README

Parallel Image Smoothing via L0 Gradient Minimization

CS 205 Fall 2012

Project Members:

- Kevin Zhang : kzhang@college.harvard.edu
- Han He : han.he@college.harvard.edu

Introduction:

Serial and parallel implementations of an image smoothing algorithm involving L0 gradient minimization. In summary, major edges within an image can be sharpened while small-amplitude changes are minimized. The algorithm minimizes the overall non-zero gradients between the pixels of an image.

Our final project is created based on the image smoothing method described in the following paper: "Image Smoothing via L0 Gradient Minimization", Li Xu, Cewu Lu, Yi Xu, Jiayi Jia, ACM Transcations on Graphics, (SIGGRAPH Asia 2011), 2011.

Dependencies:

- numpy 1.6.2
- pycuda 2012.1
- scikits.cuda 0.041
- cv2 2.4.3-rc

The above packages should already be installed on the resonance.seas GPU cluster for CS 205.

On a resonance node, `module load courses/cs205/2012` will explicitly load the required modules.

Alternatively, `virtualenv` from the `packages/epd/7.3-1` module an be used to create a new virtual environment on resonance.seas, although numpy, pycuda, and scikits.cuda must be manually installed and configured to run the project code.

Source Code:

Project GitHub

- `L0_serial.py` : serial implementation using CPU
- `L0_parallel.py` : parallel implementation mainly using GPU + CUDA
- `L0_helpers.py` : required helper functions for both implementations, used in algorithm setup before iteration

Referenced below and uploaded on the project git is a sample image that can be used for testing.

Usage:

Run `L0_serial.py` or `L0_parallel.py` with the `-h` or `--help` option to print a detailed help message.

```
-bash-3.2$ python L0_parallel.py -h
usage: L0_parallel.py [-h] [-k kappa] [-l lambda] [-v] image_r image_w

Parallel implementation of image smoothing via L0 gradient minimization

positional arguments:
  image_r        input image file
  image_w        output image file

optional arguments:
  -h, --help     show this help message and exit
  -k kappa       updating weight (default 2.0)
  -l lambda      smoothing weight (default 2e-2)
  -v, --verbose  enable verbose logging for each iteration
```

The input and output image filenames are required, but 2 optional arguments can be used to configure the smoothing parameters.

The `-k` option sets kappa, which controls the speed of convergence. Smaller kappa usually gives higher quality and smoother results but

requires more iterations. Setting kappa to 2, the default setting, is usually the optimal balance between efficiency and performance for natural images. Kappa must be greater than 1 but is usually set to within the range `[1.05, 8]`.

The `-l` option sets lambda, which controls the level of structure coarseness, which relates to the smoothness of the image. Generally, a larger lambda results in fewer edges. Lambda is usually set to within the range `[1e-3, 1e-1]`, but depends on the later uses of the output.

For example, to run the code with default smoothing parameters:

```
python L0_parallel.py images/flower.jpg output.png
```

Or with custom smoothing parameters:

```
python L0_parallel.py images/flower.jpg output.png -k 1.05 -l 0.015
```

Finally, the `-v` or `--verbose` option can be added to print verbose timings for each iteration of the run. Example output:

```
ITERATION 205
-subproblem 1: estimate (h,v)
--time: 0.000051 (s)
-subproblem 2: estimate S + 1
--time: 0.028331 (s)
```

As a note for setting `-k`, for the sample flowers image referenced below, k = 2 requires ~22 iterations, k = 1.05 requires ~301 iterations, and k = 1.01 requres ~1441 iterations.

Sources:

- Original Paper: Image Smoothing via L0 Gradient Minimization
- Sample Image: Bunch of Flowers