UAS/Spatial Data Archiving

Large Data and Model Data Archiving Session:

ESS-DIVE Community Data Workshop - 2021

NGEE-Arctic UAS data archiving example

NGEE-Arctic DOI: https://doi.org/10.5440/1778212

- Metadata for a multi-instrument UAS platform used for Arctic vegetation remote sensing
- Provides basic metadata on the platform/airframe, instruments, data collection, data products, and data spatial reference information
- Metadata based on an airborne/multi-sensor example, NASA G-LiHT
- UAS products archived in levels, from raw to more derived: L0, L1, L2, L3
- UAS platform and data products methods found in a published article (Yang et al., 2020; *Remote Sensing*) provided with the archives
- Projection information is available in imagery, and *.json files to allow for later referencing by the portal (if such support exists in the future)

Remaining challenges for the community

- **Searchability** - how to make sure that UAS and other spatial data can be found when searching for data in a specific area/region/type. Synthetic data curation (combining point, survey and spatial data in one search/retrieval)? Need to enable search bounding boxes (as a search parameter)
- **File structure:** file hierarchy (eg. different products/levels of processing) to facilitate simple curation / uploading of large datasets comprised of multiple images, locations, etc. Done in a manner to simplify data retrieval (i.e. not “zipped”)
- **Metadata quality:** Ensuring that all of the required information for proper display, projection, reprojection are provided for all contributions. Do we develop a QA/QC check system to identify critical missing information. That would ***require parsable metadata format (e.g. .txt, JSON, XML)*** to allow for automated, machine checks on submissions
- **How to “future proof” cloud-support of spatially referenced data** to allow for spatial search/queries, online mapping/viewing tools, Google EarthEngine integration, etc. Re-creating or updating files in the future would be a significant amount of work, how can we minimize file conversion in the future by providing the basic requirements now?
- **Upload and download API support:** While downloading one or two images for a workflow may be fine, the era of pulling down large data directly is over (for the most part). Given the immense amount of data available, tools and workflows will need to move toward more “cloud-based” support, e.g. **cloud preprocessing/processing/extraction, remote access, API, exposed URLs** etc
- Where appropriate, **harmonize UAS/spatial datasets with other agency standards:** E.g. USGS, NASA, NSF
- Need to **build spatial-data consistency across ESS-DIVE** - where are there overlaps with other datasets, like spatial model output? Can we leverage common support and tools within ESS-DIVE; to what extent do we even use the same file format?

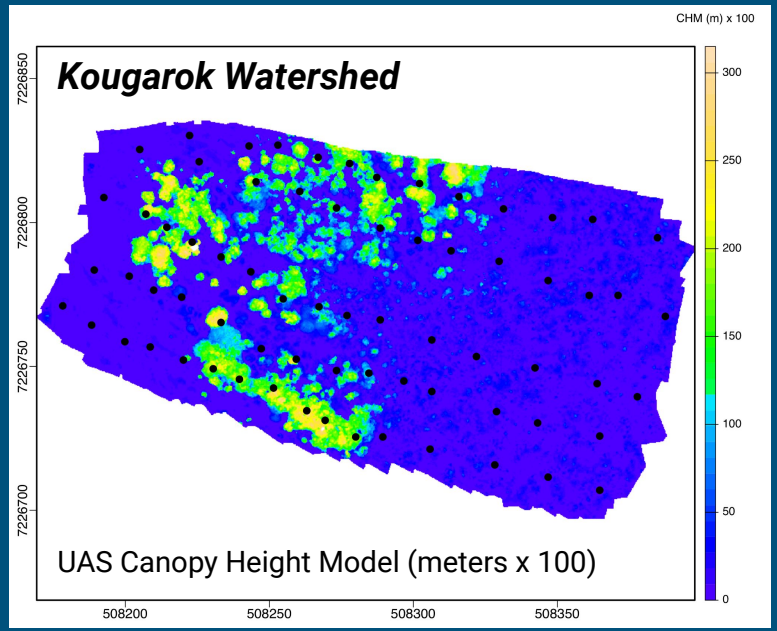
Current effort based on ad-hoc resources, requires dedicated funding to develop a fully-fledged standard

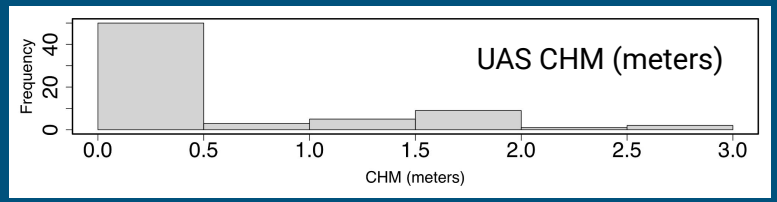
Cloud-optimized GeoTiffs (COGs)

R Script Example (https://github.com/TESTgroup-BNL/exploringCOGs):

```
Library("terra")

chm_cog.url <- "/vsicurl/https://osf.io/2mepa/download" #!! create virtual raster in memory
chm_raster_name <- "NGEEArctic_UAS_Kougarok_20180725_Flight6_CHM_cog_deflate.tif"
#!! Create raster name for later plotting

example_kg_points <- rgdal::readOGR(file.path(here(),"shapefiles"),
                    "kg_example_points")
#!! Load available shapefile data

chm_ras <- rast(chm_cog.url)
names(chm_ras) <- "CHM_meters"
#!! Get virtual raster and update the data layer name

#!! Create figure from virtual raster
pdf(file.path(outdir,gsub(".tif",".pdf",chm_raster_name)), height=9, width=11)
terra::plot(chm_ras, legend=TRUE, axes=TRUE, smooth=FALSE,range=c(0,315),
        col=topo.colors(35), plg=list(x="topright",cex=1,title="CHM (m) x 100"),
        pax=list(sides=c(1,2), cex.axis=1.2),
        mar=c(2,2,3,5.5)) # b, l, t, r
box(lwd=2.2)
dev.off()
#!! Extract CHM data through shapefile points
chm.data <- terra::extract(chm_ras, terra::vect(example_kg_points))
chm.data[,2] <- chm.data[,2]*0.01
```

Yang et al., (2020)





Why use COGs?

- A specialized GeoTIFF that enables more efficient workflows in the cloud environment. Leverages client-based HTTP GET range requests to pull only the parts of the file they need, rather than downloading the whole file. Contains embedded overviews and tiles for rapid viewing and plotting of the raster without downloading / accessing the full file. ***Similar functionality to a THREDDS server*** (netCDF support), but with ***much less effort than setting up THREDDS backend***

- Provides efficient streaming of spatial data enabling cloud-based geospatial workflows. On-the-fly processing of rasters/virtual rasters (in memory). Compatible with R, Python, and really any other software package via gdal

- Can avoid data duplication and issues of versioning as workflows always reference the authority version through the data portal instead of relying on local copies

- Community is more familiar with GeoTiffs than HDF5, netCDF, etc. Also substantial legacy software support for GeoTiff's (e.g. older ArcGIS, spatial libraries on older HPCs, etc). Some missions and portals are also moving to COG format (e.g. Landsat, possibly NASA SBG)

Example illustrating the use of COGS

https://github.com/TESTgroup-BNL/exploringCOGs

- Based on NGEE-Arctic Seward Peninsula UAS data (Yang et al., 2021; in review)
- Uses datasets on OSF.org to provide a exposed URLs for each COG file to provide remote HTTP access in the R script: https://osf.io/x6uq4/
- Illustrates how to view data as a virtual raster in R, make example maps from the embedded overview layers, extract data from the actually imagery (via HTTP GET Range requests) through spatial points, and ***subsetting the data***
- Another OSF example (to see how a UAS project is organized, non-COGs data): https://osf.io/sn6em/