FORMATS FOR DIGITAL ELEVATION DATA MANIPULATION ON MOBILE DEVICES

Peter L. Guth
Department of Oceanography
United States Naval Academy
572C Holloway Road
Annapolis, Maryland 21402
pguth@usna.edu

Filipe Vieira
Évora University, Portugal
sv.filipe@gmail.com

ABSTRACT

The rise of handheld smart phones and tablets brings sufficient storage capacity and processing capabilities to perform significant terrain analysis on the local device. This work can be done as either native apps, or using the capabilities of HTML5. Digital elevation models (DEMs) come in a wide variety of formats, and it does not make sense for handheld device have translators for all of them. The handhelds need a simple format that compresses well, but retains accurate geospatial registration information. We created a JSON format with bzip2 compression, both of which have widespread support with JavaScript libraries. This format allows data only on the WGS84/NAD83 datum, in either geographic or UTM coordinates. A command line converter, based on MICRODEM supplemented with the GDAL libraries, performs DEM reprojection if necessary, converts to JSON format, and compresses for download to the handheld device. Lidar point clouds are now supplied almost exclusively in the LAS format. While the LAS format has a number of options, all include far too many fields for use on a handheld—for instance the 8 bytes for GPS time will not help someone in the field seeking to visualize terrain. We chose a similar JSON/bzip2 solution for MICRODEM to prepare point clouds, and achieve lossy compression roughly comparable to lossless LAZ compression. A GeoJSON format specification exists, but only applies to vector data. We propose GeoJSON-G for grids and GeoJSON-P for point clouds, which will retain full geospatial registration information while intelligently limiting the volume of data for the handheld devices to download and process. The handhelds will not edit modify the data, making elimination of unnecessary fields acceptable. We have adapted HTML5 and Javascript open source projects to display both DEMs and lidar point clouds.

KEYWORDS: lidar, formats, handheld devices, algorithms, point clouds

INTRODUCTION

Handheld phones and tablets threaten to overtake desktop and laptop computers, much as the personal computer displaced mainframe and minicomputers. Computer mapping and geolocation have been driving factors in this migration to the user's pocket, and map data has been making the migration as well. Most of the common data on the mobile platforms has been vector data like roads and raster imagery. Digital topography, with preservation of x-y-z coordinates has not yet become common on mobile platforms. Despite small screen sizes, limited storage capacity, and sometimes expensive bandwidth limitations, handhelds have the power for interactive 3D viewing of digital topographic data.

We will discuss two types of elevation data, lidar point clouds and gridded DEMs. Our intent is to preserve the essential 3D elevation data, removing elements that are unlikely to be used on a handheld device, and to restrict the datum and projection allowed to reduce to computational load on the handheld. In this regard we follow in part the KML data standard (OGC, 2014), which requires all data to use geographic coordinates and the WGS84 datum. We use the JSON interchange format (ECMA, 2013), and to emphasize the geographic nature, we propose calling our format

GeoJSON. There is an existing GeoJSON format (Butler and others, 2008), but it applies only to 2D vector data. The 3D nature of point clouds and gridded DEMs suggest the proposed formats be called GeoJSON-P and GeoJSON-G.

Standardizing on a single, compact data structure allows supply of elevation data as a service, supply data over the internet and allowing the user to symbolize it intelligently and interact with the full 3D nature of the grids and points clouds. We will describe HTML5 programs, using Javascrpt and WebGL, for manipulating the elevation data.

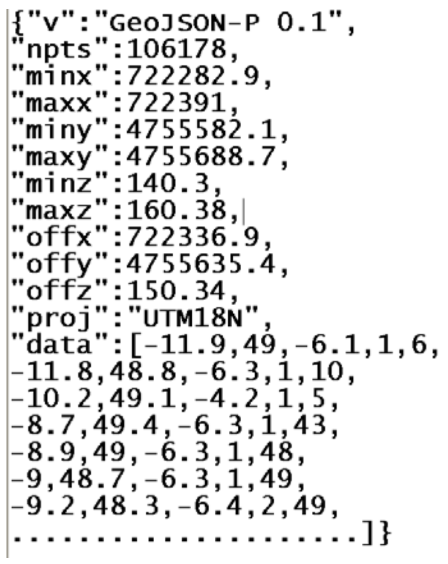
LIDAR POINT CLOUDS

Graham (2012) provided a thoughtful history of the philosophy behind the LAS Specification (ASPRS, 2011). He indicated that the LAS specification initially was meant as a transport format between lidar sensors and lidar processing tools, and was not intended for ultimate end-users. In the absence of viable alternative formats, however, LAS became a de facto standard for all parts of the lidar collection, processing, and final display and manipulation, and its use has been extended to other kinds of point cloud data. To improve performance, two free compressors can losslessly compress LAS files to 7-25% of their size (Isenburg, 2011). The LASzip algorithm (Isenburg, 2011), is available in several libraries and as a stand alone executable accessible to other programs. ESRI (2014) released their own compressor EzLASwhich creates zlas files; while the program is free, the algorithms are currently proprietary. Outside of ESRI software other programs wanting to use EzLAS must completely decompress the zLAS file before usage.

Following Graham (2012), both LASzip and EzLAS faithfully preserve all information in the point cloud. This includes a number of fields important to process, such as GPS time, scan angle, and return number, but with very little utility for end users looking to visualize the point cloud. We adopt a different perspective, that an open standard for point clouds appropriate for end users, especially those on handheld devices, makes sense. These users should appreciate that they cannot recover the full information in the point cloud; if they wanted to do that, they would return to the original data, and process it on a more appropriate platform.

Figure 1 shows our proposed format for GeoJSON-P for point cloud data. The header variables are self describing, and their order is not fixed. The fields include the version of the format, so a reading program can verify the format of the data, limits of the data set in x-y-z, offsets in x-y-z, and the projection. We envision that only UTM data sets will be used, and the WGS84 datum is required so that viewers do not have to perform datum transformations. The x-y-z offsets reduce storage, and allow the offset coordinates to be fed directly to 3D viewing software which likes to center the point cloud on the coordinate origin. The data contains five values for each return, the x-y-z coordinates, the point classification, and the return intensity.

Table 1 compares the file storage of a small point cloud in the LAS format, the two compressed versions, the ASCII and BZIP versions of the GeoJSON-P. The BZIP GeoJSON-G compresses almost as well as the LAZ and eLAS (they are about 60% of the file size), but decompressing it opening it uses standard Javascript libraries and does not require porting the LAS libraries to the handheld device. Figure 3 shows that the compression results scale linearly with the number of points in the cloud.



```
{"v":"GeoJSON-P 0.1",
"npts":106178,
"minx":722282.9,
"maxx":722391,
"miny":4755582.1,
"maxy":4755688.7,
"minz":140.3,
"maxz":160.38,
"offx":722336.9,
"offy":4755635.4,
"offz":150.34,
"proj":"UTM18N",
"data":[-11.9,49,-6.1,1,6,
-11.8,48.8,-6.3,1,10,
-10.2,49.1,-4.2,1,5,
-8.7,49.4,-6.3,1,43,
-8.9,49,-6.3,1,48,
-9,48.7,-6.3,1,49,
-9.2,48.3,-6.4,2,49,
......................]}
```

Figure 1. Proposed GeoJSON-P (point cloud) format.

Table 1. File sizes for a lidar point cloud with 10,128 points

Format	File size (kb)
KML	1658
GeoJSON-P ASCII	305
LAS	199
KMZ	71
GeoJSON-P bz2	52
zLAS	33
LAZ	28

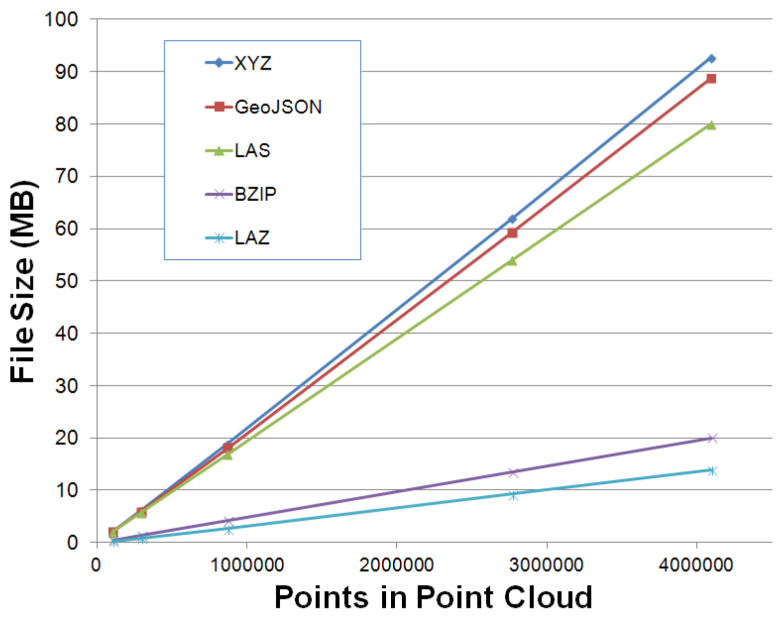


Figure 2. Scaling of file size with number of points in the point cloud.

GRIDDED DEMS

The information required for a DEM consists of the number of rows and columns, the coordinates of one corner, chosen as the southwest so the grid has a Cartesian coordinate system, the projection, and elevation range for color scaling. While most large scale DEMs will probably use the UTM projection, enough data such as NED, SRTM, and ASTER GDEM has a native geographic coordinate system to warrant an option to not reproject the data. Unlike reprojecting the lidar points, which is reversible, a reprojected DEM requires interpolating new data points. Display programs can easily scale the horizontal coordinates for an accurate map display. Figure 3 details the proposed GeoJSON-G grid format. Order of the fields is not important, but the display software must know the naming conventions.

```
{"v":"GeoJSON-G 0.1",
"w":1000,
"h":1000,
"minz":250.5,
"maxz":302.1,
"llx":582000,
"lly":4295001,
"dx":1,
"dy":1,
"proj":"UTM29N",
"data":[278.9,278.92,278.96,279.03,279.06,
..........]}
```

Figure 3. Proposed GeoJSON-G (grid) format.

Table 2. File sizes for a 1000x1000 floating point DEM

Format	File size (kb)
GeoJSON-G (ASCII)	6,720
Geotiff	3,911
Geotiff (zipped)	2,905
GeoJSON-G (bz2)	846

HTML5 DISPLAY OF DIGITAL TOPOGRAPHY

HTML-5 allows sophisticated graphics programming using Javascript, running inside a browser, and the exact same code can run on multiple platforms. Interactive 3D graphics requires WebGL (Khronos Group, 2014) to render graphics using the computer's graphics processing unit, called from Javascrpt. It requires a compatible browser, available for desktops, and available for Chrome and Firefox on handhelds. This notably excludes the iPhone and iPad, but includes Android phones and tablets.

WEBGL DISPLAY OF LIDAR POINT CLOUDS

We have adapted the XB-PointStream library (Salga and others, 2011) to display lidar point clouds with our GeoJSON-P format, compressed with bzip2 compression and decompressed by Javascript code on the browser (Antimatter15, 2013). Figure 3 shows the results on a tablet, and Figure 4 shows results from two different desktop browsers. Table 3 shows that a Nexus 7 tablet can outperform low end desktop machines in 3D manipulation of the point cloud. Figure 5 shows a larger scene, with symbolization both from the lidar return intensity and the point cloud classification.

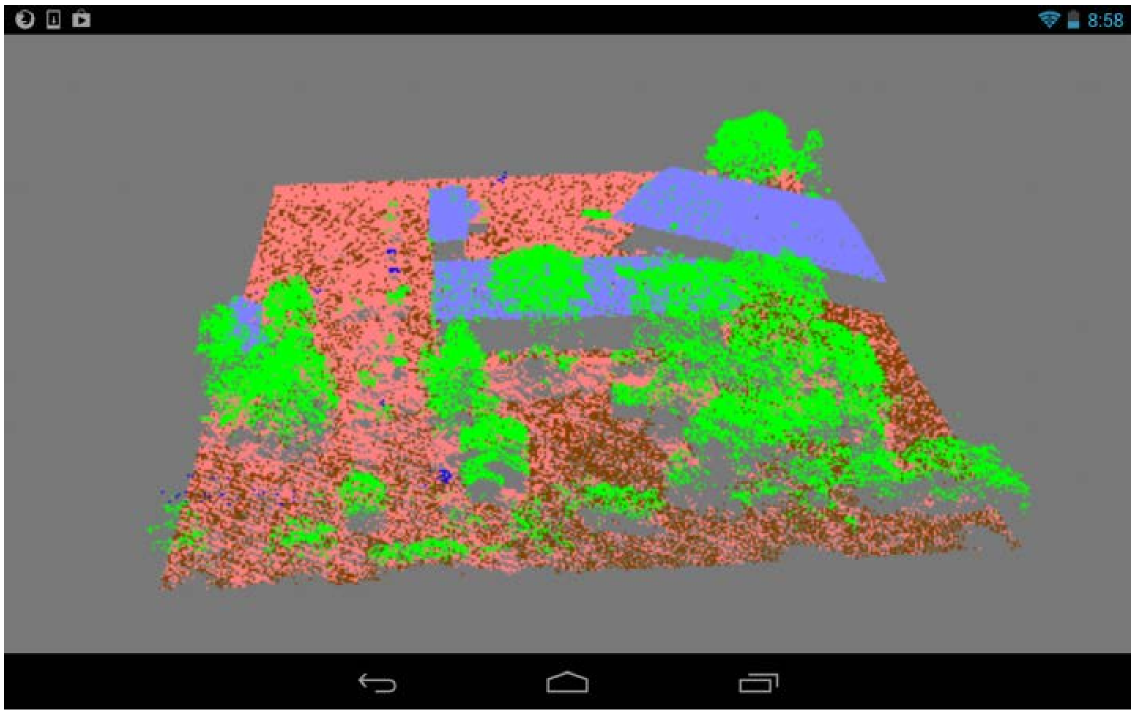
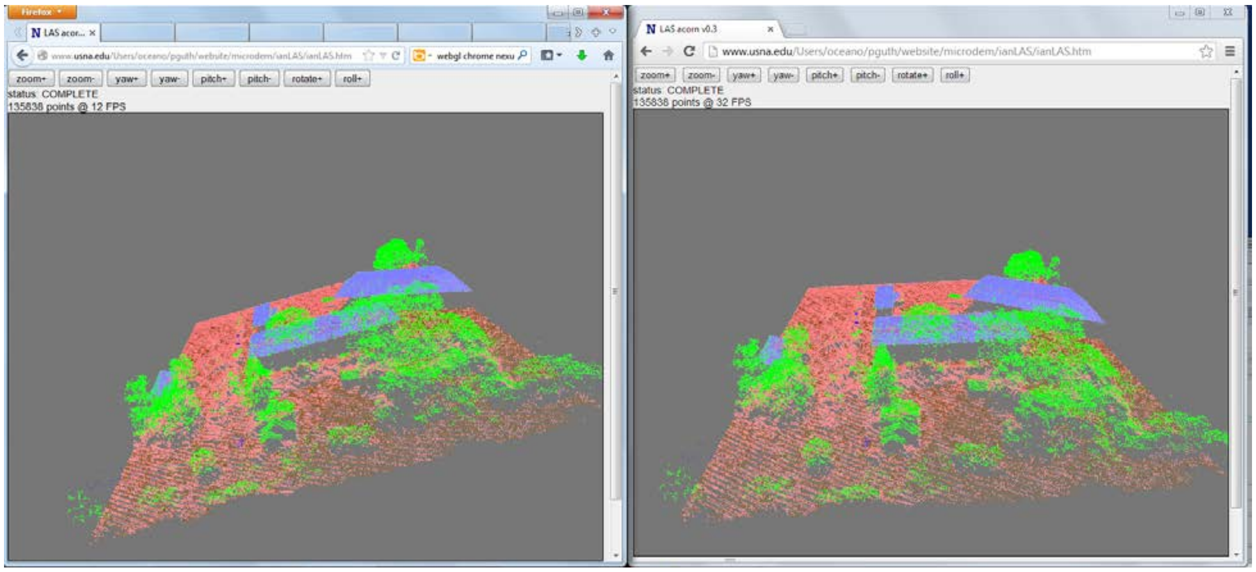


Figure 3. GeoJSON-P displayed in Chrome browser on Android Nexus 7 tablet.



```
status: COMPLETE
135838 points @ 12 FPS
```

```
status: COMPLETE
135838 points @ 32 FPS
```

Figure 4. GeoJSON-P displayed on desktop with Firefox (left) and Chrome (right) browsers.

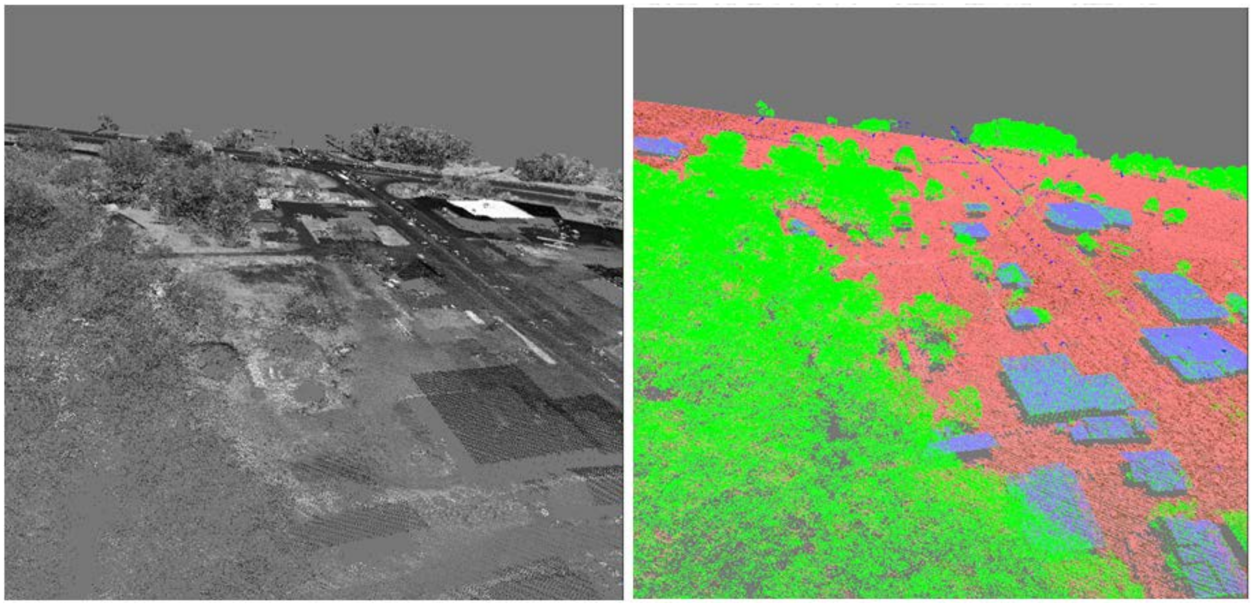
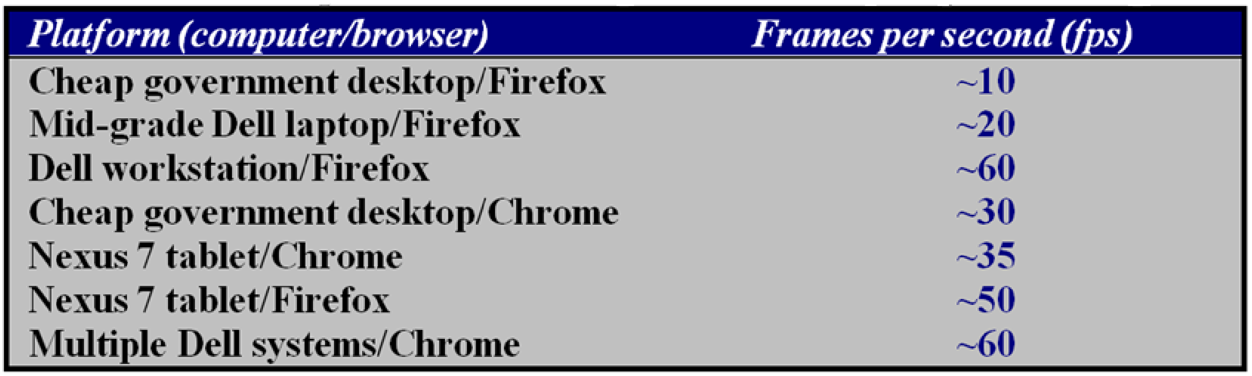


Figure 5. GeoJSON-P displayed with return intensity (left) and point classification (right).

Table 3. WebGL Performance with a small file



Platform (computer/browser)	*Frames per second (fps)*
Cheap government desktop/Firefox	~10
Mid-grade Dell laptop/Firefox	~20
Dell workstation/Firefox	~60
Cheap government desktop/Chrome	~30
Nexus 7 tablet/Chrome	~35
Nexus 7 tablet/Firefox	~50
Multiple Dell systems/Chrome	~60

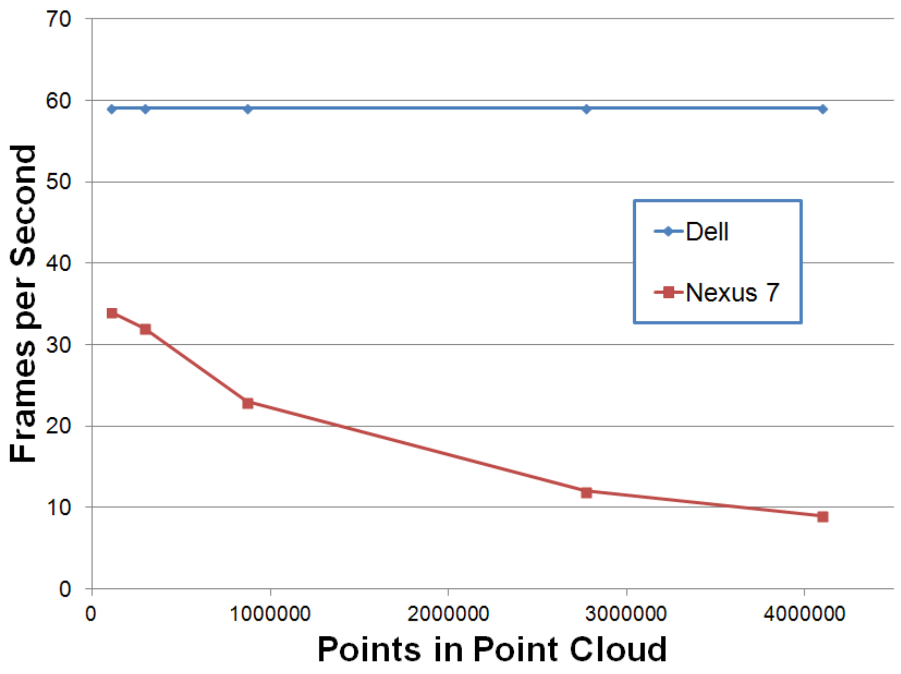


Figure 6. GeoJSON-P scene refresh as point cloud size increases, on Dell workstation and Nexus 7 tablet.

WEBGL DISPLAY OF GRIDDED DEMS

We have extended the html5-dem-viewer project (Vieira, 2013) to use the GeoJSON-G format. Figure 7 shows the 2D map display, which runs on almost all browsers, including iOS on the iPhone and iPad. Roaming on the display shows the current location's coordinates and elevation. The 3D display in Figure 8 relies on WebGL to run on most desktops and Android devices.

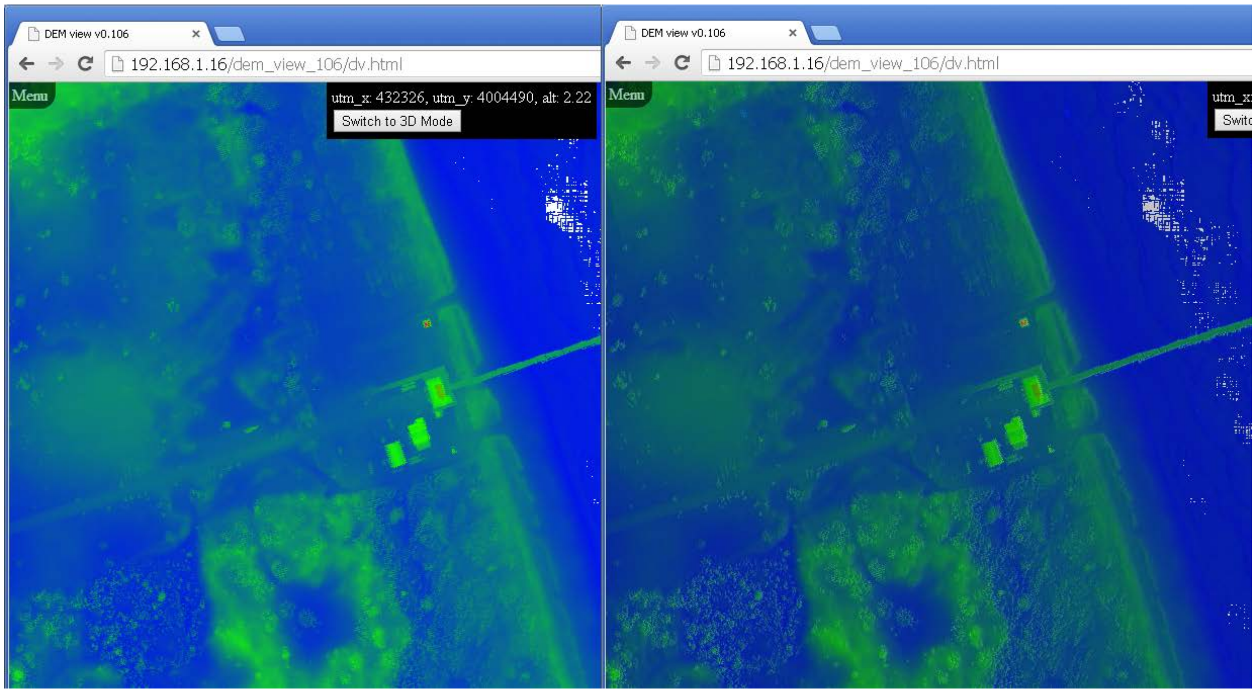

DEM view v0.106
192.168.1.16/dem_view_106/dv.html
Menu
utm_x: 432326, utm_y: 4004490, alt: 2.22
Switch to 3D Mode
DEM view v0.106
192.168.1.16/dem_view_106/dv.html
Menu

Figure 7. GeoJSON-G display of a 1 m gridded lidar dataset for Duck, North Carolina. The user has control of the elevation color ramp and reflectance shading.

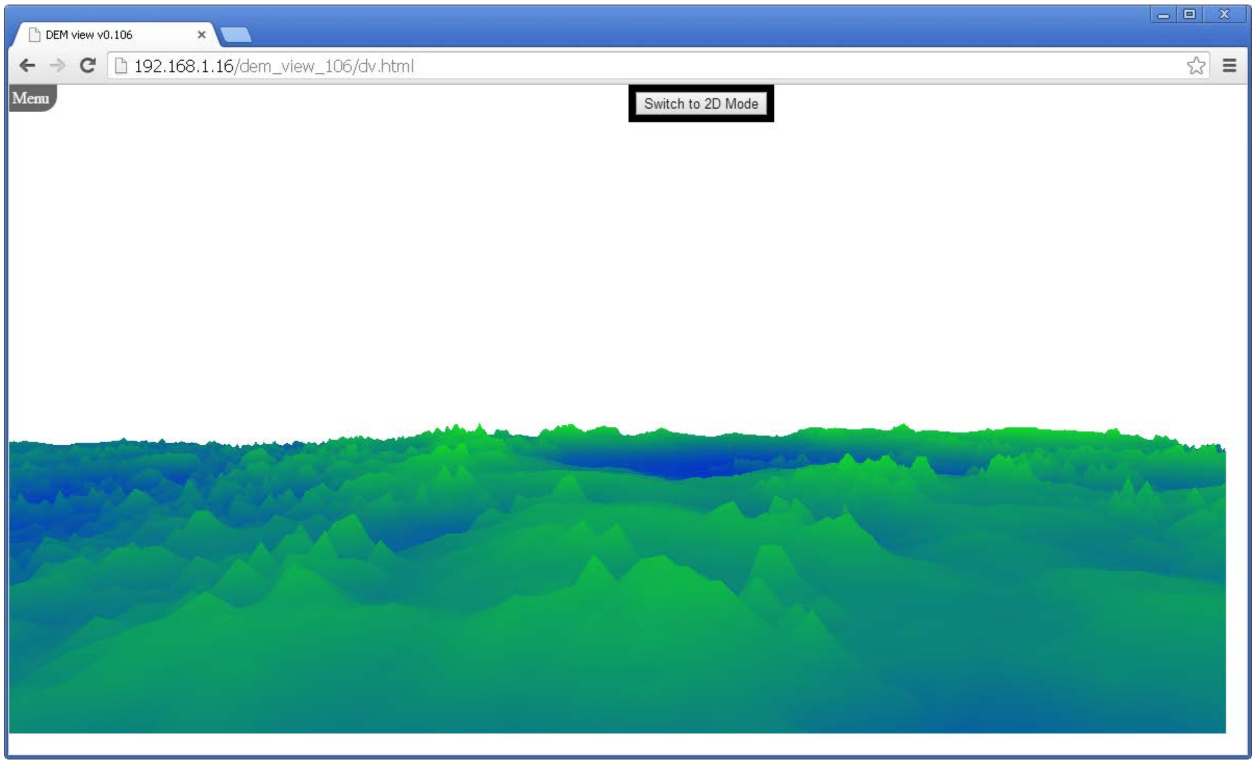


Figure 7. GeoJSON-G 3D display of a 1 m gridded lidar dataset for Duck, North Carolina

CREATION OF GEOJSON-P AND GEOJSON-G FILES

The simplicity of the formats means that creating data converts should be a simple process. To assist the process, we have exporters in MICRODEM (Guth, 2009) that will convert any LAS file or DEM into GeoJSON. The DEM conversion uses GDAL (2014) to convert DEM formats not read natively by MICRODEM. The conversions can be done in batch mode via a command line invocation of MICRODEM.

DISCUSSION AND CONCLUSION

As our results indicate, handheld devices have the capabilities to display 3D renditions of digital topography provided the limitations of the devices are recognized. Table 1 also include the file sizes for thelidar point cloud exported to KML and then compressed as a KMZ file. Google Earth can display this file, on both the desktop (Fiugre 8) and Android handhelds, but even the tiny size of this file brings the viewer to a crawl.

This proposal for GeoJSON standards for gridded DEMs and lidar point clouds is not intended to reopen or prolong the current lidar format wars, which center on the need for a proprietary format from one vendor which essentially duplicates an open source standard. Justification for such a format might be justified because a format directly tied to software implementation can probably operate more efficiently, but will also be attacked for espousing the “not invented here” mindset.

The GeoJSON formats recognize that end users, particularly on handheld devices, have very different requirements from the data producers and data providers. They want a universal format that will satisfy all customers, which the LAS format (ASPRS, 2011) does, and that can be efficiently compressed and uncompressed, which the LAZ format (Isenburg 2013) does. The wide variety of sites hosting LAZ files attests to the success of this (e.g. MNGEO, 2014; NOAA, 2014; Open Topography, 2014; William and Mary, 2014), but they have been geared to hosting data for GIS professionals. As lidar moves to wider use, simpler formats that retain the important parts of the data, and are immediately available for display and manipulation, will need to be adopted. Use of a single open standard would allow the data providers to supply point clouds as a service in addition to the full archival data sets.

Guth (2013) argued that the lidar point cloud represents a statistical sampling of the vegetation canopy present at a moment in time, and that many users could accept extreme lossy compression provided that it displayed a visually similar result. Just as MP3 music, movie DVDs, an JPEG photographs accept lossy compression to reduce file size, the lidar community should consider lossy compression and provide standards for efficient transfer and manipulation of the compressed files. The GeoJSON attacks the lossy compression by saving only the most important properties of the point cloud. It can lose some positional precision in the number of decimals used to store coordinates, which affects the file sizes.

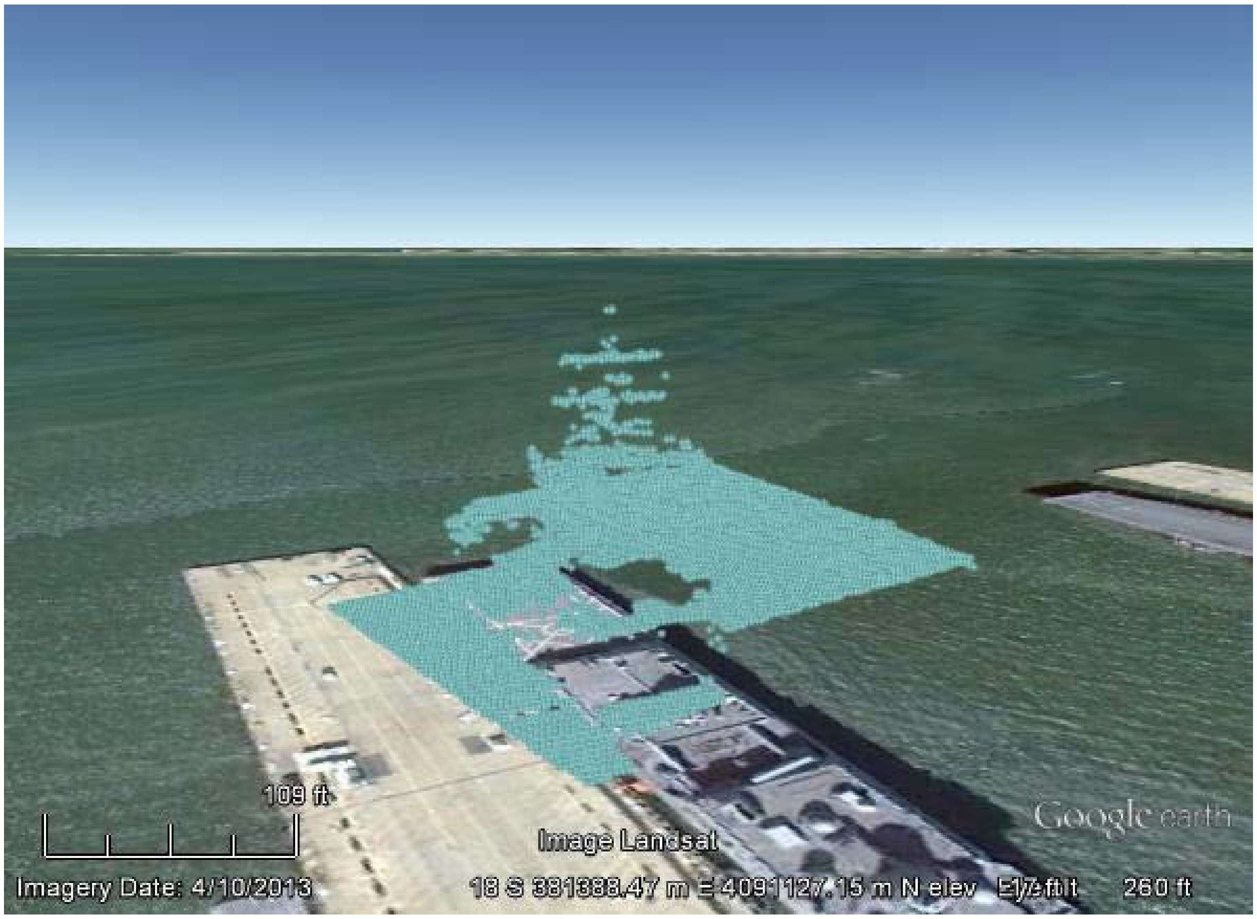


Figure 8. Point cloud with 10,128 returns displayed in Google Earth.

ACKNOWLEDGMENTS

We thank the creators of the open source packages we have leveraged to create this format and the HTML5 viewing software. The viewing software is available at http://www.usna.edu/Users/oceano/pguth/webmap/index.html.

REFERENCES

Antimatter15, 2013, bzip2.js: https://github.com/antimatter15/bzip2.js, last accessed 16 Feb 2014.

ASPRS, 2011, LAS specification Version 1.4—R12: http://www.asprs.org/a/society/committees/standards /LAS_1_4_r13.pdf , last accessed 16 February 2014.

Butler, H., Daly, M., Doyle, A., Gillies, S., Schaub, T., and Schmidt, C., 2008, The GoeJSON format specification: http://geojson.org/geojson-spec.html, last accessed 16 Feb 2014.

ECMA, 2013, The JSON interchange format: http://www.ecma-international.org/publications/files/ECMA-ST/ECMA-404.pdf, last accessed 16 Feb 2014.

ESRI, 2014, LAS optimizer (v.1.1): http://www.arcgis.com/home/item.html? id=787794cdbd384261bc9bf99a860a374f, last accessed 16 Feb 2014.

GDAL, 2014, GDAL—Geospatial data abstraction library: http://www.gdal.org/ , last accessed 17 Feb 2014.

Graham, L., 2012, The LAS 1.4 specification: *Photogrammetric Engineering & Remote Sensing,* 78(2):93-102.

Guth, P.L., 2009, Geomorphometry in MICRODEM, In Hengl, T., Reuter, H.I. (eds), Geomorphometry: concepts, software, applications. Developments in Soil Science Series, Elsevier, p.351-366.

Guth, P.L., 2013, Lossy compression of lidar point clouds: ASPRS 2013 Conference, Baltimore, MD, March 23-26, 2013, 9 page paper on conference proceedings web site.

Isenburg, M., 2013, LASzip: lossless compression of lidar data: *Photogrammetric Engineering & Remote Sensing,* 79(2):209-217.

Khronos Group, 2014, WebGL: OpenGL ES 2.0 for the web: http://www.khronos.org/webgl/ , last accessed 16 February 2014.

MNGEO, 2014, Lidar elevation data for Minnesota: http://www.mngeo.state.mn.us/chouse/elevation/lidar.html , last accessed 17 Feb 2014.

NOAA, 2014, Digital coast: http://csc.noaa.gov/digitalcoast/dataregistry/#/coastallidar, last accessed 17 Feb 2014.

OGC, 2014, KML: http://www.opengeospatial.org/standards/kml , last accessed 16 Feb 2014.

OpenTopgraphy, 2014, OpenTopography: A portal to high-resolution topography data and tools: http://www.opentopography.org/, last accessed 4 February 2013.

Salga, A., Medel, M., Leung, C., and Humphrey, D., 2011, XB-PointStream: https://github.com/asalga/XB-PointStream, last accessed 16 Feb 2014.

Vieira, F.S., 2013, html5-dem-viewer: https://github.com/fsvieira/html5-dem-viewer, last accessed 17 Feb 2014.

William and Mary, 2014, Virginia lidar: http://www.wm.edu/as/cga/VALIDAR/, last accessed 17 Feb 2014.