ECE 459: Programming for Performance
Assignment 3

Patrick Lam

March 11, 2013 (Due: March 25, 2013)

This assignment is done in groups of at most 2. Email me with the WatIDs of your group members.

Archive URL:
`http://patricklam.ca/p4p/assignments/provided-assignment-03.tar.gz`

git clone URL (easier to pull changes from:)
`git://patricklam.ca/nm-morph.git`

Reminder: keep your repositories private.

Handy References

- `http://www.cplusplus.com/reference/`
- `http://en.cppreference.com/w/cpp` (C++11)
- `http://en.wikipedia.org/wiki/Beier%E2%80%93Neely_morphing_algorithm` (not as useful as one might hope)

Important Notes

- Make sure your working directory is called "assignment-03".
- You may modify any code or build flags.
- Make sure you run your program on `ece459-1.uwaterloo.ca`.
- I will replace your `test_harness.cpp` file when running your code.
  - Do not try to modify this file.
- You must not change any high level operations of the algorithm.
- You must not change the interface between `test_harness` and `model`.
- If you have any questions about your limitations, email me.

Background

Lecture 17 contains the background information. The source is as I found it on the Internet, except that I've refactored the image-processing code. There are many ways to improve the performance of this program. You may improve the serial execution, parallelize it as much you can and/or use compiler flags. Just remember, you cannot change any high level operation. Do not make any modifications that change the output of the program; this program should be totally deterministic.

Report

Baseline Performance (30 marks)

First, we need a baseline of performance. Take the provided application and run it for at least 3.5 seconds on the default `mountains/beaches` morph. (You can specify different files on the command line). I'd like to find a longer runtime; I may update the assignment with that.

```
% ./test-harness
```

The output file `out.jpg` contains the result of the morph. Now, profile your application with the profiler(s) of your choice.

This is your baseline performance. Record which profiler you are using along with the results.

Identify which parts of the code take up the majority of the execution time. Record your observations. All of your observations should be in the context of the code in `model.cpp` (or other source files); if built-in functions take up a majority of the execution time, identify its caller, in one of those files.

Improvements (at least 2, 60 marks)

Now, it's time for improvements. You'll probably want to target the functions taking the most time. Make at least 2 changes that will improve the performance of the program. For your most significant changes do the following:

- Provide a code listing of the diff of the change from the provided code.
- Profile only this change (using your original baseline, or another baseline for comparsion) showing a large improvement.
- Discuss why this change improves performance and how it does not change the fundamental operation of the program.

Note: a compiler flag is only considered valid for this section if it offers large speedup (at least 20%). You may use as many improvements as you want in order to get the best performance. Just briefly outline any other changes you made to improve performance.

Your Performance (10 marks)

You are going to be evaluated against the other groups in your class. It's simple; the faster you make your program, the more marks you'll get. My intent is to create a leaderboard on the class website, so you can see how you're doing. Your program will be tested on `ece459-1` for this, so make sure it runs how you expect. You may use any optimizations specifically targeting this architecture, although it may not help (reminder, `ece459-1` is a Intel Core i7-2600K).