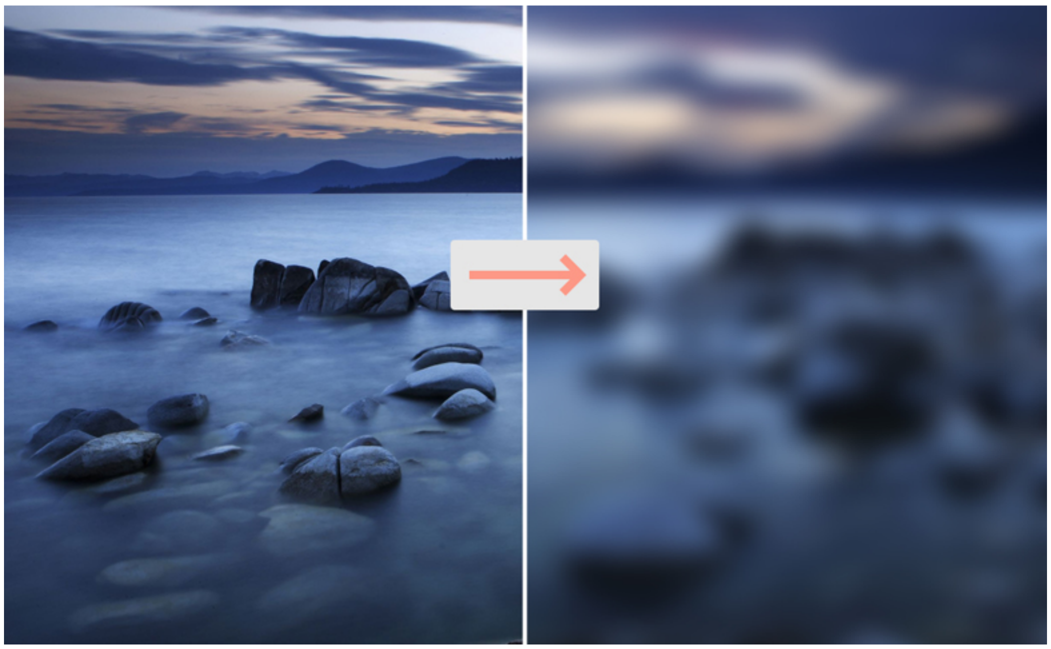
Goal

Design and implement a parallel algorithm to perform gaussian blur of an image using the ispc programming language.



Details

Digital image processing techniques, including image manipulations like distortions, are used across many different computer science domains. Many image manipulation algorithms operate on a per-pixel basis, without dependencies between pixel calculations. Because these calculations are normally independent, image distortions can be parallelized using hardware like vector processors. The Gaussian blur is an example of a simple parallel algorithm to blur pixel values using the average value of neighbouring pixel values.

Pseudo-code for Gaussian blur can be found online and can be used. The student is still responsible for porting any filters to ISPC. Given below is the algorithm to perform a simple Gaussian blur:

1. Read the image into 3 arrays (R, G, and B).
2. For the "R" array, for each pixel, identify the indices of the 8 neighbours (take care of edge-cases).
3. The Gaussian-blurred value for this pixel is the averaged value of its neighbours.
4. Write the newly-calculated pixel value into a **new** "R" array (call it R-modified).
5. Note that the operation described is **independent** and thus parallel for each pixel. (Hint: Use ISPC here to parallelize!)
6. Repeat these steps for the G, and B pixel arrays.

7. Using the R-modified, G-modified, and B-modified arrays, create and save a **new** image.

Steps

1. Develop the C++ code needed to read and write images from files. One option is to use the Cool Image (CImg) library, available here:

   `http://cimg.eu/download.shtml`

   For OSX users, the library can be downloaded using homebrew, `brew install cimg`. Here is a useful example that can be used as a starting point:

   `https://gitlab.com/lamb/Intro-to-Parallel/raw/master/LabAssignment1/grayscale.cpp`

2. Develop the serial C++ code needed to perform Gaussian blur using the algorithm.

   After generating the correctly distorted images, implement timers around the region of code that performs the distortion. Although the command line `time` program can be used to time the entire execution, this includes the time spent reading and writing the files, initializing data structures, etc. The following link contains useful information for implementing timers in C++:

   `https://stackoverflow.com/questions/1487695/c-cross-platform-high-resolution-timer`

3. Develop the ISPC code needed to perform gaussian blue in parallel. This will require rewriting the region of the code that performs the distortion in ISPC syntax.

   Time the execution of the ISPC implementation using the same timers as the serial implementation. (Note, you might try playing around with how you index through the image array, for example, row-wise or column-wise, and see the performance effects.)

Submission

To submit this assignment, create the following directory in your git repository:
***lastname*_parallel/lab_assignment_1/**
Within this directory, please include the following:

1. Both the C++ serial and ispc versions of your gaussian blue implementations.

2. **Please include a Makefile along with your code**. The code should compile successfully on cerberus using the "make" command. **Failure to do so may result in no points being granted for this assignment.**

3. A text or pdf file containing a description of how to make your code (in case using any external libraries or modules apart from the "ispc" module. Also include any observations made, and comparison and justification of the timing results for the serial and ispc versions.

Grading criteria: Total 100 points, may be normalized later

- Code compiles on cerberus using ”make” — 10 points
- Gaussian blur implemented correctly using serial code — 30 points
- Gaussian blur implemented correctly using ISPC — 40 points
- Performance improvement of 2x or greater — 20 points

Assigned: January 15, 2019
Last Update: January 25, 2019
Due: January 26, 2019 (To-be-confirmed with Prof. Allen Malony)