COMP 206 Assignment 5

Building integrated software systems
Due Dec. 5, 11:59PM
Marked out of 100. Total 150 possible offsets any points missed in previous assignments.

1 Pedagogical objectives

Implement a larger software system that includes components from the web, C libraries written by others, as well as your own code written in Python in order to produce a functional integrated system.

2 Background - Image Filtering

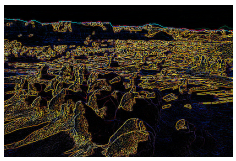
A common operation in graphics and video production is to apply a so-called *filter* to an image through the process of **convolution**. We are considering convolution because it easily shows the differences in efficiency between languages and implementations. For example, a code sample convolve_slow.py can be found in COMP206_A5_provided.tar.gz, and compared with the convert built-in we have used during A1. 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

3.1 Simple filtering (30 marks)

Filters an image by connecting to the libfast_filter.so C library provided, using ctypes. The header file, fast_filter.h, gives the spec for this library. You can assume the library is located in the present working directory and that LD_LIBRARY_PATH has been set appropriately. The filtering process is controlled by the following command-line arguments:

- The input image path
- The output image path
- The width of the square image filter, W, as one integer number
- The filter weights as WxW floating point numbers

Your code must be written in q1_image_filter.py, and should result in the same outputs as convolve_slow.py, but should be much faster (50x speed-up is possible, but not required).

3.2 Filtering with command history (70 marks)

Extend the functionality in Q1 by adding a session history that allows undo and redo, similar to a text editor. Each call to the program implements one image operation, which is specified by the command-line arguments:

- load input_image_path: loads the specified file, discards any existing history and makes this image the active image.
- filter filter_width filter_weights (same argument format from Q1): applies the specified filter to the previously active image. The filtered result becomes the new active image and is placed at the next step in the history. Any previous history from this point forwards must be truncated.
- undo: moves the active image backwards by one step in history, if this is possible.

- redo: moves the active image forwards by one step in history, if this is possible.

The *active* image must always be written to result.bmp in the present working directory. Whenever an operation cannot be completed properly, print a sensible message to standard output. In order to implement the history, only one additional file called history.pickle is allowed be used, and it must also be stored in the present working directory. Your code must be written as q2_filter_with_history.py.

3.3 Filter with web interface (50 marks)

Adapt your functionality from Q2 to form a Python CGI that runs on the SOCS web server in your student account. Use the html file A5Q3_template.html as the starting page for each session. Your code must run when the form on that page is submitted, and must dynamically generate a page with the same elements, forming a recursive CGI that allows filtering with history. Note that we will restrict the filter size to always be 3x3 for this question. Your code must be written as q3_cgi_filter.py. It must be submitted to My Courses, and you must also have a working version of the page running at `www.cs.mcgill.ca/~<yourusername>/A5Q3_template.html`.

A sample of this functionality is located at `http://www.cs.mcgill.ca/~dmeger/A5Q3_template.html`, but note that it is not set up to handle multiple concurrent users. Please use it lightly to get the rough idea and then start to build your own solution.

4 Submitting

Create a single zip file, A5_solutions.zip that contains your 3 code files: q1_image_filter.py, q2_filter_with_history.py and q3_cgi_filter.py. If you wish to include any notes to the markers, you may optionally add the file A5_written_solutions.txt. Submit via My Courses.

5 Suggestions and Hints

5.1 ctypes tips

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_floats 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.

Ensuring no reference between input and output

The provided filtering code requires that the memory represented by out_img_data is different than the input, img_data. Depending on how you load the file in Python, sometimes even copy.deepcopy() is not enough to ensure this. A fail-proof method to allocate a number of bytes from scratch is:

```
out_img_data = ' ' * len(img_data)
```

5.2 Pickling data in Python

The pickle module (import pickle) is the quickest way to achieve the history saving and loading. The matching pair pickle.dump(object, filename) and object = pickle.load(filename) will store a Python object to file, including its references and sub-elements, and restore it again in the same condition. Pickling is often not very efficient, so it may result in slow operation when your history becomes large. This is fine for marks, but you are encouraged to think about faster serialization methods for your own glory (and learning).

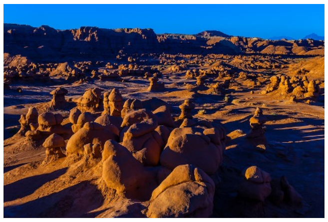
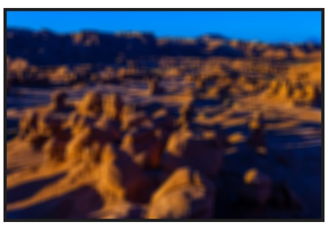
5.3 CGI tips

For your HTML and CGI to be displayed by the web server properly, you must be extremely careful about naming your folders correctly and setting permissions. Follow the directions on this page word for word (i.e., run every chmod exactly as typed there):
`http://socsinfo.cs.mcgill.ca/wiki/Webserver_FAQ#CGI_via_suEXEC`

6 Interesting Filters to Try

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

- **Identity Filter** $ python q1_image_filter.py input.bmp output.bmp 1 1
- **Emboss Filter** $ python q1_image_filter.py input.bmp output.bmp 3 -2 -1 0 -1 1 1 0 1 2
- **Mean Filter** $ python q1_image_filter.py 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 0.04 0.04
- **LoG Filter** $ python q1_image_filter.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	$\begin{matrix} 0.01 & 0.01 & 0.01 & 0.01 & 0.01 & 0.01 & 0.01 & 0.01 & 0.01 & 0.01 \\ 0.01 & 0.01 & 0.01 & 0.01 & 0.01 & 0.01 & 0.01 & 0.01 & 0.01 & 0.01 \\ 0.01 & 0.01 & 0.01 & 0.01 & 0.01 & 0.01 & 0.01 & 0.01 & 0.01 & 0.01 \\ 0.01 & 0.01 & 0.01 & 0.01 & 0.01 & 0.01 & 0.01 & 0.01 & 0.01 & 0.01 \\ 0.01 & 0.01 & 0.01 & 0.01 & 0.01 & 0.01 & 0.01 & 0.01 & 0.01 & 0.01 \\ 0.01 & 0.01 & 0.01 & 0.01 & 0.01 & 0.01 & 0.01 & 0.01 & 0.01 & 0.01 \\ 0.01 & 0.01 & 0.01 & 0.01 & 0.01 & 0.01 & 0.01 & 0.01 & 0.01 & 0.01 \\ 0.01 & 0.01 & 0.01 & 0.01 & 0.01 & 0.01 & 0.01 & 0.01 & 0.01 & 0.01 \\ 0.01 & 0.01 & 0.01 & 0.01 & 0.01 & 0.01 & 0.01 & 0.01 & 0.01 & 0.01 \\ 0.01 & 0.01 & 0.01 & 0.01 & 0.01 & 0.01 & 0.01 & 0.01 & 0.01 & 0.01 \end{matrix}$
 Laplacian of Gaussian (LoG)	$\begin{matrix} 0 & 1 & 1 & 2 & 2 & 2 & 1 & 1 & 0 \\ 1 & 2 & 4 & 5 & 5 & 5 & 4 & 2 & 1 \\ 1 & 4 & 5 & 3 & 0 & 3 & 5 & 4 & 1 \\ 2 & 5 & 3 & -12 & -24 & -12 & 3 & 5 & 2 \\ 2 & 5 & 0 & -24 & -40 & -24 & 0 & 5 & 2 \\ 2 & 5 & 3 & -12 & -24 & -12 & 3 & 5 & 2 \\ 1 & 4 & 5 & 3 & 0 & 3 & 5 & 4 & 1 \\ 1 & 2 & 4 & 5 & 5 & 5 & 4 & 2 & 1 \\ 0 & 1 & 1 & 2 & 2 & 2 & 1 & 1 & 0 \end{matrix}$