Creation, curation and delivery of high resolution spatial datasets ensuring reliability for product development

Chris Williams*, Andrew Hulbert, and Russell Lawley
British Geological Survey, Environmental Science Centre, Keyworth, Nottingham, NG12 5GG, United Kingdom
Contact: chrwil@bgs.ac.uk

Introduction

- Large terrain datasets provide the foundation for the work carried out by the British Geological Survey (BGS) in developing various environmental data products.
- It is vital that:
  - [1] datasets are kept up-to-date
  - [2] any uncertainty associated with the dataset is effectively accounted for and communicated
- Hurdles hindering these considerations relate to:
  - available memory
  - processing time
- Presented is our workflow for creating derivative datasets from a digital terrain model (DTM) considering uncertainty.

Methodology

- Fully automated Python workflow working on Geotiffs.
- Essentially convolves a D8 window across the dataset to derive slope and aspect values along with associated uncertainty (cf. Heuvelink et al., 1989).
- Data are tiled into 10 km areas according to the British National Grid.
- Tile indexing enables fast searching and partitioning of neighbouring GeoTiff tiles to deal with calculations at grid corners.
- Calculations at each pixel are fully vectorised (*numpy*).
- Uncertainty simulations require the breaking down of each tile into manageable blocks to meet memory requirements.
- Outputs aspect/slope grids with associated uncertainty.

Results and delivery

- Provision of derivative datasets and uncertainty surfaces to be integrated into the BGS product development workflow.
- Minimizes data that need to be held in memory: **reducing memory requirements and processing time.**
- Increases ability to re-deploy as required to incorporate data updates.

Conclusions and next steps

- Moving window operator can be adjusted as required (just pass the function e.g. for roughness etc.)
- Provides the BGS with a ready-to-go uncertainty simulator.
- Uncertainty products will be fully incorporated into all future products and associated updates.

1. DTM preparation: data compilation

- Acquire all tiles.
- Merge into Ordnance Survey defined 10 Km regions.

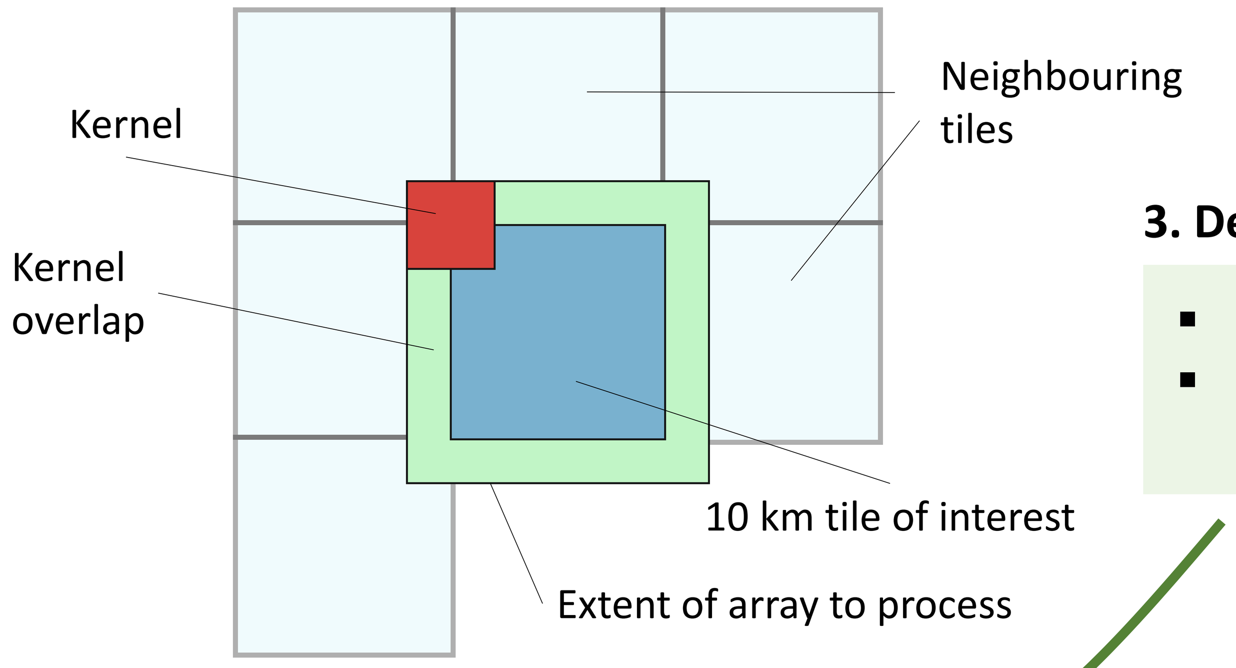
Repeat as new data become available

2. Tile indexing

- Create tile index (Python dictionary)
  - Index details tile names and spatial extent.
- Get first 10 km tile.
- Use the dictionary to identify neighbouring tiles based on spatial extent.
- Keep only data from the tile being worked on and the pixels from the neighbours required for edge processing.

```
# British National Grid

NG100Dic = {
     'HL': [0,120000],
     'HM': [10000,120000],
     'HN': [20000,120000],
     'HO': [30000,120000],
     'HP': [400000,1200000],
     'HQ': [0,1100000],
     'HR': [10000,110000],
      … }
```

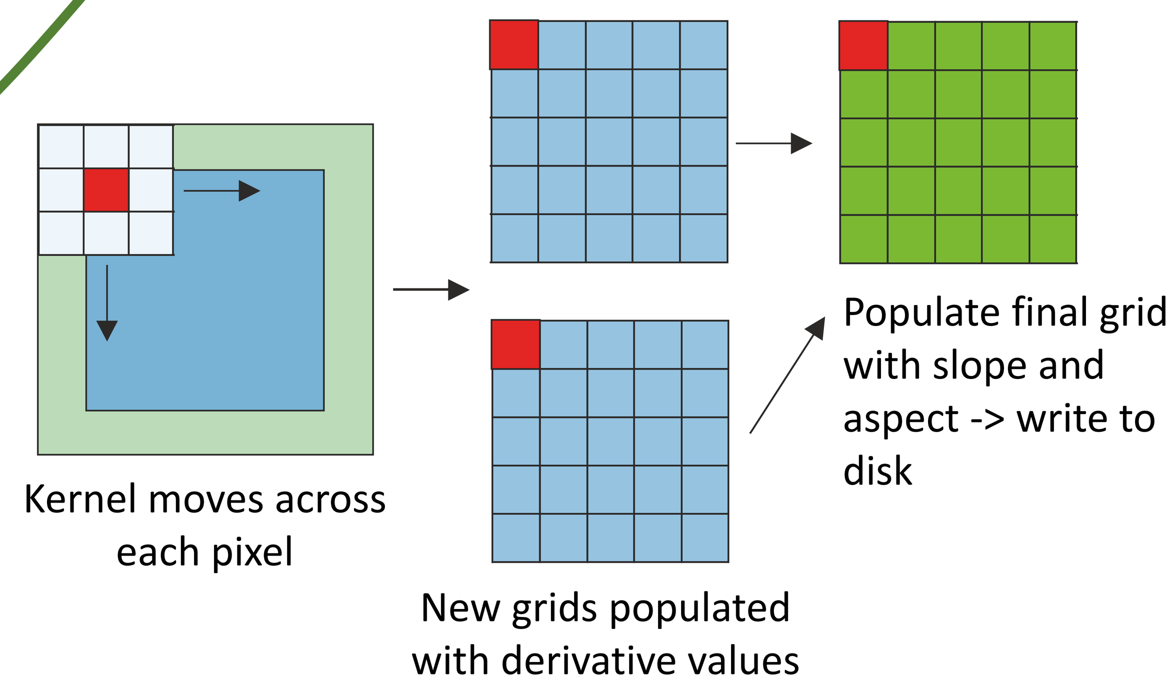


3. Define window function

- Define window size e.g. 3x3, 5x5 etc.
- Define function: for slope we use Zevenbergen and Thorne (1987) and Horn (1981).

4. Implement the function

- Apply window function to each pixel and its neighbours
  - fully vectorised using numpy.
- For a second order slope calculation, this will return the *dx* and *dy* components for each pixel, the values of which populate new grids.
- Calculate slope and aspect given the derivatives.



5. Output

- Repeat until all pixels of the current 10km tile have been processed.
- Write out new slope and aspect grids to disk.
- **Move to the next 10 km tile.**

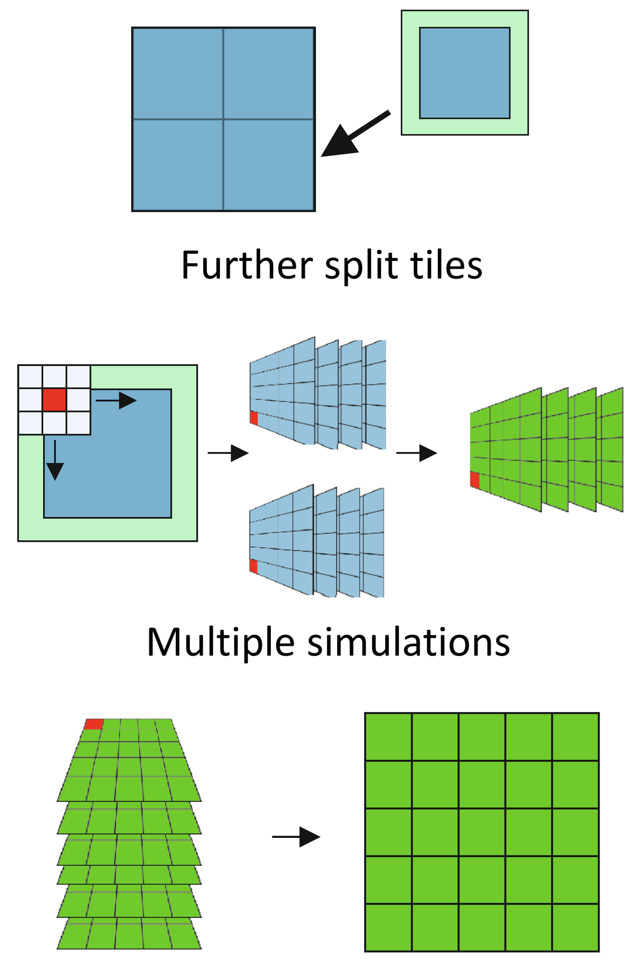
Uncertainty simulation

- The 10 km tile to be used for a simulation is further split into manageable sized sub-tiles (hardware dependent).
- Use tile index to get neighbouring pixels from other tiles.

- Define the number of simulations to be run e.g. n=100
- For each simulation:
  - Add noise (DTM uncertainty) to the sub-tile pixels.
  - For each sub-tile pixel, calculate required derivatives (*as per steps 3 and 4*) and add to an empty grid - a new grid will be created for each simulation.

Repeat on next sub-tile

- Reduce the *n* grids to a single grid consisting of standard deviations of the calculated slope or aspect at each pixel.



Examples of existing approaches for storing and processing geospatial data

- Once you have lots of tiles and you want to store, query and process them more efficiently, it's worth investing time in integrating them into some type of framework.
- Below are some examples of available software which can assist with various use cases, helping with databases, visualisation and processing:

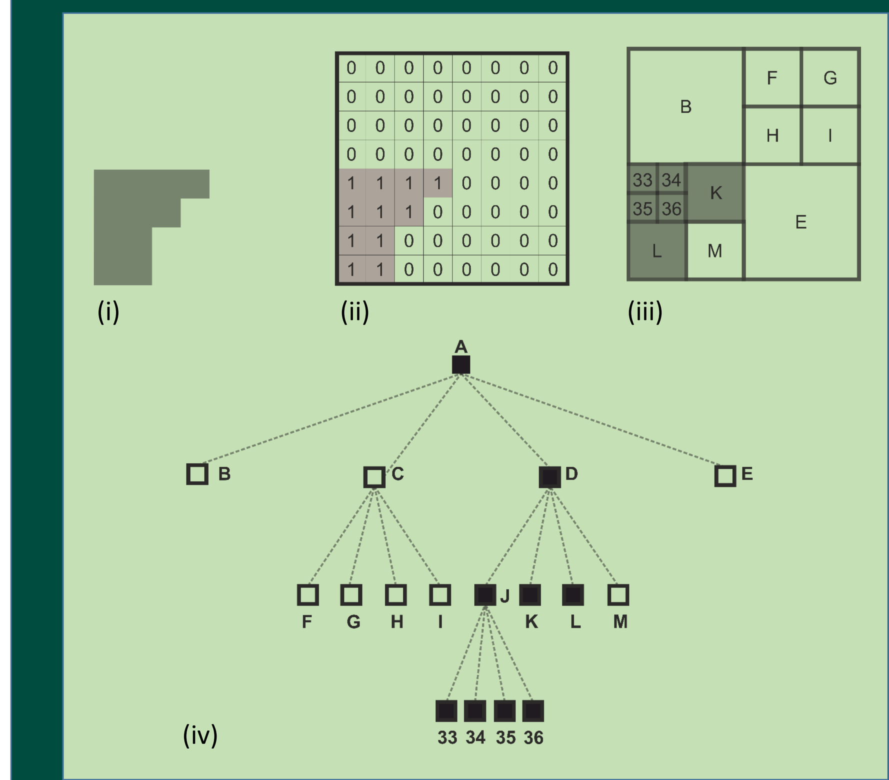
Proprietary
- ESRI Geodatabases
- Oracle Spatial

Open source
- GDAL
- PostGIS (no longer updating raster support)
- Open Data Cube
- Cloud Optimised Geotiffs (COG)
- Pronto Raster

Data structures

- The speed-up achieved by working with your data once integrated into one of the above (or other) frameworks is based on the underlying architecture.
- There are 2 core architectures for spatial data:
  - Raster: Quad-tree
  - Vector: R-tree
- New architectures are being developed to further increase efficiency e.g. K^2-tree (Brisaboa et al., 2017).

What's a quadtree?



i. Spatial area of interest
ii. Binary representation of extent
iii. Raster block break down
iv. Block quadtree

Figure adapted from Samet et al. (1984)

References

- Brisaboa et al., 2017. Efficiently Querying Vector and Raster Data. *The Computer Journal*, 60 (9). pp1395–1413.
- Heuvelink et al., 1989. Propagation of errors in spatial modelling. International Journal of Geographical Information Systems, 3 (4), pp303-322.
- Horn, 1981. Hill shading and the reflectance map. Proceedings of the IEEE, 69 (1), pp14-47.
- Samet et al., 1984. A Geographic Information System using Quadtrees. Pattern Recognition, 17 (6), pp647-656.
- Zevenbergen and Thorne, 1987. Quantitative analysis of land surface topography. Earth Surface Process and Landforms, 12 (1), pp47-56.

British Geological Survey
Expert | Impartial | Innovative