COMP 206 Assignment 4

Due: Dec. 5th, 11:30pm

1 Pedagogical objectives

Experience with code profiling and methods to speed-up implementations. Mixing Python and C code through ctypes. Work with pre-existing library code written by others.

2 Background - Image Filtering

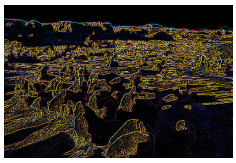
A common operation in graphics and video production is to apply a so-called *filter* to an image through the process of **convolution**. We are considering convolution because it easily shows the differences in efficiency between languages and implementations. For example, the following two commands both apply a Laplacian filter to find edges in an image, but the built-in function is over 100 times faster:

```
$
$time convert utah.bmp -morphology Convolve '3x3: 1 1 1 1 -8 1 1 1 1' utah_edges.bmp

real    0m0.023s
user    0m0.038s
sys     0m0.011s
$time python convolve_slow.py utah.bmp utah_slow_edges.bmp 3 1 1 1 1 -8 1 1 1 1

real    0m4.892s
user    0m4.323s
sys     0m0.076s
$
```

utah.bmp



utah_edges.bmp

You are not responsible for the finer details of convolution, you must just have a basic understanding, and be able to work with some existing code that performs the operation. However, it will probably be useful to read some background at: `http://en.wikipedia.org/wiki/Kernel_(image_processing)`.

3 Problems

1. Modify the provided **convolve_slow.py** script so that it uses the **cProfile** module to print a profiling report. After your changes, the filtering must still be performed properly. The time spent in each function must be printed to standard output.
2. Write a C program called **convolve_fast.c** that performs image filtering using the same command-line interface as **convolve_slow.py**. Do not implement convolution from scratch. Rather, write a simple main function that calls the provided **fast_filter** C library to do the complicated convolution work for you. The resulting program should execute at least 50 times faster than **convolve_slow.py** on the **utah.bmp** sample file for any filter of width 3.
3. Write a Python program called **convolve_ctyes.py** that also uses the **fast_filter** C library to again achieve fast convolution. Follow the same command-line interface as above. Use the ctypes.cdll.LoadLibrary function to connect your Python code to the library and use any other Python or ctypes functionality that you need to correctly pass the data.

4 Restriction on Compute Environment

Since this assignment and the sample code that we're providing are heavily dependent on the library and language formats, this assignment **MUST be done on SOCS computers** such as the mimi.cs.mcgill.ca server or the lab machines in the Trottier building.

5 Submitting

Create a single zip file, A4_solutions.zip that contains your 3 code files: convolve_slow.py, convolve_fast.c and convolve_ctypes.py. If you wish to include any notes to the markers, you may optionally add the file A4_written_solutions.txt. Submit via My Courses.

6 Hints and Resources

6.1 Working with Shared Libraries

- Produce the fast_filter shared library:
  - *gcc fast_filter.c -shared -fPIC -olibfast_filter.so*
- Link with the library libfast_filter.so:
  - #include the library's header ("fast_filter.h") in your .c file
  - gcc option *-lfast_filter* says you want to use the library. Leave off the "lib" and ".so" parts of the filename due to convention
  - gcc option *-L.* says to look for libraries in the current directory when linking
- Run code (Python or C) that needs a shared library:
  - *export LD_LIBRARY_PATH=${LD_LIBRARY_PATH}:.* says to look in the current directory when running
  - This command must be re-run every time you start a new terminal session

6.2 Passing an Array of Floats Using ctypes

ctypes provides the *ctypes.c_float* type which is compatible with float in C. You can create a new type which is an array of c_float's as follows:

```
builtin_float_array = [ 1.0, 2.0, 3.0 ]
CFloatArrayType = ctypes.c_float * len(builtin_float_array)
cfloat_array_instance = CFloatArrayType( *builtin_float_array )
```

The final array instance created above is suitable for passing to a C library when an argument of type *float* is required.

6.3 Viewing Image Results

Since we've required that you work on SOCS computers, if you connect from home using ssh, putty, SecureCRT etc, you may find it awkward to view the BMP images produced by your code. One nice trick is to do your assignment within your public_html folder, and to place a small html page there with these contents:

```
<html>
<img src="output.bmp"/>
</html>
```

If you've done this, and set the permissions properly ("other" must have read access), you can navigate to the page in your brower and use it to conveniently check the file. Any time your result changes, refresh your browser's page to see the new result. An example is at `http://www.cs.mcgill.ca/~dmeger/view_image.html`.

If you choose to work a Trottier lab, it is even easier. Use eog (eye of Gnome) or any other image viewer to check your output.

7 Interesting Filters to Try

Here are a selection of test-cases that we may include during marking:

- **Identity Filter** $ ./convolve_fast input.bmp output.bmp 1 1
- **Emboss Filter** $ python convolve_ctypes.py input.bmp output.bmp 3 -2 -1 0 -1 1 1 0 1 2
- **Mean Filter** $ ./convolve_fast input.bmp output.bmp 5 0.04 0.04 0.04 0.04 0.04 0.04 0.04 0.04 0.04 0.04 0.04 0.04 0.04 0.04 0.04 0.04 0.04 0.04 0.04 0.04 0.04 0.04 0.04 0.04 0.04 0.04 0.04 0.04 0.04
- **LoG Filter** $ python convolve_ctypes.py input.bmp output.bmp 9 0 1 1 2 2 2 1 1 0 1 2 4 5 5 5 4 2 1 1 4 5 3 0 3 5 4 1 2 5 3 -12 -24 -12 3 5 2 2 5 0 -24 -40 -24 0 5 2 2 5 3 -12 -24 -12 3 5 2 1 4 5 3 0 3 5 4 1 1 2 4 5 5 5 4 2 1 0 1 1 2 2 2 1 1 0

 Identity	$$1$$
 Emboss	$$\begin{matrix} -2 & -1 & 0 \\ -1 & 1 & 1 \\ 0 & 1 & 2 \end{matrix}$$
 Mean filter	$$\begin{matrix} 0.04 & 0.04 & 0.04 & 0.04 & 0.04 \\ 0.04 & 0.04 & 0.04 & 0.04 & 0.04 \\ 0.04 & 0.04 & 0.04 & 0.04 & 0.04 \\ 0.04 & 0.04 & 0.04 & 0.04 & 0.04 \\ 0.04 & 0.04 & 0.04 & 0.04 & 0.04 \end{matrix}$$
 Laplacian of Gaussian (LoG)	$$\begin{matrix} 0 & 1 & 1 & 2 & 2 & 2 & 1 & 1 & 0 \\ 1 & 2 & 4 & 5 & 5 & 5 & 4 & 2 & 1 \\ 1 & 4 & 5 & 3 & 0 & 3 & 5 & 4 & 1 \\ 2 & 5 & 3 & -12 & -24 & -12 & 3 & 5 & 2 \\ 2 & 5 & 0 & -24 & -40 & -24 & 0 & 5 & 2 \\ 2 & 5 & 3 & -12 & -24 & -12 & 3 & 5 & 2 \\ 1 & 4 & 5 & 3 & 0 & 3 & 5 & 4 & 1 \\ 1 & 2 & 4 & 5 & 5 & 5 & 4 & 2 & 1 \\ 0 & 1 & 1 & 2 & 2 & 2 & 1 & 1 & 0 \end{matrix}$$