Assignment 3
Introduction to GPGPU with Nvidia Warp
Weight: 18% of the final grade
Due date: Friday, November 29, 11:59pm.

Assignment overview

In this assignment you will get a chance to play with real-world applications of GPGPU programming. You will need to implement two basic image-processing algorithms - one for sharpening and one for noise removal - using the Nvidia Warp API. Each algorithm should be implemented using a dedicated kernel.

You can complete this assignment individually or in teams of two.

Implementation

For sharpening, use unsharp marking described in class. For noise removal, you can pick one of the low-pass filtering algorithms described in Lecture 12.

Your executable will run as follows:
```
python3 a3.py algType kernSize param inFileName outFileName
```
where:
- `algType` is either `-s` (sharpen) or `-n` (noise removal)
- `kernSize` is the kernel size - e.g. `3` for 3x3, `5` for 5x5, etc.. It must always be positive and odd.
- `param` is the additional numerical parameter that the algorithm needs - e.g. the scaling value `k` for unsharp masking or sigma for the gaussian. If your algorithm doesn't need any additional parameters once it knows the kernel size, just pass some dummy value here (e.g. 0)
- `inFileName` is the name of the input image file
- `outFileName` is the name of the output image file

Note: do not use the simple averaging filter for noise removal - use one of the more sophisticated algorithms discussed in Lecture 12. The averaging filter is already part of the unsharp masking implementation, so you will not get any marks for it if you also use it for noise removal.

Your algorithm must work with both greyscale and colour images. You will use the PIL image library. A stub that shows you how to open an image using PIL and how to check its mode has been provided for you.

Report

In addition to the implementation, provide a report document in which you:
- Provide any information that we might need when running your code

- Describe which algorithms you have implemented
- Explain how you have implemented the kernel for this algorithm using the Warp API. You can paste the code into the document so you can more clearly explain it.
- Make sure your explanation includes a description of how you handle the borders. Note that you cannot leave them unprocessed. You must implement an approach that computes new values for the entire image, including the border - i.e. Option 2 or 3 from Lecture 12

We will assume that you develop and test your GPGPU code on Docker, where it will run on a single CPU thread. As a result, you will not see any performance improvements, unless you happen to own a machine with an Nvidia GPU.

Therefore the A3 report does not need to include any timings or performance analysis (unless you do the bonus - see below). If you have access to Nvidia hardware, you are welcome to run our algorithm with `device="cuda"` and discuss algorithm performance (e.g.CPU vs GPU execution). However, since we do not have a CUDA-capable GPU platform available to everyone in the class, this is definitely not required, and it will not affect your grade.

Bonus mark (up to 10 marks)

1. Alternative sharpening: implement a sharpening algorithm other than unsharp masking for sharpening. This algorithm cannot be based on anything described in class (**5 marks**)
2. Alternative denoising: implement a noise removal algorithm that is not based on what we have discussed in class (**5 marks**)

To get each of these bonus marks, you must do the following:
- Clearly describe the algorithm in your report what the algorithm and state where you found its description
- Clearly demonstrate, with visual examples, that the algorithm you have implemented produced better results than the algorithm it replaces. Specifically:
  - If you are implementing a denoising algorithm, demonstrate with sample images that it does a better job or removing noise while retaining detail than the algorithms described in class
  - If you are implementing a sharpening algorithm, demonstrate that it does a better job of enhancing details in an image than unsharp masking

Both bonus components are meant for those of you who wish to take a deeper dive into image processing. They also give you a chance to think about "algorithm performance" in a different context - not how fast it runs, but how well it does what it is supposed to do. As a result, the bonus will take some work.

To do the required performance analysis ("bonus options works better than the class option"), you will have to implement both the default algorithm and the bonus one. You will also have to find and process images that demonstrate the performance of the algorithm, and conduct an analysis explaining why you think the alternative algorithm produces better results.

Please do not attempt the bonus unless you have already fully implemented the assignment - see thebgmrade breakdown for details.

Grace period

The grace period for this algorithm is 48 hours. There will be no penalties applied during the grace period. No submissions will be accepted after the end of the grace period - the dropbox will close at **11:59pm on Sunday, December 1**.

Grading

The assignment will be marked using the standard CIS*3090 Docker image provided on the course website and discussed in class. We will download and run a fresh image, create a container, and use Docker to run and grade your code. Make sure you test your code accordingly.

Please note that, since this assignment evaluates your knowledge of GPGPU and stream programming, **no marks at all** will be given for a solution that is not parallelized using the Warp API.

Also, no partial marks will be given for each of the two bonus components. Moreover, the bonus marks will be awarded only if the required functionality received the minimum grade of **90/100**.

A3 Grade breakdown:

- Implements the image processing algorithms: **70 marks**
  - Noise removal: **35 marks**
  - Sharpening: **35 marks**
- Report with all the analysis: **30 marks**
  - Each algorithm description **15 marks**

Deductions will include, but will not be limited to:

- Incorrect run-time behaviour up to **-70 marks**
  - Incorrect sharpening / denoising
  - Errors in the Warp implementation or missing Warp implementation
  - Deviations from the input/output specifications
  - etc.
- Missing report components, including up to **-30 marks**
  - Missing or incomplete algorithm descriptions
  - Missing or incomplete handling of borders
  - etc.

Submission

Submit the assignment using Moodle. Submit only the source code and the makefile. Bundle the code in a zip file.