15618 - Parallel Computer Architecture and Programming

Project Milestone Report

Agastya Sampath (agastyas), Shijie Bian (sbian)

TITLE : PIPL - Parallel Image Processing Library

URL : https://agastya-sampath.github.io/618-project-s23/

Project Schedule:

Shown below is the original schedule as included in the project proposal, where the milestones are defined per week. Now that we gained a better understanding of the difficulty and workload, we propose a more refined schedule, which is shown in the table following this one.

Week	Deadline	Task
1	April 7	Set up the framework for library implementation
2	April 14	Implement one image processing algorithm (denoising), possibly two
3	April 21	Implement second image processing algorithm (color conversion)
4	April 28	Implement third image processing algorithm (color augmentation)
5	May 4	Final touches + Record performance numbers on all implementations. Stretch goal: Combine into a pipeline, or attempt multiple implementations across frameworks

Here is our revised and refined schedule (with group member assigned):
We would also like to clarify that this schedule is also tentative.

Deadline	Task	Responsible

April 23	Finish the profiling of performance of the 2 denoising algorithms (median filtering and non-local mean), with OpenMP and possibly CUDA implementations	Agastya (profiling the performance and experimenting with OpenMP configurations), Brandon (debugging the CUDA implementation)
April 26	Finish the serial and OpenMP implementation of the color conversion algorithms, profile their performance. If we have time, implement the CUDA version of paralleled execution.	Agastya and Brandon (work together on this task)
April 29	Finish the serial and OpenMP implementation of the color augmentation algorithms, and profile their performance. If we have time, implement the CUDA version of this algorithm and the one before.	Agastya and Brandon (work together on this task)
May 2	Organize the code and makefiles so that we can easily specify the implementation method, the algorithm, the hyperparameters, the input output channels, and the profiling of performance. Produce images and illustrations of performance profiling, demonstrating how the parallelism helped with the task, and why.	Agastya (profiling performance of the 3 functionalities and corresponding algorithms and parallelism methods, create illustrations), Brandon (organize code and makefiles, make sure the specified channels and building works to create a usable library)
May 4	Final touches, and time to finish essentials parts that we have not yet completed from previous milestones and timestamps. Create documentation and poster for presentation. If possible, consider	Agastya and Brandon (work together on this task)

	stretch goals.	

Work you have done so far:

We have already successfully implemented two algorithms for image denoising: one using median filtering and the other using non-local mean. For both algorithms we have implemented serialized versions as well as OpenMP versions. We have finished preliminary profiling of both algorithms and under both serialized and OpenMP parallelism, and noticed that the OpenMP parallelism indeed increased the performance significantly. Furthermore, we have also utilized different denoising sample images of different sizes and noise-inclusion difficutilies to help us profile our performance. We are also in the process of finishing the CUDA implementation of the median filtering algorithm, which, after finishing, will complete our major milestone of finishing the image denoising algorithm and performance profiling.

Describe how you are doing with respect to the goals and deliverables stated in your proposal. Do you still believe you will be able to produce all your deliverables? If not, why? What about the ”nice to haves”? In your milestone writeup we want a new list of goals that you plan to hit for the poster session.

We think we are on a pretty good track in terms of the goals and progress, and are fairly confident that we will be able to finish the project on time with all milestones as listed above completed. We spent some time in implementing the serialized algorithm for the two approaches of dealing with image denoising. The median filtering algorithm is fairly easy to implement, but we found that it produced poor results, since it will reduce the resolution and blur the overall image quite significantly for images that have a lot of noise. This motivated us to develop the more sophisticated non-local mean filtering denoising. This new algorithm is much slower to run due to its highly complicated computation, and

deep nested for loops. However, we observed that this algorithm gave us more opportunities to boost the performance using OpenMP parallelism, which is valuable for our project. Furthermore, we met some difficulties in completing the CUDA implementation of the algorithms, but we are still working on this, since CUDA performance profiling is very essential to our overall goal of profiling how different parallelism approaches may affect the performance boosting of different image processing algorithms. For a more detailed list of goals, please see below (**note that this is more or less tentative**):

- Image denoising
    - Median filtering
        - Serialized
        - OpenMP
        - CUDA
    - Non-local Means Filtering
        - Serialized
        - OpenMP
- Color Conversion (1 algorithm minimum, 2 if we have sufficient time)
    - Serialized
    - OpenMP
    - CUDA
- Color Augmentation (1 algorithm)
    - Serialized
    - OpenMP
    - CUDA
- Performance profiling and visualizations, possibly live-demos

---

What do you plan to show at the poster session? Will it be a demo? Will it be a graph?

We plan to produce a comprehensive chart and visualization (similar to what we did in the report of the final two projects), demonstrating the performance

boosting of the three image processing tools of our library, with different implementation algorithms, and under different parallelism approaches and resources allocated. We hope that these documentation and visualizations can help illustrate how paralleled programming and thinking can help significantly boost the performance of commonly computationally expensive image processing algorithms. Furthermore, we also wish to show some demonstrations (such as timed GIFs, side-by-side comparison of how paralleled version of the same algorithm takes much less time for processing the same pictures).

Preliminary results:

Using CBSD68 dataset
(https://github.com/clausmichele/CBSD68-dataset/tree/master/CBSD68)

Input noisy image:

Median filtering:

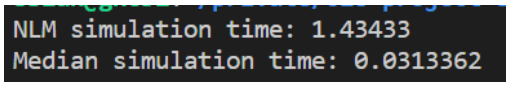
Non-local mean filtering:

From the comparison we can see that non-local mean is producing pretty good results with much less blurring:

However, non-local mean is taking much longer time to complete than median filtering:

```
NLM simulation time: 10.1228
Median simulation time: 0.241472
```

But, if we enable OpenMP, we see a significant boost in performance in both algorithms:



In this case, using NLM has become affordable with insignificant overhead, but much better quality and overall performance - indicating that we have successfully produced a denoising tool for our library that is both fast (using OpenMP) and of high quality (using a non-trivial and computation-extensive algorithm).