ONI
Oxford Nanoimaging

ONI Take-home Coding Challenge

At ONI, and particularly in Data Science, we are looking to empower both our customers, and our internal employees within the team to rapidly gain insights in a collaborative fashion. Balanced with this is the need for fast and efficient algorithms and numerics, and the interaction with hardware acceleration or external devices.

A key to this is the ability to interact with and utilize C++ libraries for numerics and hardware control, while making this code available to a wider audience cross platform via Python. We approach this problem by providing Python wrappers to C++ code and packaging it via deployable wheels, employing more of our team to use, test, and extend our tools.

Background:

Our microscope images biological samples using a specialized camera, producing unsigned 16-bit images. This process is complicated because a small portion of the millions of pixels on the camera as delivered by the manufacturer behave in a faulty manner, giving inaccurate results.

To correct for this, the “hot-pixels” are detected in a calibration step and encoded in a reference image as negative values. For each image produced by the camera, the “hot-pixels” are then replaced by a simple average of the surrounding pixels in the image (8-neighborhood).

Task:

For your task we would like you to develop a small Python package, that takes in the calibration image, and returns a corrected image of the same size.

The expectation is that the “heavy lifting” of the algorithm will be done in C++, with the function callable and images readable via Python.

Inputs:

- Bead image (uint16)
- Hotpixel image (float)

Output:

Oxford Nanoimaging

- Corrected bead image (uint16), same dimensions as input, with all locations in the Hotpixel image (i,j) with negative values having the corresponding value in the bead image being replaced by the average of the surrounding pixel values in the bead image.

Ideally the function can be run by installation of a python wheel, *cmake* would be used for compilation, and *pybind11* for wrapping the code.

In the interest of reducing the scope, correcting for boundary conditions for the image can be ignored.

If you find other tools that would allow you to make more progress under the short constraints, please feel free to use them, and it is also acceptable to have several sections of work left as "TODO", as the main motivation is to present a piece of work, and discuss your approach and solution together.

Please find the recommended tooling below.

Recommended Tooling:

- Python (version up to you, preferably 3.6+)
- C++
- Cmake
- Pybind11
- pip

Expected time allocation:

3-6 hours ~two evenings.

Follow up:

20-30 minute in person follow up.

Attached files:

example_images.zip

Contains two images saved in the tif format.

- hotpixel_callibration.tif
- bead_image.tif

Oxford Nanoimaging