Enabling PointsDC

A capability of the Australian Scaleable Drone Cloud (ASDC)

Dr Adam Steer, Spatialised

Dr Timothy Brown, Australian Plant Phenomics Facility, ANU Node,

13 August 2020

ABN 69 632 365 687 :: https://spatialised.net :: adam@spatialised.net

Contents

Executive summary

The Australian Scaleable Drone Cloud (ASDC) is a collaborative effort between multiple institutions at different stages in their implementations and usage of remotely piloted aircraft (RPA) platforms and data. It seeks to exploit strengths of its partner institutions and deploy a range of tools aimed at helping researchers manage and process data from sensors aboard RPA, or drones. Inevitably, overlapping capability exists because of fairly common data workflows and end user cases.

This document aims to provide a view from the PointsDC capability within the ASDC - tasked with providing visualisation and data services on 3D point cloud products from RPA data collection. A processing and visualisation capability cannot cover all possible data outcomes - effective data usage and visualisation requires consistent input from upstream workflows, and consistent methods for acquiring data products.

With this in mind, this document looks at the entire workflow from aircraft to end user, in order to provide guidance about tooling which can be applied to building ASDC components. The document is structured as a walk through:

- Data visualisation examples
- Engines for processing data
- Managing processed data
- Tools for data exploitation and visualisation

Using a generalised workflow as a guide, this report outlines technologies which fit into various workflow components and provides a 'state of readiness' for each. The design principles for this task are:

- Try to converge on common standards which work and have existing communities
- What are drone pilots and data analysts used to using?
- What are some tools we can introduce to make work easier?
- How can we make sure all this still runs easily a decade from now?

Where new or untested tooling is introduced, it is because it fills a gap and appears to be (at the time of writing) a 'fast win' in an agile sense - get something imperfect happening now and fill out the details as the system becomes stable.

Most importantly, the entire approach is based on open principles: use open software, use open algorithms, use what communities already adopt, use design patterns which are well established. With this in mind a critical consideration in the development of ASDC / PointsDC tooling is *giving back to open source communities*. This simply means 'where we have a criticism or see an improvement, fix it and commit it back to the main repository'. In this way, we ensure that whatever ASDC creates is long-term sustainable, taking advantage of the largest possible community of contributors.

A glossary of acronyms is given at the end of the report, with links to software resources and relevant prior work.

Recommendations

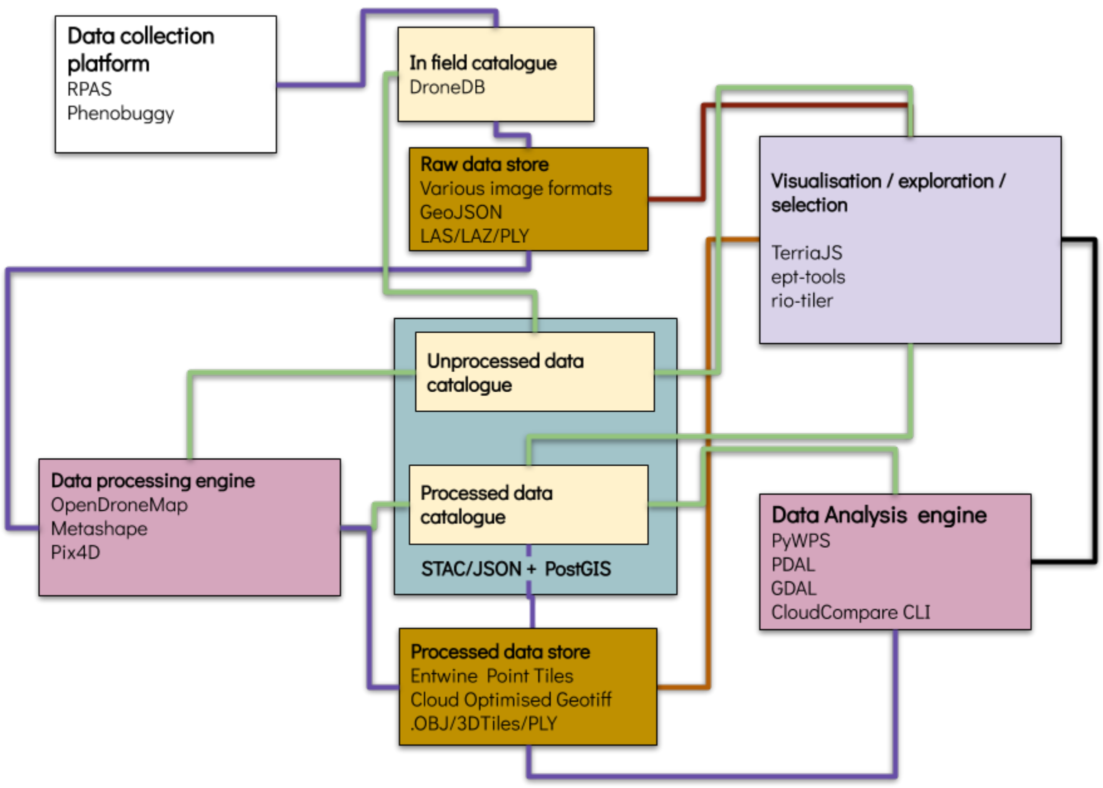
To enable PointsDC outcomes, data from ASDC pipelines must arrive in a set of standardised products for consumption by PointsDC services (see references at end of report for links):

- Point clouds should arrive as Entwine Point Tile (EPT) datasets
- Raster data should arrive as Cloud Optimised Geotiff (COG)
- Point cloud and raster bounds should arrive as GeoJSON geometries
- Camera centre locations should arrive as time-labelled GeoJSON geometries, with camera metadata in geometry properties.
- Image footprints should arrive, or be derivable from camera centres as, timestamped GeoJSON geometries with camera metadata in geometry properties
- Flight paths should arrive as timestamped GeoJSON geometries
- Other metadata should arrive in a standardised JSON format

These enable PointsDC to assemble capabilities from open, standards driven components. PointsDC sees the following software components as 'ready to use', 'close to ready' or 'ready with work' in order to build the system it requires:

- DroneDB as an initial effort to standardise flight metadata. DroneDB is the least ready component - it needs substantial work, however it provides functionality now - allowing ASDC to get started on metadata farming from drone missions.
- TerriaJS/Leaflet/Cesium Javascript libraries for web based data discovery and display
- Entwine to generate EPT point datasets
- Potree and CesiumJS as 3D point cloud viewers
- ept-tools to deliver EPT data to CesiumJS
- Potree / webXR to deliver immersive point cloud exploration
- PDAL as a point data manipulation engine
- PyWPS to provide a standards-driven data processing endpoint

A functional diagram is given below:



Wherever open source tooling is adopted for PointsDC / ASDC purposes, it is critical that improvements funded by ASDC flow back to the community, and the ASDC works with existing communities rather than creating bespoke copies of open source tools. Aside from recognising the fundamental uplift that the project gains from using open software, working in the community provides additional eyes on the project to help fix components, and also provides a means for transparent accountability.

Key drivers for PointsDC recommendations

In order to provide the PointsDC capability, data workflows and products in the ASDC system need to be relatively standardised. Because ASDC requires working examples as a base for iterative improvement, the following body of work is suggested as a general pattern which will help PointsDC (and the ASDC as a whole) achieve its aims:

1. Use **DroneDB** as a field data catalogue and to drive storage catalogues
    a. Work with the DroneDB community to build on this foundation
    b. While DroneDB is imperfect, it is available now and works.

2. Use the SpatialTemporal Asset Catalogue specification (**STAC-spec, or STAC)** as a cataloguing template
    a. Determine metadata required to meet ISO19115, DCAT, and spatiotemporal query requirements
    b. Investigate adopting / extending **STAC** to act as a base 'feature rich' catalogue
    c. Work with droneDB team on **STAC**-like metadata creation

   d. Extract droneDB data and reformat to STAC spec with TERN ontology additions

3. Use **PostGIS** as a dynamic data catalogue, ingesting STAC-spec metadata for spatial queries

4. Use **OpenDroneMap (ODM)** as the initial processing engine, since it is open source and provides processed data in formats pointsDC requires already - especially its web application wrapper, **Web OpenDroneMap (WebODM)**
   a. Work with the **ODM** community on post-processed data cataloguing.
   b. Work on methods for ASDC users to provide their own licensing information for other processing engines

5. Store points as **Entwine Point Tiles (EPT)** (already provided by ODM)
   a. Work on a simple tool for EPT building from **metashape/pix4D** output - which could be implemented as a **Web Processing Service (WPS)** or programmatic tool.

6. Store rasters as **Cloud Optimised Geotiff (COG)**
   a. Work with ODM community to render out orthophotos as **COG** instead of plain geotiff

7. Work on methods for ingesting ‘preprocessed’ or ‘externally processed’ data into the ASDC catalogue
   a. Many researchers have preprocessed data already, and need a data organisation service rather than a processing service

8. Use TerriaJS as a common ‘raw data’ and ‘data products’ discovery interface.
   a. Investigate dynamic data catalogue generation in **TerriaJS**
   b. Work on method for visualising flight data and metadata
   c. Work on tools to drive **DroneDB**
   d. Work on tools to drive **WebODM**
   e. Work on UI tools to drive **WPS**

9. Deliver EPT points as on-the-fly rendered 3D tiles to TerriaJS/**Cesium**
   a. Collaborate on building improvements to EPT tools

10. Use **PyWPS** as a data request API, driven by **TerriaJS** as a user interface
    a. Work with **EcoCommons** (formerly EcoCloud), the birdhouse community and TerriaJS on ASDC-relevant processing services with a user-friendly front end.

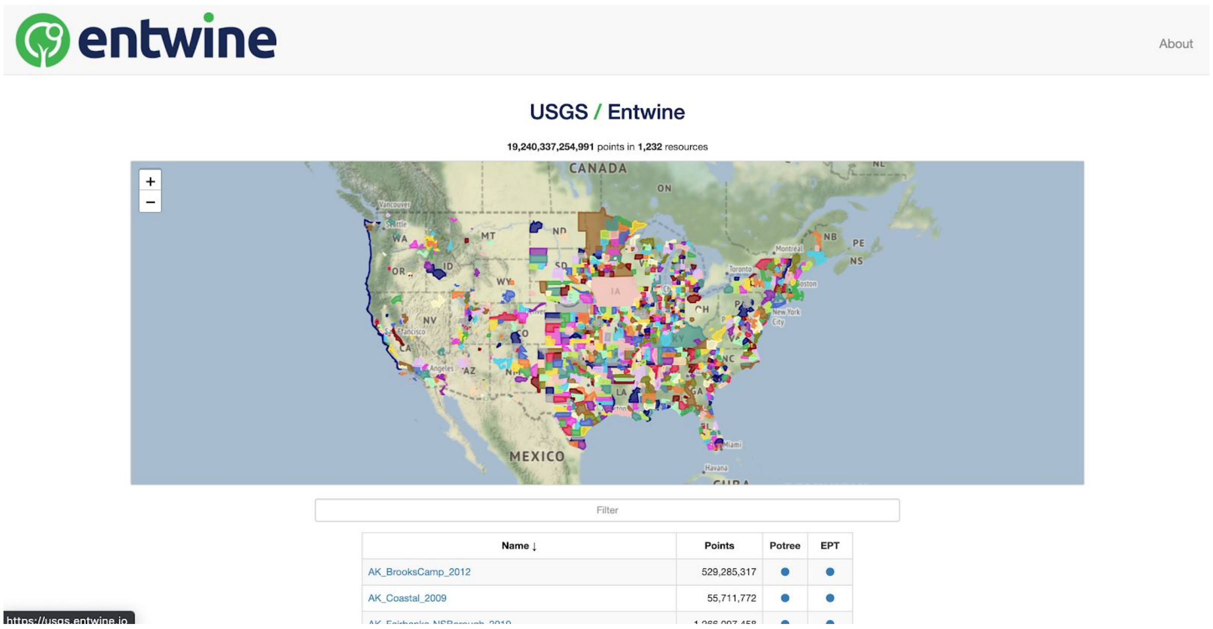
11. Develop point clipping and processing tasks, using PyWPS as a wrapper around:
    a. **PDAL** to read and process EPT
    b. **CloudCompare** command line tooling where PDAL does not fit
    c. **GDAL** to read and process rasters
    d. Other tooling as required for aircraft trajectory analysis

12. Use **EPT** resources in Virtual Reality (VR) viewers
    a. Improve EPT rendering in the Potree viewer
    b. Exploration into relative feasibility of web-based VR visualization using potree or other web-based VR viewers vs Unity or Unreal implementations

Data visualisation examples

USGS lidar as Entwine Point Tiles

The United States Geological Survey (USGS), in collaboration with the Amazon Web Services public data programme, hosts 19 trillion points of lidar data over the continental United States of America using EPT and a simple **leafletJS** + static GeoJSON user interface at https://usgs.entwine.io. On arrival the user is presented with lidar ‘coverages’ as well as a list of available datasets.



USGS / Entwine

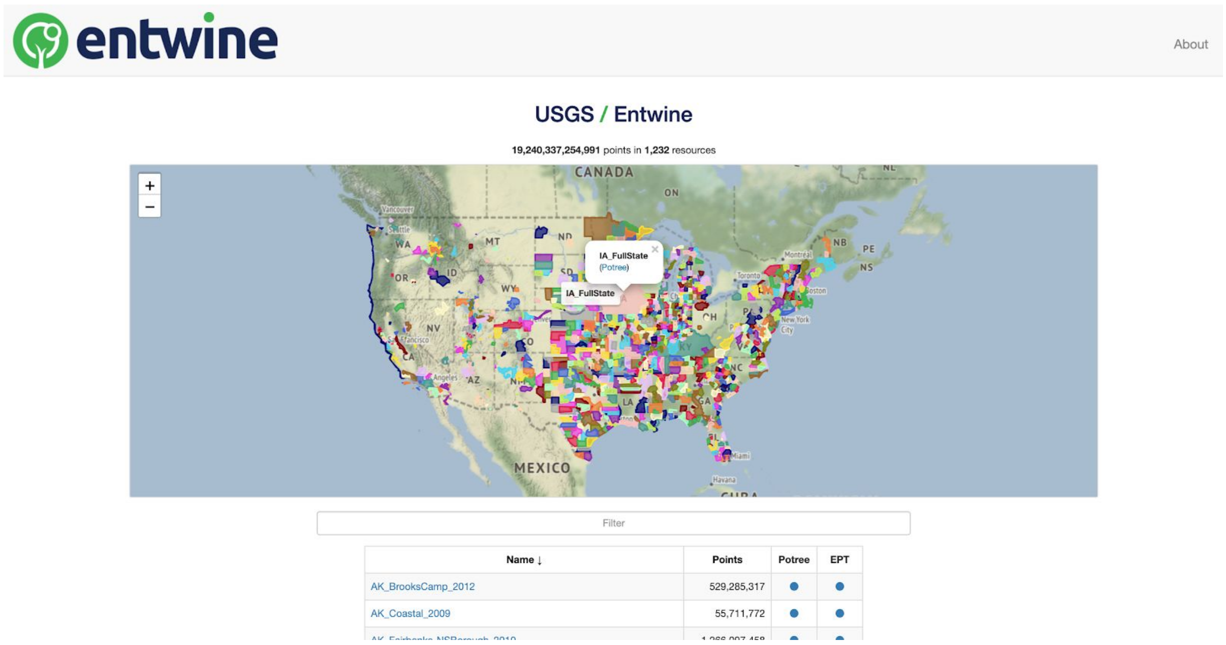
19,240,337,254,991 points in 1,232 resources

Filter

Name ↓	Points	Potree	EPT
AK_BrooksCamp_2012	529,285,317	●	●
AK_Coastal_2009	55,711,772	●	●

USGS lidar data bounds displayed in a simple web map

Hovering over a polygon raises a window with a link to a 3D viewer:



USGS / Entwine

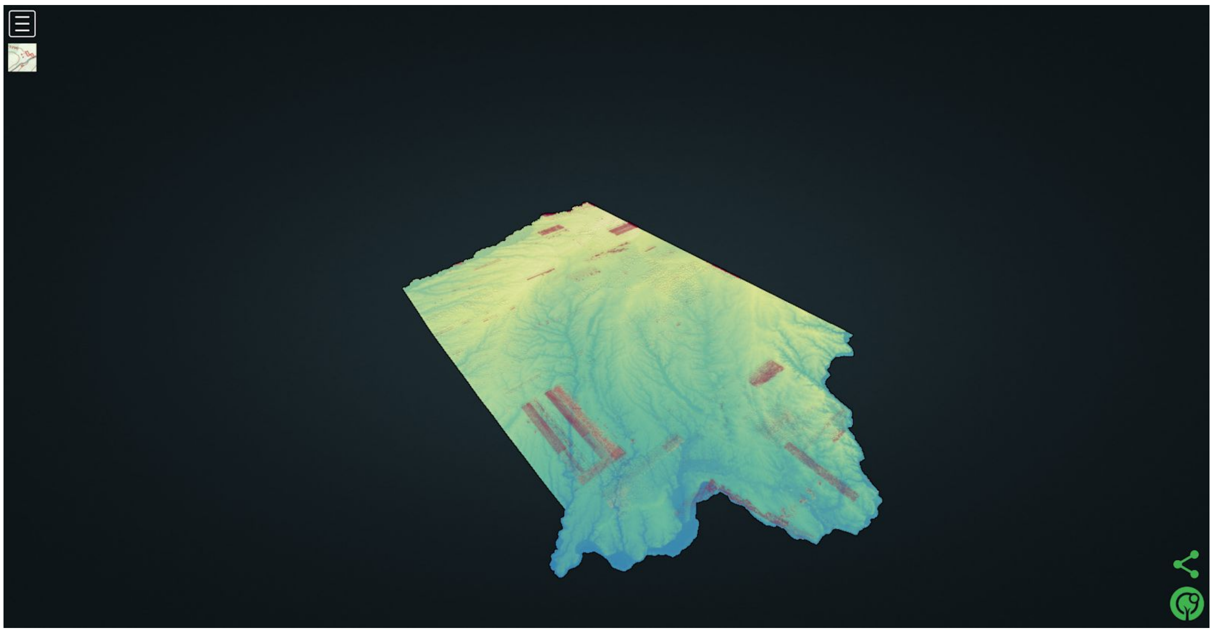
19,240,337,254,991 points in 1,232 resources

Filter

Name ↓	Points	Potree	EPT
AK_BrooksCamp_2012	529,285,317	●	●
AK_Coastal_2009	55,711,772	●	●

Clicking polygons to expose viewing options

Clicking the link takes the user to an interactive web-based visualisation using the Potree viewer:



Selected lidar point cloud data visualised in 3D using the Potree web viewer

This website is open source, with the code available at https://github.com/hobu/usgs-lidar/. It operates in a very similar fashion to WebODM's user interface for post-processed data - offering discrete 2D and 3D viewers for point data. It does not offer processing tooling, although it guides users to PDAL's EPT reader - which is able to query and process data from any of the listed public point clouds directly.

It shows an example of deploying EPT as a standardised point cloud management format, in a lightweight cataloguing and discovery platform. Static GeoJSON is used as both a discovery tool and a catalogue for data extents.

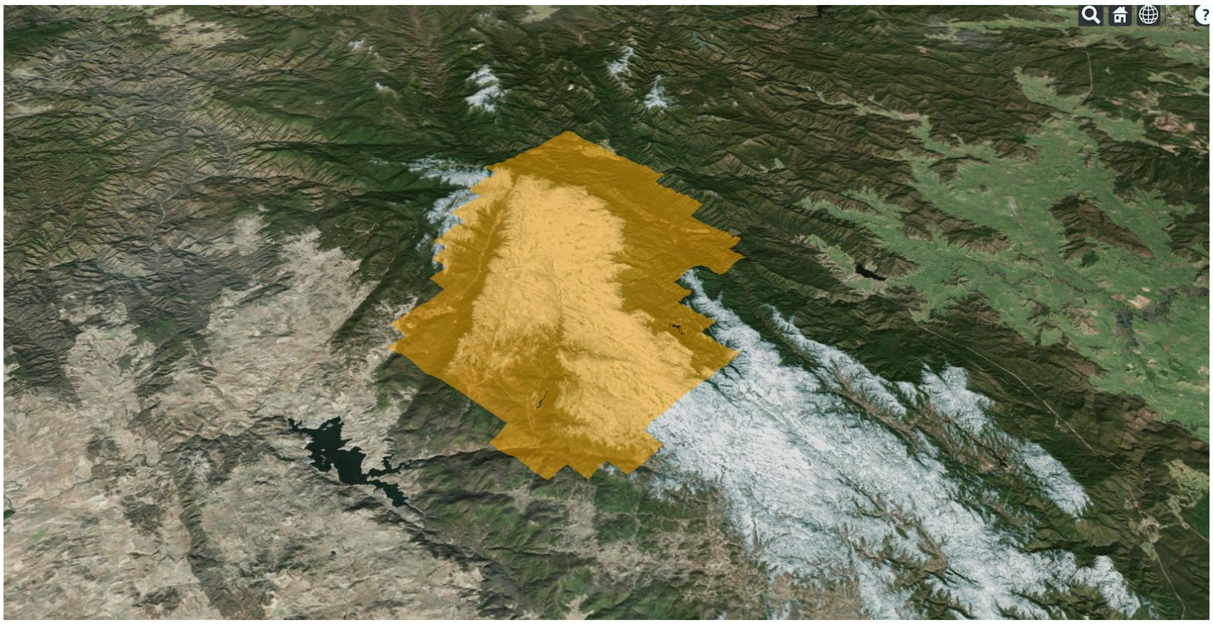
Taking this approach is very lightweight. The primary work package to implement a data exploration interface is to reorganise output products (EPT point clouds and GeoJSON polygons) so that this user interface can be applied.

It is important to note that all point data underlying this approach are reprojected to web mercator (EPSG:3857). Great care must be taken when using these high resolution data for research purposes, particularly accounting for any precision scaling if a dataset extends over a substantial latitude range.

Combining discovery and visualisation using TerriaJS

TerriaJS is a CSIRO data61 product for data exploration using a web-based virtual globe, exemplified in the New South Wales Government digital twin project: https://nsw.digitaltwin.terria.io/. Because it can display both 2- and 3D data in a single user interface, it is an attractive candidate for data discovery and visualisation interfaces. Its core 4D display engine is the open source CesiumJS library. Its native 3D data format is Cesium 3D Tiles, an hierarchical JSON-based data format which was recently adopted as an OGC community standard. 3D Tiles are able to encode vector, point and model data - using level-of-detail dependent cues for adaptive visualisation.

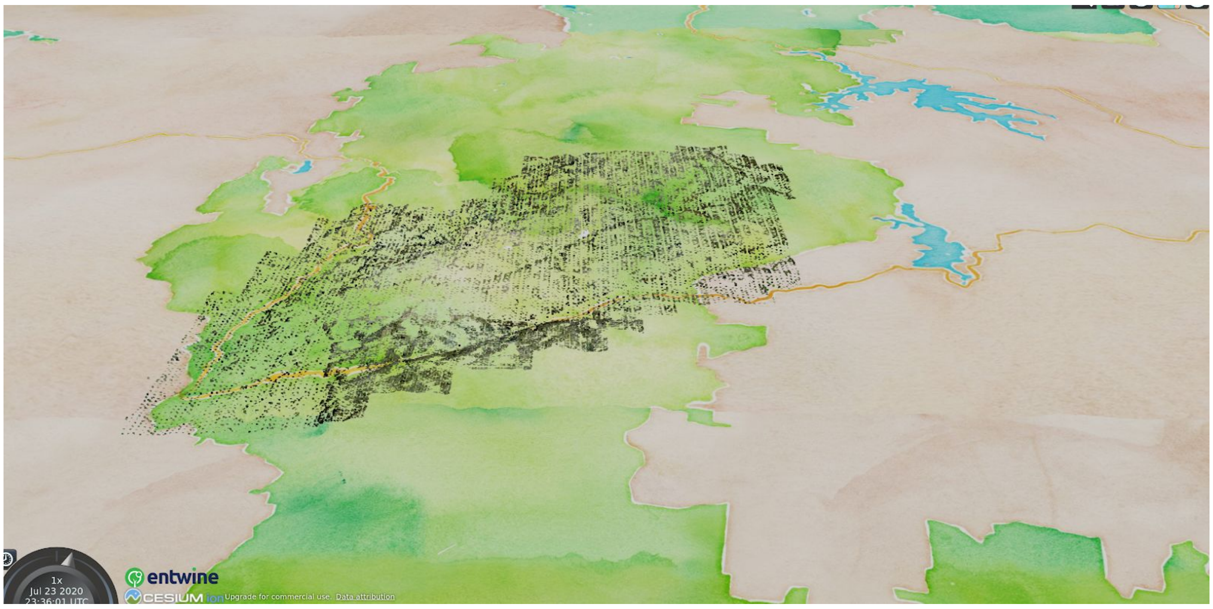
Two dimensional data in most common geospatial formats is also usable in CesiumJS, meaning an interface like the one demonstrated for https://usgs.entwine.io could be rapidly redeployed in 3D using TerriaJS. An example of GeoJSON polygon loading and display is given here:



Lidar point cloud boundaries over terrain in a CesiumJS web-based virtual globe

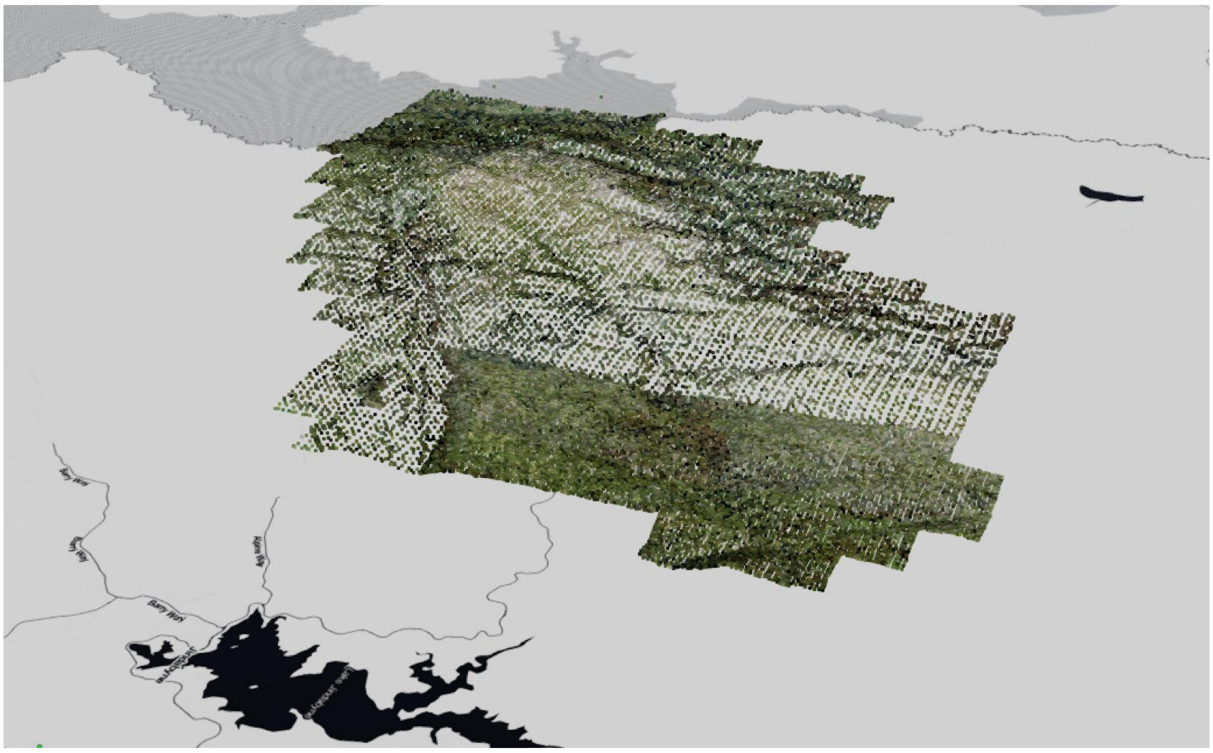
For point data, most approaches at present require conversion of point clouds to static 3D tiles. This has the same implications for storage as using the Potree point cloud format - duplicating data for the purpose of visualisation.

EPT-tools (is a lightweight server application designed to solve this issue. It is an on-the-fly EPT to 3D tiles converter, allowing direct display of static EPT resources in CesiumJS. The following image shows an EPT tileset dynamically rendered to CesiumJS using ept-tools. It covers around 900 square kilometres and contains 615 602 157 points.



Lidar point cloud over terrain in a CesiumJS web-based virtual globe. Note terrain interference.

Because Cesium’s native terrain interferes with the point cloud, the next image shows the same data with terrain turned off in the browser. Here, level of detail based point rendering is apparent.

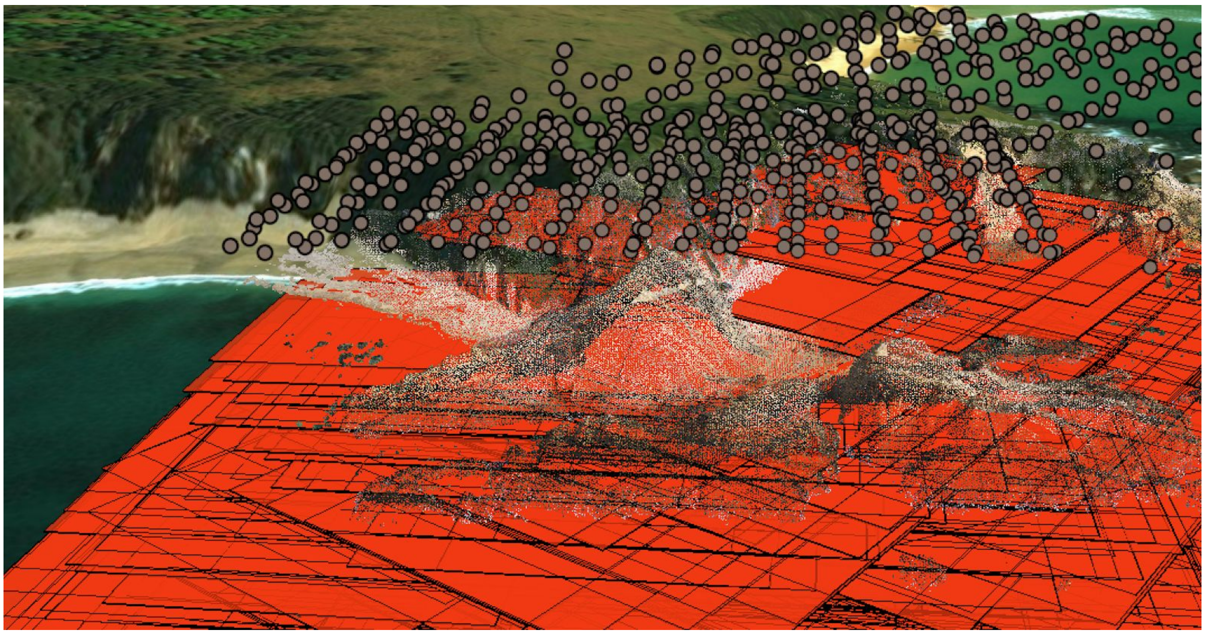


Lidar point cloud without terrain in a CesiumJS web-based virtual globe, showing level-of-detail rendering.

The aim of this approach is to present users with a bounding box showing data coverage until zoomed in close enough to present point clouds. It is not yet fully built out as a functional system - EPT tools expect data to exist on the same filesystem as the server. Ideally data hosted in object stores or on remote filesystems would be equally useable.

A key drawback is that point cloud elevations often interfere with CesiumJS terrain, making exploration with ‘terrain on’ difficult.

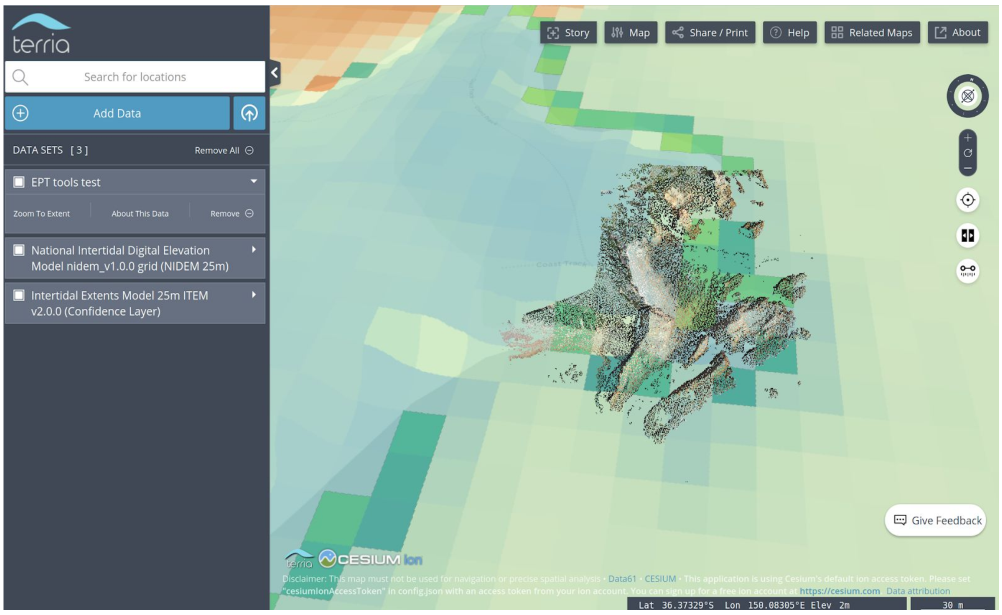
As a final example, TerriaJS can be used for visualising metadata, raw data and data products. This example shows DroneDB-collected camera locations and estimated image footprints (red), with an EPT point cloud rendered on the fly.



Camera centre and image footprint data with derived photogrammetric point cloud, visualised over terrain in TerriaJS

Co-visualisation of georeferenced numeric data

Co-visualisation of georeferenced numeric data comes 'out of the box' with TerriaJS as an exploration platform. This example shows Geoscience Australia intertidal extent model (ITEM)[1] and national intertidal elevation model (NIDEM)[2] data overlaid on CesiumJS terrain, with a photogrammetric point cloud. These are used as examples only. Depending on the use case, these layers could be replaced with spatiotemporal data for landcover, crop health, surface geology, forest cover - in fact whatever data can be displayed using OGC standard services, or any of TerriaJS' data layer types[3].



Disclaimer: This map must not be used for navigation or precise spatial analysis • Data61 • CESIUM • This application is using Cesium's default ion access token. Please set "cesiumIonAccessToken" in config.json with an access token from your Ion account. You can sign up for a free Ion account at https://cesium.com Data attribution

Lat 36.37329°S Lon 150.08305°E Elev 2m 30 m

Photogrammetric point cloud and spatially-coincident data visualised over terrain in TerriaJS

[1] Intertidal Extents Model 25m v. 2.0.0: https://ecat.ga.gov.au/geonetwork/srv/eng/catalog.search#/metadata/113842

[2] National Intertidal Digital Elevation Model 25m 1.0.0: https://ecat.ga.gov.au/geonetwork/srv/eng/catalog.search#/metadata/123678

[3] https://docs.terria.io/guide/connecting-to-data/catalog-items/

Suggested roadmap for implementation

- Present users with data bounding boxes using either leafletJS or CesiumJS
- Allow users to select a 2D or 3D view
- In the latter case, replace bounds with points as users zoom in
- Present a plain potree viewer option as a fallback
- Allow users to click and select data from a catalogue
- Allow users to adjust point elevations to avoid terrain interference
- Allow users to click on, or draw subsetting regions over, data to select it for downstream processing
- Present users with a library of processing capabilities using TerriaJS menus

Suggested work packages

ept-tools
- Remote data asset access for ept-tools (eg serving points hosted in cloud object stores)

TerriaJS
- TerriaJS interface component allowing users to dynamically adjust point data height to just above terrain
- TerriaJS interface tweaks to swap boundaries for points as users zoom in
- Dynamic TerriaJS data catalogue construction using ASDC data catalogue queries
- TerriaJS user tools to allow selection of point datasets or subsets by polygon drawing
- Tools to send drawn polygons and selected dataset details to an API (eg WPS) for processing
- Tools to display a catalogue of data extraction and processing services

Exploring point clouds in XR

Virtual and augmented reality can be grouped into ‘mixed’ or ‘extended’ reality - denoted here as XR. An ability to immersively explore dense point data; and project point cloud data over real world objects has been subject to attention in research, industry and defence. Previous work has demonstrated unlimited scale point clouds in XR platforms (augmented and virtual) based on the Potree viewer (http://potree.org/) and potree’s bespoke octree data format. Its key outcome was breaking a common ‘expensive/bespoke/limited by GPU’ model for point cloud rendering in XR.

Suggested roadmap for implementation

For pointsDC, this demonstration presents a roadmap to XR capabilities using predominantly existing components:
- Implement EPT Octree rendering in Unity and Unreal engines using Potree components

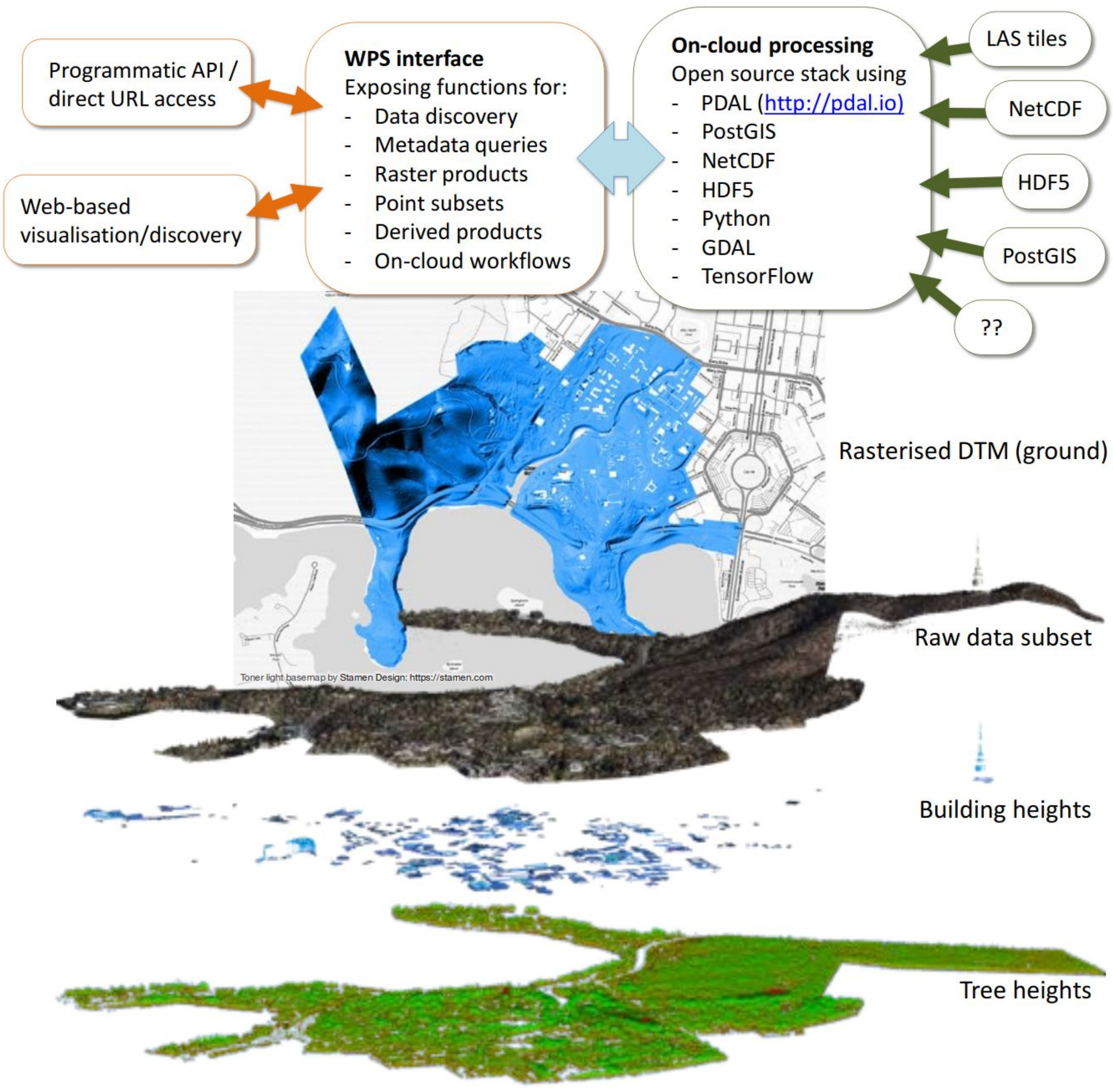
Suggested work packages

- Modify XR renderers to read and display EPT octrees

- Further work on user interaction and data discovery tools in an XR environment

On demand analysis for ASDC outputs

Part of the PointsDC remit is providing easy download / analytical tooling for ASDC outputs. A common issue for researchers is needing to manage huge datasets in order to use a tiny part that they require. Inspired by efforts in the satellite earth observation community, pointWPS was developed at the National Computational Infrastructure in 2017. The project was discontinued at a prototype stage, however it provides a pattern for 'clip and ship' analysis of massive point cloud datasets and is publicly available on github (here). At its core, all it did was provide a standard API to drive PDAL pipelines using data identified by a postGIS cataloguing system. In doing so it proved a concept - using an OGC standard API to deliver what research users wanted on massive point data collections with a minimum of fuss, using the state of technology in 2017.



The PointWPS concept - a method for user-needs driven access to massive point cloud collections

The proposed PointWPS implementation presented at EGU 2017[4] was conceptually useful but represents a number of solutions (for example on-cloud processing of many different source file formats) that have turned out not to be feasible. A longer and more constrained explanation was delivered at FOSS4G 2017: goo.gl/WU7Pwc[5] .

PyWPS continues to be developed, and is in use at the NCRIS ecocommons facility. It has a support community as a formal PSGeo project, and a large user/developer community in the birdhouse WPS project (bird-house.github.io). For ASDC purposes, PyWPS provides an OGC standard API which can be integrated into desktop GIS (eg QGIS) with a library of on-demand, repeatable analytical products.

Suggested roadmap for implementation

- Develop a library of WPS processed derived from the research community
- Develop a spatially-enabled, rich metadata catalogue with at least the data identified here: https://github.com/adamsteer/pointWPS/blob/master/docs/metadata-attributes.md
- Provision a demonstration WPS endpoint

Suggested work packages

- Rewrite pointWPS to better use PDAL Python API and EPT data sources (or abandon pointWPS and start over knowing it was done once and can be done better)
- Rewrite pointWPS to use ASDC data catalogues for metadata queries
- Distributed processing from PyWPS API calls
- UI components to expose available functions
- UI components to allow data selection by drawing polygons

Metadata considerations

Using metadata to power most data queries

PointsDC capabilities rely entirely on data being catalogued with rich and appropriate metadata. Visualisation tools need to know *'where and when*' data exist, and be able to display some form of coverage information ahead of users diving into actual data. Analytical tools (eg cloud based processing services) need to be able to run fast queries about many aspects of data without needing to read the data themselves - for example tight bounding polygons, types of points present, coordinate reference systems, temporal data, point counts and densities, number of bands and so on.

There is a need to provision metadata appropriate to meet ISO19115

[4]https://www.researchgate.net/profile/Adam_Steer/publication/316272527_An_open_interoperable_transdisciplinary_approach_to_point_cloud_data_services/links/58f83b5da6fdcc86f8124fc6/An-open-interoperable-transdisciplinary-approach-to-point-cloud-data-services.pdf
[5] 10.13140/RG.2.2.27345.02409

(https://www.iso.org/standard/53798.html) and DCAT standard (https://www.w3.org/TR/vocab-dcat-2/) descriptions. However, it is not apparent whether either of these provide sufficient data to power the types of spatial queries pointsDC will need.

STAC-spec (https://stacspec.org/), while not a formal standard, provides a community-supported means of enabling deeper geospatial queries, for example '*find me the exact intersection of my query polygon and data*'; or '*in my selected region, find me all datasets which have points labelled as tall trees*'. It has an active development and usage community (eg: https://stacspec.org/#examples). While aimed at earth observation data, STAC uses a modular extension system for other data types - for example point clouds: https://github.com/radiantearth/stac-spec/tree/master/extensions/pointcloud

STAC assets are written out as JSON structures, which are not dynamically queryable themselves. The model for usage is to maintain a static catalogue as JSON, and create 'views' from it as required for active querying (using, for example, PostGIS). This way the static catalogue can be updated and modified in situ, with views refreshed at some time period.

Associated with a catalogue is a file crawling system which, in some cases, automatically extracts rich metadata from incoming datasets. A crawler should also check that datasets continue to exist at the catalogued locations.

Metadata for incoming datasets

At the time of writing the DroneDB project provides a partially complete set for incoming data, and should be actively expanded to collect more information from incoming datasets. LandRS is another option, although it is not ready for usage now. While neither of these derives data according to (for example) TERN plot ontologies, they are both open source projects and therefore extensible as needed.

Importantly, metadata should be programmatically derived wherever possible. Firstly, nobody wants to manually enter thousands of image locations (for example). Secondly, it may be useful to expect that certain metadata are encoded at capture time. Missing metadata (eg camera centre locations) need not be a blocker, it should be a flag that the data may not be usable for functions which depend on those metadata.

Metadata for processed datasets

When data products are constructed, metadata should be programmatically added to the catalogue. Inside the ASDC platform, ASDC has full control over delivering processing parameters and other provenance data alongside geospatial and descriptive metadata.

The key driver here is '*are we keeping enough detail to reproduce these data?*' - along with the considerations for downstream visualisation and processing described above.

When externally processed data are ingested into ASDC, metadata should be programmatically determined as far as possible, with user-entry fields for (eg) processing parameter sets. Again, incomplete metadata should not be a blocking factor - although datasets should be flagged as non-reproducible or of limited use if sufficient information is not provided.

Key data formats

Cloud Optimised GeoTIFF

Cloud Optimised GeoTIFF (COG) is a GeoTIFF format with internal pyramids, designed for fast range read queries.

Because COG is optimised for range reading and contains internal pyramids, it is fast to view in desktop GIS. Its structure allows dynamic map tile service requests using lightweight servers (for example rio-tiler: https://github.com/cogeotiff/rio-tiler).

Like an ordinary GeoTIFF, COGs are analytical products. Again, the format is openly defined and openly described.

https://www.cogeo.org/
https://www.ogc.org/standards/geotiff

Entwine Point Tiles

Entwine Point Tiles (EPT) is a data management format for point cloud data. Conceptually it acts like slippy map tiles for 3D point data, reorganising data at any scale in a lossless static octree. This is designed to allow both data visualisation using level-of-detail capable visualisation tools (Potree, CesiumJS) and data processing using the open source Point Data Abstraction Library (PDAL). On disk, data are stored as compressed LAS 1.4 (an OGC community standard: https://www.ogc.org/standards/LAS), or as binary files using a customised user-provided data schema.

Its key advantages over LAS tiles, or Potree Octrees, or postGIS-pointcloud are:

- EPT assets are designed for both visualisation and processing, a key capability missing in standard LAS formats.
- EPT datasets can be directly read over http by PDAL, meaning a data processing infrastructure already exists
- EPT datasets are fully lossless (Potree is not lossless). Using LAS tiles as an example, if an EPT dataset is constructed from 800 LAS tiles, every single tile can be completely (losslessly) reconstructed from the EPT dataset
- EPT is almost infinitely scalable. It can store data at sub-centimetre resolution or over thousands of square kilometres, or both.
- Metadata from the entire dataset and individual source files is retained in straightforward plain text JSON structures. This makes discovery of data attributes fast and simple.

Using EPT removes a requirement to hold duplicate data for analysis and exploration/visualisation. Both aims are achieved with a single data source, lowering both storage cost and data management complexity. EPT ties analysts into using PDAL for data analysis at the time of writing this report. However, the EPT data structure is openly described and the EPT reader is open source C++ code. It is available for anyone to recast into their favourite analytical toolkit.

Using PDAL, EPT assets can be directly queried by spatial bounds and desired point resolution (loosely translating to octree depth). From there statistical, point based or raster products can be generated in standard formats for consumption by downstream processes.

Its main drawback is that octree structures are written out as many small files on disk. This is undramatic for object stores (eg Amazon S3), but may become problematic for HPC storage. For very large pointclouds, an EPT octree will be composed of millions of files in a flat directory structure.

https://entwine.io/
https://entwine.io/entwine-point-tile.html

GeoJSON

GeoJSON is a description of a JSON format which encodes geospatial information. It is simple to parse in many languages, and useable for both desktop and web GIS applications (eg QGIS and TerriaJS). It is useful in PointsDC terms for storing instrument and instrument platform metadata, for example aircraft descriptions, camera centres (aircraft position and attitude), camera parameters (eg radiometry). In addition, it provides a simple method of describing, visualising and performing operations on processed dataset bounds.

Its primary inflexibility is around coordinate systems - it requires all coordinates to be expressed in EPSG:4326 (WGS 84 latitudes and longitudes). Using GeoJSON-compatible structures to encode different CRS may be an option, as long as the coordinate system used is well described in allowable property declarations.

https://geojson.org/

Key software tools

Visualisation

CesiumJS / TerriaJS

Cesium is a Javascript 'virtual globe' which was developed for the visualisation of 4D geospatial data. TerriaJS uses Cesium as a 3/4D viewer, adding cataloguing and convenient 'drag and drop' data visualisation tools. TerriaJS also adds the foundation for data query tools - drawing regions for data selection, or querying a single point. A key use case for Cesium/TerriaJS is the co-visualisation of multiple data types (raster, vector, points) in one platform. A key drawback is its resource consumption on client computers.

https://terria.io/
https://cesium.com/cesiumjs/

ept-tools

Ept-tools is an application developed by the Entwine Point Tiles team. For PointsDC purposes its role is to convert static EPT indexes to Cesium 3D tiles on demand from a discovery interface. It can run on local EPT data or as an AWS lambda. Ideally, however, remote datasets would be useable, for example an ept-tools instance at “http://eptserver.com” could serve EPT indexes from remote s3 or other object stores, or from remote http:// accessible filesystems. It is also uncertain at the time of writing whether all point attributes are made available to the on-demand 3D tileset. Developing this capability could form a discrete work package for ASDC.

https://github.com/connormanning/ept-tools

Potree

Potree is a javascript application for visualisation of massive point clouds using webGL. It relies on octree-structured points. It can currently read its own data formats and EPT. As a point cloud exploration utility it has a lot of built in functionality, which has helped its popularity. Its key weaknesses are lack of modularity - it is very hard to modify, and its inability to easily display coincident map data.

https://github.com/potree/potree/

Processing

PDAL

PDAL is an open source application for translating and processing point cloud data. It reads and writes many common point data formats, including native EPT reading. It has Python, Julia and Java bindings. There is no GUI for PDAL, it operates entirely as a programmatic interface. It expects the user to manage parallelisation for most tasks. PDAL uses a powerful JSON-based pipeline syntax for configuring complex point cloud processing tasks, using a modular system of data processing stages.

PDAL is an OSGeo project, with a mature community of developers.

https://pdal.io

GDAL

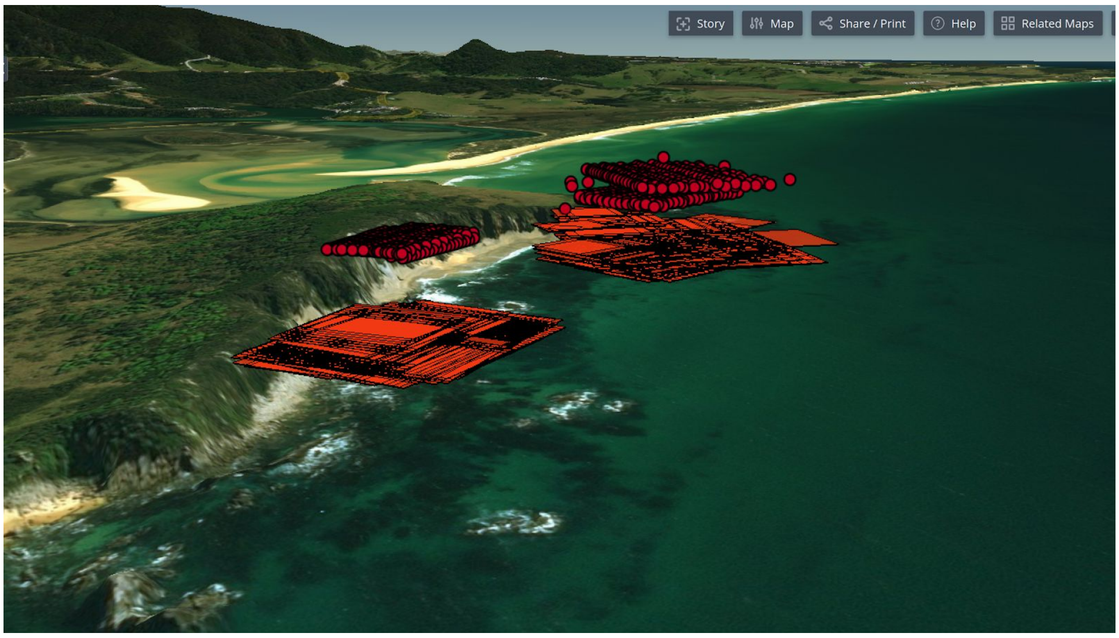
GDAL is used for raster and vector processing in almost every open source geospatial application. It is an essential tool for PointsDC, providing the geospatial underpinnings to PDAL, and allowing convenient raster IO functionality. GDAL uses filesystem and object store drivers, meaning it takes very little work to run the same code on data hosted on either. GDAL can operate on files in memory; and has a powerful virtual raster format (VRT) allowing stacking of operations before any pixels need to be read. In short, it offers methods for lazy compute approaches, and allows for building processing engines with very minimal (sometimes no) local storage requirements.

https://gdal.org

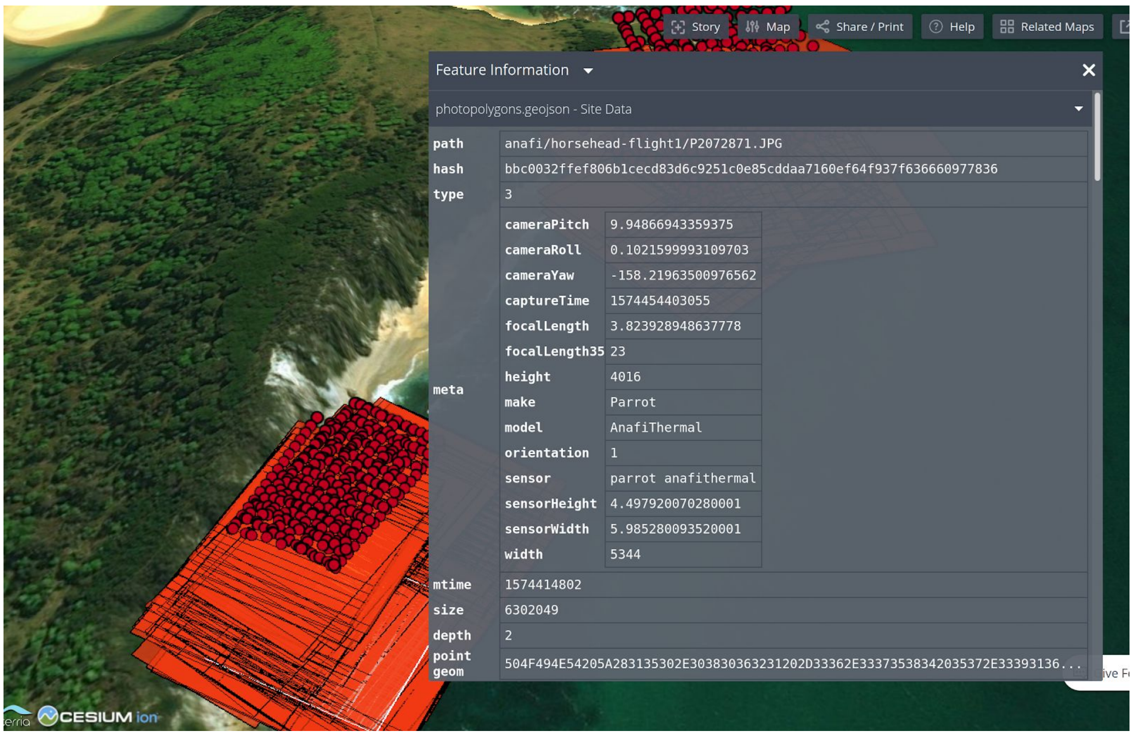
Data Organisation

DroneDB

https://github.com/uav4geo/DroneDB is an open source project in development, aimed at the problem of 'remote users and central management'. It is a system designed to ingest imagery metadata and create a local database for a directory tree of data. These local databases can be used directly in desktop GIS software, and ingested into centralised data stores as opportunities arise. Because it uses spatialite (https://spatialite.org) as a data storage method, spatial queries can be used to generate derived products, for example flight tracks, labelled image sets, or pop-up overlays describing each image.



DroneDB data products displayed in CesiumJS



path	anafi/horsehead-flight1/P2072871.JPG	
hash	bbc0032ffef806b1cecd83d6c9251c0e85cddaa7160ef64f937f636660977836	
type	3	
meta	cameraPitch	9.94866943359375
	cameraRoll	0.1021599993109703
	cameraYaw	-158.21963500976562
	captureTime	1574454403055
	focalLength	3.823928948637778
	focalLength35	23
	height	4016
	make	Parrot
	model	AnafiThermal
	orientation	1
	sensor	parrot anafithermal
	sensorHeight	4.497920070280001
	sensorWidth	5.985280093520001
	width	5344
mtime	1574414802	
size	6302049	
depth	2	
point geom	504F494E54205A283135302E303830363231202D33362E33373538342035372E33393136...	

Camera metadata made available in DroneDB data products, exposed by clicking a camera centre point in CesiumJS

DroneDB provides an immediately-useable toolset for ASDC to work with, and has some key benefits:

- Field-first design
- Already interoperable with web and desktop data visualisation tools
- Already interoperable with metadata storage catalogues
- Extendable - adding TERN and RDA ontology concepts can be done at any point
- Early in the project - the roadmap is still being constructed, giving ASDC the opportunity to have a product built ground up with ASDC needs in mind.

DroneDB competes directly in the same space as the LandRS proposal developed as part of the RDA UAS interest group. At present it has key advantages in that it exists now and can be used for data collected by a huge range of off the shelf (or bespoke) aircraft + sensor combinations. It is relatively low in complexity, and lets researchers use what they feel comfortable with for flight planning and execution.

Ultimately DroneDB may end up as one option people can choose to use, although at the time of writing it seems substantially more useful to more pilots and researchers.

https://uav4geo.org/dronedb
https://uav4geo.com/software/dronedb

Entwine

Entwine is a set of tools for writing entwine point tiles. It is open source, lightweight and configurable. It is highly integrated with PDAL, which has a specific EPT reader. Because writing EPT is considered a one time operation for most datasets, the Entwine writer is managed separately from PDAL's standard set of data translation tools. Being a living software suite, that may change given user demand and funding.

https://entwine.io

PostGIS

PostGIS is a strawman in this case - since any database with spatial querying will do. PostGIS has a very mature development and user community, and can handle all of the metadata queries a pointsDC capability would require, before heavyweight data lifting. For example:

- Data discovery - finding datasets overlapping a query polygon
- Metadata extraction - if sufficiently data-full, handle queries about data completeness, labelling status, point density, accuracy, and so on.
- Usage of static views into the catalogue, allowing background updating of 'living' static or dynamic catalogues without service interruption.

PostGIS handled all of the metadata queries for pointWPS, using a nightly-generated static view into the NCI 'Metadata Attribute System' which was exposed to pointWPS via an http API. An example data schema is given here: https://github.com/adamsteer/pointWPS/blob/master/docs/metadata-attributes.md. This pattern is repeated often in earth observation, because datasets grow so large it is necessary to perform many operations on metadata ahead of firing up machinery to actually process data.

Delivery

PyWPS

PyWPS is a python implementation of the OGC Web Processing Service standard. It provides the OGC WPS API for driving data processing tasks on somebody else's computer. Run as a server which tasks other machines to do processing work, it is lightweight and flexible. The processes which can be run are limited by available compute power and an ability to write Python. As an example, pointWPS used Python to manage jobs and outputs. All the processing tasks were achieved using shell scripts. PyWPS is already in use by the NCRIS ecocommons facility (https://github.com/ausecocloud/silvereye_wps_demo) and has an international community of practice in the birdhouse climate analysis tools: http://bird-house.github.io

https://pywps.org/

rio-tiler

Rio-tiler is a python library which, in the ASDC context, has one job: act as a tile server which translates from COG to TMS on the fly. It is already deployed in webODM to deliver map results in a browser interface.
https://github.com/cogeotiff/rio-tiler

Glossary

ARDC: Australian Scalable Drone Cloud
APPF: Australian Plant Phenomics Facility
B2: Backblaze object storage (https://www.backblaze.com/b2/cloud-storage.html)
CASA: Civil Aviation Safety Authority (Australia)
COG: Cloud optimised geotiff, a format for geo-located image data (https://www.cogeo.org/)
EPT: Entwine Point Tiles, an organisation structure for point cloud data (https://entwine.io/entwine-point-tile.html)
EXIF: Exchangeable image file format - an image metadata transmission standard (https://en.wikipedia.org/wiki/Exif)
GeoJSON: a standardised method of writing Javascript Object Notation (JSON) files to hold geospatial data (https://geojson.org/)
GeoTIFF: 'geospatial tagged image file format' - an image format capable of holding geospatial reference data in its header, giving locations for pixels (https://github.com/OSGeo/libgeotiff)
LAS: a standardised data exchange format for lidar products (https://github.com/ASPRSorg/LAS)
LAZ: a compressed las format (https://laszip.org/)
LIDAR: light detection and ranging, in this case detecting objects overflown by airborne lidar sensors to produce 3D maps of those objects. Lower case in use (eg 'The lidar...')
NCRIS: National Computational Research Infrastructure
OGC: the Open Geospatial Consortium, a body concerned with geospatial data standards (https://www.ogc.org/)
OSGeo: the Open Source Geospatial foundation, a body supporting development and use of open source geospatial software (https://osgeo.org)
RGB: red, green, blue. The three colour bands of 'regular photography' imaging
RPA: Remotely Piloted Aircraft, informally 'drone'.
RPAS: Remotely Piloted Aircraft System (including ground control segment)

S3: Simple Storage Service, an object storage service run by Amazon Web Services (https://aws.amazon.com/s3/)
XR: eXtended Reality - used as a catch-all term for 'Virtual Reality' (VR) and 'Augmented Reality' (AR)

References

Software and tools

Cloud Optimized Geotiff: https://www.cogeo.org
DroneDB: https://uav4geo.com/software/dronedb
DroneDB registry: https://github.com/DroneDB/Registry
Ecocommons Silvereye WPS: https://github.com/ausecocloud/silvereye_wps_demo
Entwine: https://github.com/connormanning/entwine
Ept-tools: https://github.com/connormanning/ept-tools
LandRS: http://www.landrs.org
OpenDroneMap: https://www.opendronemap.org
PDAL: https://pdal.io/
PointWPS: https://github.com/adamsteer/pointWPS
Rio-tiler: https://github.com/cogeotiff/rio-tiler
WebODM: https://www.opendronemap.org/webodm/

Related organisations

https://www.rd-alliance.org/groups/small-unmanned-aircraft-systems%E2%80%99-data-ig

Related proposals

https://github.com/landrs-toolkit/DroneDataBuddy

Related presentations and papers

PointWPS (Adam Steer, FOSS4G 2017, Boston):
https://docs.google.com/presentation/d/1XTrg0tMc7uLkJa8d2ezVAP_tb0VOpdPeKGjGeuNw4qw/

Managing continent scale point clouds (Connor Manning, FOSS4G 2019, Bucharest):
https://data.entwine.io/slides/foss4g2019.html

Examples of related commercial products

End to end solution:

- Dronedeploy: https://www.dronedeploy.com

Drone flight tracking:

- https://airdata.com
- https://www.dronelogbook.com/

Drone data ingest, mapping and modelling:

- Propellor aero: https://www.propelleraero.com/

Example commercial implementation of ODM:

- Aerosurvey: https://aerosurvey.co.nz/