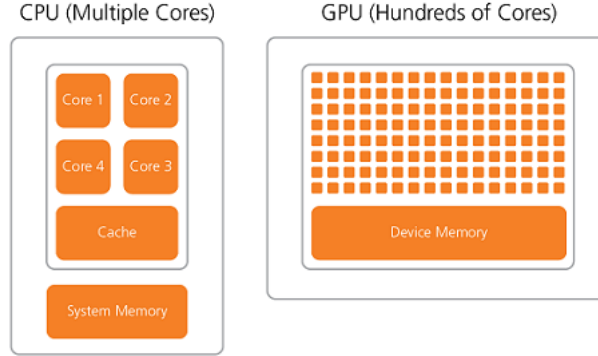
Internship - High Performance Programming with Graphic Card																
Sr.	Work Packages	Ownership	Work Sub-packages	Dec	January				February				March			
				W4	W1	W2	W3	W4	W1	W2	W3	W4	W1	W2	W3	W4
1	Initial Project Plan	Eram, Dinesh, Priyashi	Setting up project planning tools													
			Setting up work packages													
			Estimating times and assigning ownership for work packages													
2	Literature Survey & Concepts of NLM	Eram, Dinesh, Priyashi	Brainstorming various scenarios relevant to the NLM.													
			Literature Survey about NLM													
			Exploring the various ways of implementing NLM, FAST, Denoising													
			Narrowing down to one/two algorithm/s and understanding it													
3	CPU Implementation	Dinesh Subburaaj	Setting up the code structure/skeleton, set up github for the project. Code clean up to begin implementation.													
			Implementation using FAST													
			Identification of possible errors. Attempts to fix the errors.													
4	Kernel Code Implementation	Eram Arfa	Implementation using denoising technique. Testing													
			Kernel Implementation using FAST													
5	GPU Implementation	Priyashi Yadav	Kernel Implementation using Denoising Technique													
			Allocating the input and output buffers, launching the kernel function and profiling for both FAST and Denoising Technique													
6	Testing	Eram, Dinesh, Priyashi	Testing of the overall functionality.													
7	Bug Fixing	Dinesh, Priyashi	Fixing the CPU and GPU implementation bugs and gathering the results.													
8	Project Report	Eram Arfa	Preparing the report.													

Fig. 3. Planning and Effort Estimation

### 4 CPU vs GPU

CPU and GPU are microprocessors, and both are silicon-based. CPU consists of fewer cores compared to GPU and handle sequential signal processing, while GPUs have a parallel architecture and allow multiple functions to run at the same time. The hundred-plus cores of GPU support execution way faster than CPU at a cheaper rate. GPU executes the same operation on work items such as pixels in an image, and we can think of it as a Single Instruction Multiple Data processor arrays which support 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

C++ is used to implement the Non-Local Means algorithm in the CPU. To store the noisy input image and the resultant denoised image, memory is allocated in the host. The input arguments to this algorithm are width, height and the search window. The Euclidean distance is calculated within a similar window, and the sum of all the distances is computed. After that, the weight matrix calculation is done using the formula

$$w = e^{\frac{-v}{h^2}} \quad (3)$$

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.

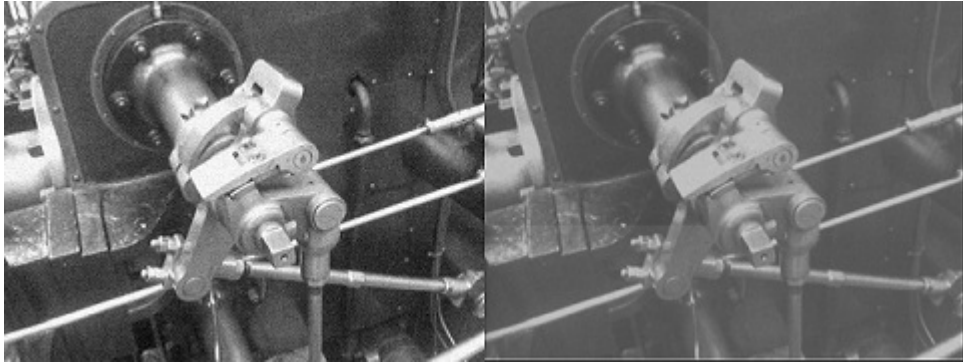
### 5.2 Implementation on GPU

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 workgroup will be 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 workgroup 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 the image, which will 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 workgroups calculated above.
- The memory is initialized to the starting point. It is useful for debugging because, otherwise, GPU memory will contain information from the 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. The Euclidean distance is calculated within a similar window, and the 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.

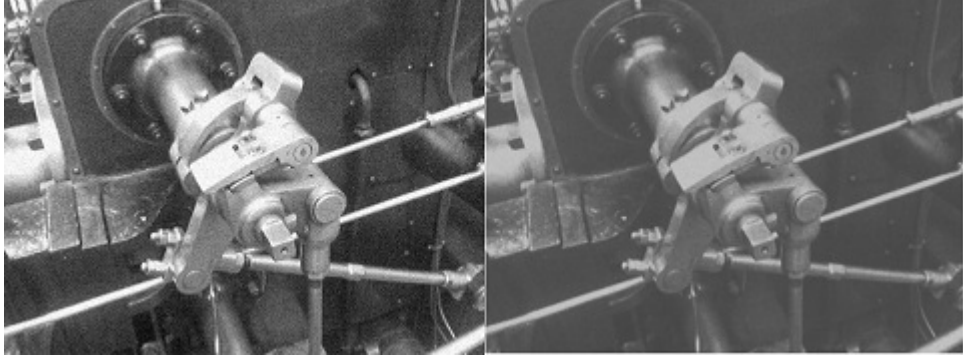
## 6 Results

### 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 based on metric execution time. The execution times on both devices is computed, and the speedup is measured. The following figure is the algorithm's output that shows the computation times of running the code on GPU and CPU. While profiling, the time taken on GPU and CPU is captured, and the speedup is calculated.

Ideally, the GPU should be faster than the CPU, but in our implementation results, it is visible that CPU time is less than GPU time. We have tried various approaches like multiple kernel launches and local memory to improve GPU performance, but it did not work out.

---

```
1 Using platform 'NVIDIA CUDA' from 'NVIDIA Corporation'
2 Working
3 Using device 1 / 1
4 Running on GeForce GT 610 (2.1)
5 inputWidth:: 640inputHeight:: 480Starting to run nlm algorithm on cpu
6 Debug:: I VALUE: 0
7 Debug:: I VALUE: 1
8 Debug:: I VALUE: 2
9 Debug:: I VALUE: 3
10 Debug:: I VALUE: 4
11 Debug:: I VALUE: 5
12 Debug:: I VALUE: 6
13 .....
14 .....
15 .....
16 Debug:: I VALUE: 477
17 Finishing host calculation
18 Image written
19 Creating kernel object
20
21 GPU Time: 2032.451837s
22 CPU Time: 1638.424021s
23 Speedup: 0.806132
24 Creating Output Image GPU
```

---

**Fig. 7.** Performance Comparison

## References

1. Overview of image noise reduction based on non-local mean algorithm Baozhong LIU<sup>1,2,a</sup> and Jianbin LIU<sup>1,2</sup> <sup>1</sup>Computer School, Beijing Information Science Technology University ,Beijing 100101 ,China <sup>2</sup>Software 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