23-10-2014

Thursday, 23 October 2014

11:33 AM

Summary

1. AGDC data query/wrangling
   1. a separate activity to Sambuca
   2. May not be the most efficient way to interact with AGDC (which contains its own parallel processing capability), but it is the simplest way to get a first-cut Sambuca that has the option of running independently from AGDC
      1. Once the PySambuca per-pixel code has stabilised, it should be possible to integrate this directly into the AGDC per-pixel processing if required (out of scope for phase 1)
   3. Workflow:
      1. Query and retrieve data tiles
      2. Apply tidal offset attribution
         1. Tidal offset data not available until next year
            1. Use hard-coded values for now
      3. Subsurface reflectance correction
      4. Write tiles
2. Sambuca
   1. Best reference for initial python implementation is the Matlab code
   2. Workflow
      1. Entry point
         1. Find and load input data
         2. Parse the options (either from GUI or driver script)
         3. Assemble all the Python components for the run into a "sambuca assembly"
            1. Minimiser
            2. Error function
            3. Sambuca optimisation functions
            4. Inputs (constants, filenames, metadata etc)
            5. Required outputs
            6. Later: serial or parallel Sambuca
         4. Runs the Sambuca assembly
      2. Sambuca
         1. Subclasses to either a serial or tiled parallel version.
            1. Can this be hidden from the entry point? Does it need to be?
         2. Proposes a model
         3. Substrate loop
            1. Discretised linear combination of the substrates

Check the code!:  
What constraints are there on the combinations?  
EG: Given substrates X, Y, Z and coefficients a, b, c, such that the combination used is aX + bY + cZ, do any constraints apply to a, b, c?  
a + b + c = 1   
a,b,c >= 0   
a, b, c <=1

Currently uses a 0.01 step over [0 .. 1] to scale each substrate contribution

Configurable step size would be valuable

In effect, this is a brute force search over some space defined by the substrates (not sure yet if it is a linear, conical or convex space). Theoretically it could be driven by an optimisation function which may be more efficient than the brute force.

Also, the discrete steps used (0.01) might miss the best substrate combination

* + - * 1. Forward model, sensor filter, error check as per diagram
        2. Per-pixel calculations

Does the current code process one pixel at a time to completion before starting the next?

Can I vectorise this, and compute a number of pixels in parallel (via numpy/scipy)?

If so, how would I handle pixels that complete (satisfy error criteria) earlier than others?

* + - 1. Check optimisation error
         1. Is this still per-pixel, or is it only the substrate loop that is per-pixel. Does the green box ("Is Optimisation Error Minimised?") require global access to all pixels?

Check the matlab and IDL code before asking

Other points

* Temporal aspects
  + Tidal offset requires knowledge of the date and time
  + Other aspects of sambuca just operate on the spectra as-is. No temporal aspects to the computations
  + However, a feature I should prepare for involves temporal runs
    - Each run is still time-agnostic, but
    - Outputs from t are used as starting values/ranges to t+1
      * Presumably replacing some/all of the original inputs
* Output variables should be configurable
  + Will need to know the possible outputs so that information is captured appropriately
  + Recorder pattern might be useful if output data needs to be captured before it gets overwritten by Sambuca
* Output raster formats should be flexible
  + Gdal supported formats should do for a start
  + NetCDF is not well supported by ENVI

Post-meeting thoughts

1. Implementation plan
   1. ADGC interactions are a separate activity, to be conducted in parallel with Sambuca
      1. Python 2 must be used in AGDC interactions
   2. Serial Python Sambuca
      1. Since PySambuca is now decoupled from AGDC, could I use Python 3? This would preclude the possibility of calling into PySambuca routines directly from an AGDC pixel processor.
      2. Why?
         1. Allows my initial focus to be on getting the core Sambuca implementation right
         2. I need to understand the data and code interactions before the best way to distribute the work becomes clear.
      3. Primary references: Matlab code & interactions with the team
      4. Secondary reference: the IDL code
   3. Performance enhancement (parallelisation, other)
      1. Standard approach is to have a functional serial version before optimising and scaling out across a cluster
      2. Potential approaches
         1. Course-grained task parallelism
            1. pre-processing

tiles the input rasters

Generates scripts to execute each tile as an independent Sambuca run

batch system jobs on NCI/other clusters

* + - * 1. Tasks submitted to cluster
        2. Post-processing

Reassembles output tiles from each task into final output rasters

* + - 1. In-process course parallelism
         1. Integrated runs, no separate pre and post processing steps
         2. Python code is more complex, as it includes the code to distribute work across a cluster (using MPI or the IPython parallel execution framework)
         3. But if one tile or node fails, the whole job will fail. No possibility to rerun individual tiles
      2. Fine-grained parallelism (at the pixel level)
         1. Vectorised operations via numpy/scipy (SIMD)
         2. Threading
      3. Other approaches that might help
         1. numexpr + numpy arrays (JIT compilation), might be useful
         2. Numba: too bleeding edge, difficult to use with numpy arrays
         3. Cython: code is ugly and too far from standard python. Harder for Sambuca team to maintain