MIT EECS 6.815/6.865: Assignment 11:

Halide

Due Wednesday December 4 at 9pm

1 Summary

The description of most of the pset is in the .py tutorial files.

- Smooth gradient. Deliverables: Normalization in Halide. Python variables in Halide. RGB gradient
- Brighten an image. Deliverable: luminance calculation.
- Finite-difference gradient (part I and II). Deliverable: Sobel
- Scheduling smooth gradient. Deliverable: time measurement
- Scheduling the separable blur. Deliverable: Time measurement, Python equivalent code
- Selection tutorial. Deliverable: local maximum
- Reduction tutorial. No deliverable.
- Convolution as reduction. Deliverable: separable Gaussian blur.
- Convolution schedule. Deliverable: timing and best tile.
- Implement Harris corner detection in Halide
- Two schedules for Harris corner detection in Halide. Root for producers of stencils, fast. Timing for Frédo's numpy version and these two schedules.
- Autotuning for Harris.

For time measurement, please also provide your machine characteristics.

Your implementation of Harris corner detection should do the same as the reference python code provided. The part that will look the most different in Halide is probably the local maximum.

For Harris, write two schedules. The first one corresponds to standard Python or C-code and schedules all producers of non-local consumers as root. The second one should be fast and leverage parallelism and offer good locality.

1.1 6.865 only: simple autotuner

This is an open-ended part.

Write Python code that systematically tries many schedules for your Harris corner detector.

Automatically explore different scheduling strategies (e.g. what gets tiled, what gets computed at what granularity) and different numerical values for things such as tile size. Tutorial 10 might give you some idea but it is relatively simplistic. In particular, note that some strategy decision (e.g. tiling a particular computation) can generate new possible choices (tile size) that might not be relevant for other strategies.

Describe your overall approach in one paragraph, turn in your code and your best schedule.

2 Notes

The Halide Python bindings currently exist only on Mac and Linux. Windows users will have to use Athena machines.

You should not need to install anything and halide should work from the included directory.

Be careful with the order of x and y. In this problem set, we use the order `Image[x,y,c]`.

Do not use IDLE. Use Python from a terminal. Halide sometimes crashes the Python interpreter and the easiest way to work is to call `python tutorialXXX.py` from the OS each time you want to run the code.

Do not override Halide's `clamp` function. Call your clamped input something like `clamped`, not `clamp`.

3 Submission

Turn in your images, python files, and make sure all your functions can be called from a module `a11.py`. Put everything into a zip file and upload to Stellar.

Include a `README.txt` containing the answer to the following questions:

- How long did the assignment take?
- Potential issues with your solution and explanation of partial completion (for partial credit)
- Any extra credit you may have implemented
- Collaboration acknowledgement (but again, you must write your own code)
- What was most unclear/difficult?
- What was most exciting?

- Speed of box schedules
- What machine did you use when measuring speed?
- speed of box schedule round II, best tile size.
- description of your auto tuner and its results.