Shrinking Big Data: A Novel Compression Algorithm for LIDAR Elevation Maps

Jack D. Carson
Booker T. Washington High School, Tulsa, Oklahoma

Problem

Image courtesy of Velodyne LIDAR

Recent years have seen an increasing use of LIDAR, a method of capturing the depth of surrounding terrain by latency in laser pulses, to facilitate automation technologies such as Automatic Ground Collision Avoidance for aircraft (Auto-GCAS) autonomous vehicles, and robotics. These use cases, which have become some of the most exciting work in academia, supplement the traditional usage of LIDAR as a technology facilitating precision agriculture, surveying, and land management. Since its inception, the detection method has provided unparalleled accuracy and data abundance, which has been considered heavily in improvements to renewable energy, infrastructure develop-ment, and earth sciences. Due to the ultra-high resolution and unstructured data, file sizes are unmanageable (well within the gigabyte and terabyte range for even small regions). , Hence, large-scale systems of Google Waymo, Boston Dynamics, and Lockheed Martin haven chosen to create LIDAR point clouds in real time with mounted sensors mapping a three-dimensional environment for the vehicle as it travels through space.

The system is not without compromise, however. LIDAR has been limited in its accessibility due to the great cost of high-quality sensors and unpredictable reliability of concurrently generated data. Pre-analyzed data is a proposed solution to this problem; yet, as improvements in digital telemetry begin to slow, the task is now one of data encoding and digital compression to facilitate the vast data transfers required for autonomous, life-saving technology.

Objective

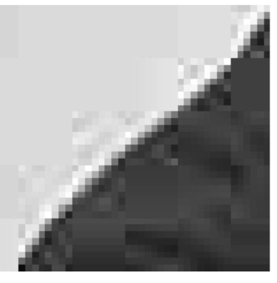
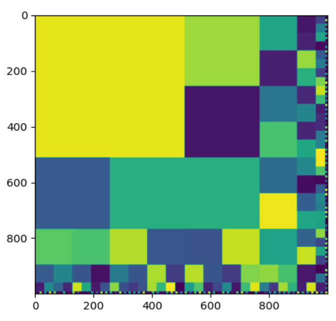


Image codecs largely rely on the tolerance of the human eye to mask imperfections within an image. JPEG, for example, uses the intersection of cosine waves to model data, a notion that leads to unbelievable compresison, but fails at closer inspection. Examine a black and white edge that undergoes a high efficiency DCT transform (JPEG)

The codec has obscured the edge and introduced uncertainty into the raster. For real world geospatial scans, this errant data (artifacting) has destroyed the integrity of the raster and cannot be used for autonomous decision making. **A novel codec must be developed that can compete with high efficiency transforms, while programmatically ensuring no artifacts. This board presents a novel algorithm, General Purpose Geospatial Compression (GPGC), that fills this objective.**

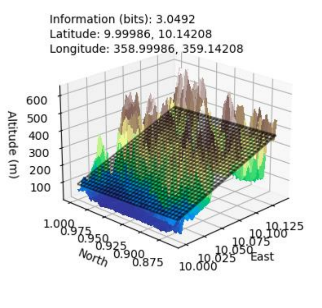
Encoding Procedures

Preprocessing



GPGC relies on data to be in a consistent, square, and 2^n size to allow the most efficient encoding and quadrisections possible. There-fore, data is preprocessed into a mosaic of 2^n sized fragments that fill bands radiating from the top left origin of the raster. Each fragment in the mosaic is compressed individually and encoded as a separate tree structure and stitched together in the decoding step.

Linear Regression Analysis



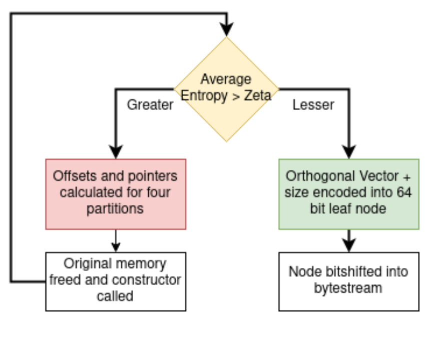
Within each fragment a block of data is passed by pointer to form a partition object. Within this, a regression plane is fitted with a LU composition operation performed on a 3 x n matrix formed using the block, solving for a orthogonal vector x that forms the plane. Here you can visually see a meshed geospatial datablock that has been fitted a least squares regression plane

The I, j, k vector components of the LSRP are stored as 16 bit IEEE754 Half Precision Floats in 64-bit leaf node structure gpgc_vector. From the plane, each differential is evaluated and accumulated into an unsigned long which is distributed across each point in the block. However, instead of evaluating the pure differential (as GEDACS and ECW do), GPGC calculates a z-score for each point based on a supplied compression parameter sigma, representing the standard deviation of acceptable errors. From this, the integral of probability density function to z-score is calculated, and the information entropy of each point is calculated as a function of the probability

$$\mathrm{P(C}_{mn}) = \frac{1}{\sqrt{2\pi}} \int_{-\infty}^{z} e^{\frac{-x^2}{2}}\, dx \qquad \mathrm{H(C)} = \sum_{n=1}^{M} \sum_{m=1}^{M} \frac{\log_2 P(C_{mn})}{M^2}.$$

Recursive Quad Tree Generation

Average information entropy per point evaluated with above formulae and compared to compression parameter zeta. If entropy is greater than the threshold allowed, the offset and pointer for the block of memory to be allocated and a 4 new partitions are created with size 2^{n-1} where the original had size 2^n. Hence lay the beauty of the above mosaicing system. This recursive pattern is also helpful as it ensures that once the maximum depth is reached, the data will be organized in a depth first search, an aspect of the encoding that is essential for the decoding process. If the entropy is found to be within the acceptable range zeta. The memory holding the structure gpgc_vector is cast to an integer and bitshifted into the bytestream, writing the resultant file in a dense binary of encoded partitions. The tree structure depicted below is an illustration of the hierarchical relationship of nodes that forms the basis for the codec. Similarly the decision and recursion are visualized in a decision flow chart



Average Entropy > Zeta

Greater

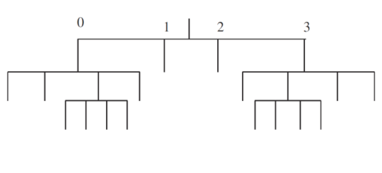
Lesser

Offsets and pointers calculated for four partitions

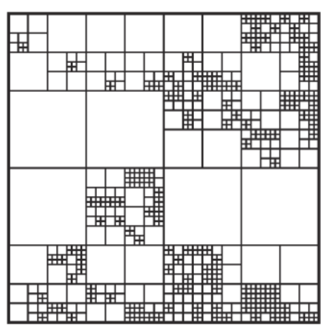
Original memory freed and constructor called

Orthogonal Vector + size encoded into 64 bit leaf node

Node bitshifted into bytestream



Decoding



The GPGC decoder relies on a depth first search of the above quad tree structure. It unpacks each element into three 16-bit floats, following the IEEE754 Half Precision float specification that is implemented in a submodule, as well as a 16-bit integer repres-enting the base 2 log of the size. The project takes a novel approach to a depth first search of an unknown-size quad tree, relying on a stack-based model. When a new size is added, it is placed at the top of the stack and substacks are created within it that accumulate the size of leaf nodes until the square of their size is met. This algorithm is able to determine the offsets for each leaf node in a raster. The concept of this decoder is visualized in the figure above. From here, it is simple to create a raster with the orthogonal vectors. The orthogonal vectors are multiplied by the index within the partition follow ix+jy+k = z to calculate each point. The decoder runs in O(n) time, superceding existing methods of quad-tree decoding such as that of GEDACS and some JPEG2000 implementations

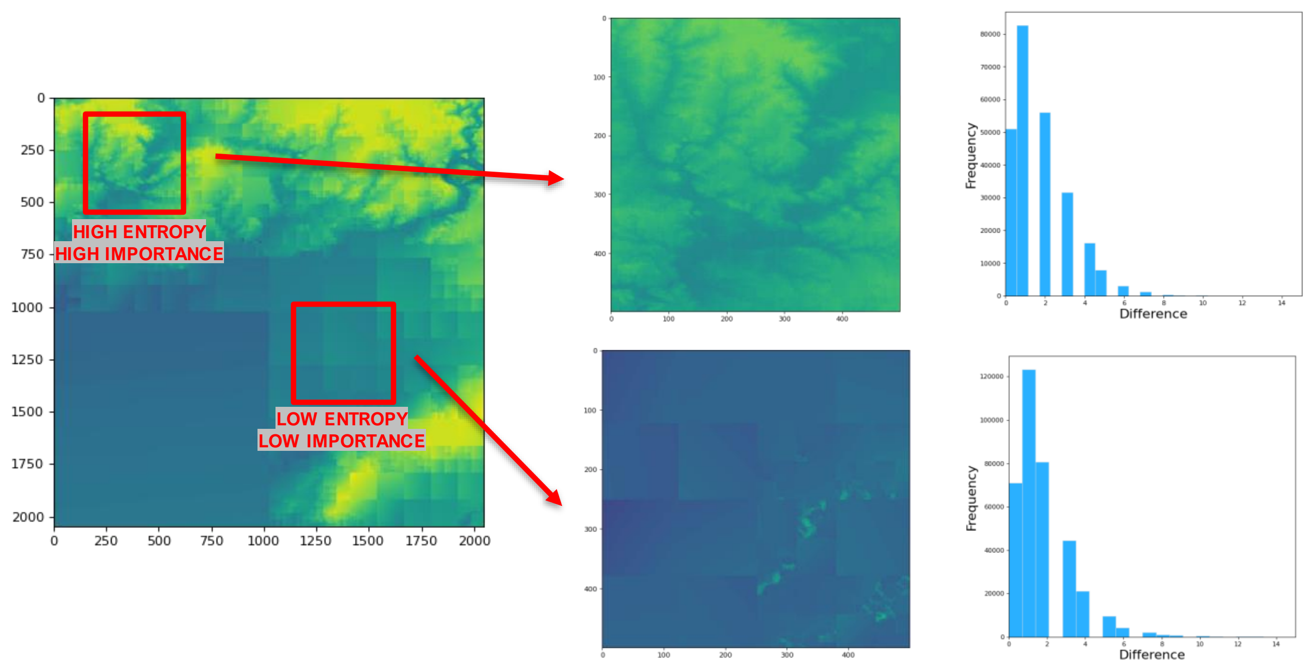
G P
G C
General Purpose
Geospatial Compression

Entropy Encoding

The algorithm defined in the left panel, GPGC, fundamentally differs from similar encoding schemes by its consideration of information entropy. Information entropy is a measure of the "uncertainty" or "unpredictability" of a dataset. The use of information entropy has been a critical mathematical concept in telecommunications and signal processing, specifically in filtering out meaningful data from noise. The operator of entropy represents, literally, the calculation of the minimum number of bits it would take to represent some amount of data. **Fundamentally, it is used within this algorithm to process data in a very human, intelligent way**. Qualitatively, if you were flying an aircraft, you can think of the landscape around you in terms of the concept. If you are flying over the Oklahoma prairie, you can be fairly confident that topography will not immediately change in a significant way. Hence, the topography has a very low information entropy. Because the data is predictable and unchanging, each point defining has a lower importance, and can be compressed more aggressively. If a sudden, highly unlikely shift did occur, it would increase the entropy of the region, and any accurate algorithm would have to consider that change and increase the resolution accordingly. Around a mountain, or near a valley, where data is much less predictable, a pilot needs more accurate, high-resolution information about the topography of the region to plot a safe course over the terrain.

The partitioning entropy evaluation of GPGC excels at making these distinctions and controls the resolution of the data via the variable-size partitions that fill the data set decoded from the encoded hierarchical tree structure. In mountainous, or high-entropy areas, it achieves nearly pixel-perfect accuracy by stitching very small partitions together. While, in lower-entropy areas, it tolerates the loss of data to skip over smaller, less important features. However, the sensitivity of the algorithm can be tailored depending on the application and is currently implemented as two runtime flags to the executable. The difference between the compression of high and low entropy areas can be visualized in the figure below. Note: the figure is not at a acceptable resolution for general use in order to improve the visualization of partitioning at high versus low entropy regions.

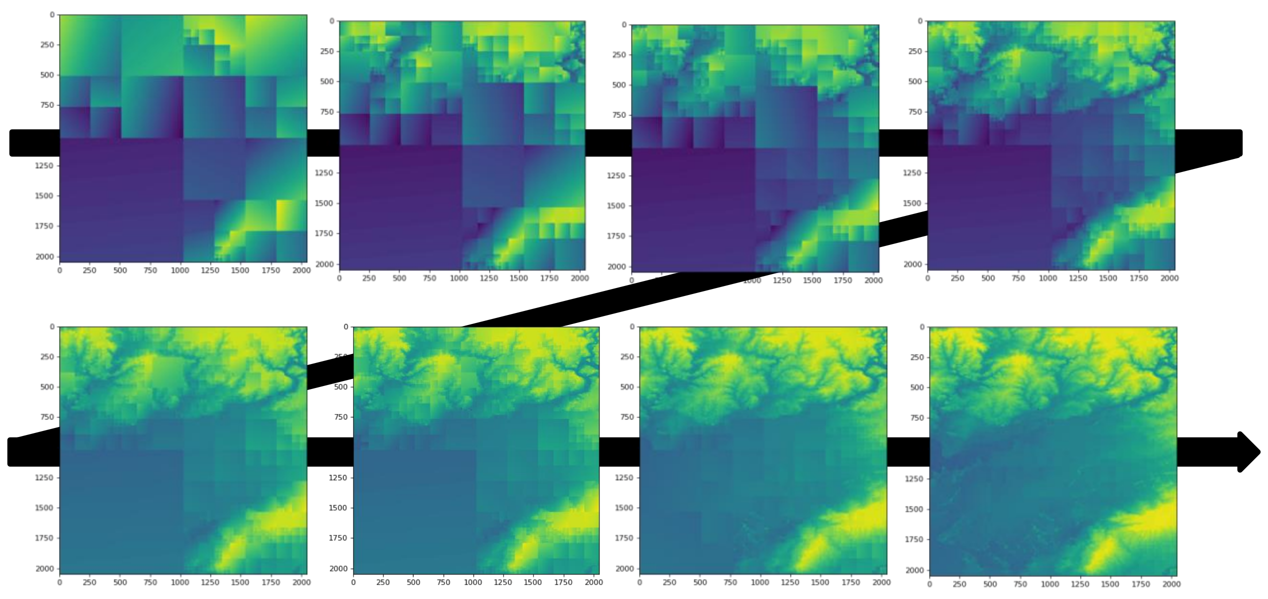
This model differs from JPEG, which partitions all data into 8 x 8 chunks making up the resolution of the image. From these it calculates the Discrete Cosine Transform and can encode them with extremely minimal information, yet unpredictable loss. This model works well for images, where every part of an image can be considered as holding relatively equal information. However, geospatial data benefits from certain predictable patterns as mentioned above and can thus be entropy coded more efficiently. Note, the algorithm would be even more efficient if the DCT was generated for all partitions. However, as delineated in the Objective slide, unpredictable transforms such as DCT are unfit for real-world LIDAR data.



We can inspect both visually and quantitatively the quality of this compression. For the high entropy region, the data is visually lossless. While we *can* inspect the partition borders in the low entropy region, where the compression has been more aggressive, the differences between each border is quite small and inessential for autonomous systems. Still, GPGC has done an outstanding job preserving the small terrain features that appear in a slight green.

Compression Snapshots

GPGC's set partition code is a refining algorithm. It makes no assumptions about the number of partitions that it will take to compress a raster to the desired entropy threshold: a key difference from related partitioning algorithms such as JPEG2000. A simple way to understand what exactly is happening experimentally is to visualize the compression while it is happening. The following images are in order, showing the process of partitioning the raster from large square blocks to visually lossless data.



Consider that the encoding process does not happen linearly, we can see how it immediately focuses its efforts to compressing the high information mountains, while letting the accuracy of the plains below fall behind.

Encoding Schema

0001110010101010101000101110111001011100101010001000101110111 0

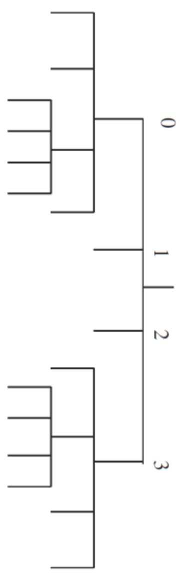
i-hat vector component (16-bit float)

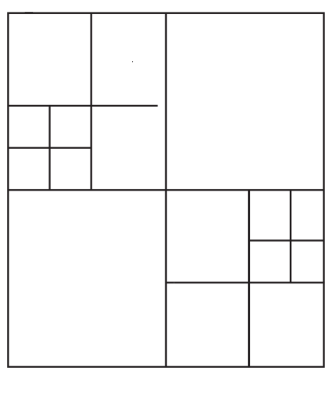
j-hat vector component (16-bit float)

k-hat vector component (16-bit int)

log2(n) (16-bit int)

```
 0.607422   0.0917969 210  16
-0.107117   1.51172   216  16
-0.550293   1.33594   216  8
-0.053558   1.44629   211  8
-0.367432   1.84668   228  8
-0.767578   2.10938   224  8
 0.740723   0.672363  234  8
-0.550293   1.33594   219  16
 0.34668    0.679688  243  32
 0.916504  -0.532715  230  32
 2.17188   -0.710938  238  16
-0.604492   0.0419006 244  8
 0.380371   0.718262  224  8
 0.194214   0.801758  224  8
 0.284424   0.812988  241  8
-0.380127   0.404785  250  16
 0.081665   0.337891  240  16
```





Testing

GPGC is compared against a total of seven algorithms. Three geospatial algorithms are identified: GEDACS (NASA), MrSID (Los Alamos National Laboratory), ECW (Hexagon AB). Three binary codecs are identified: ZSTD (Facebook), PACKBITS (Apple), and Deflate (aka. Lempel-Ziv 77) (MIT). Finally, GPGC is compared against the JPEG lossy image algorithm, seen as the golden standard for extremely high efficiency compression, although the reasons for its unsuitable use have been explored thoroughly in the presentation.

Using the GDAL CLI and the proprietary MSVC solution for GEDACS, each possible codec was implemented onto 135 randomly selected United States Geographical Survey Continental 1-arc second rasters (12,967,201 points per raster) captured from the SRTM project. This dataset was chosen due to its unprecedented reliability, and the simplicity of the existing encoding, ensuring as many of the codecs could be tested as possible. Testing was done inside of a Jupyter IPython notebook where each file was evaluated for compression ratio and mean average error.

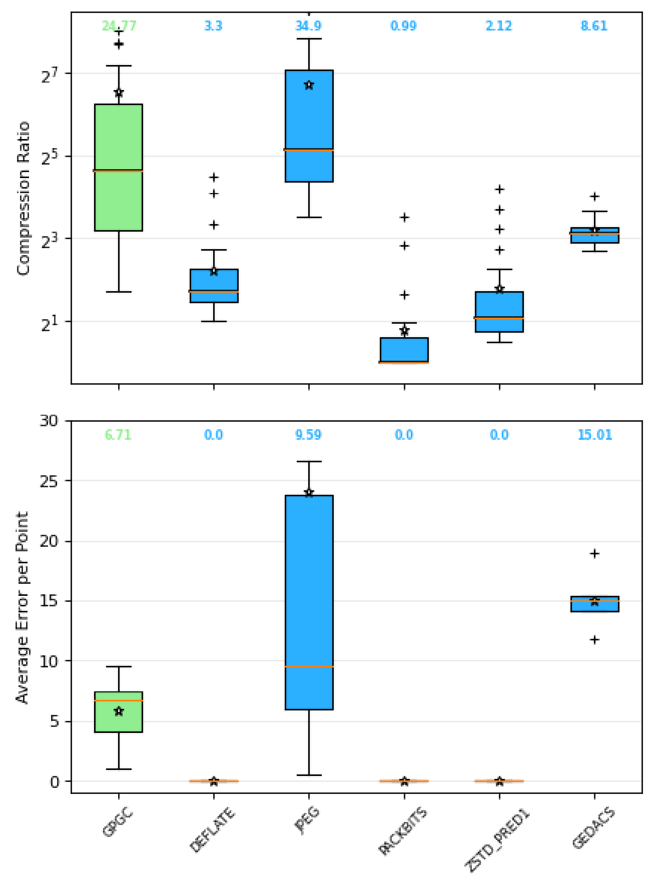
Results

Compression Benchmarks

Compression Algorithm	Comp. Ratio 25%	Comp. Ratio Med.	Comp. Ratio 75%	Percent Decrease	Average Error
GPGC	**8.2**	**24.7**	**72.8**	**95.97%**	**6.71**
GEDACS	7.7	8.48	9.2	88.21%	15.7
MrSID	4*	n/d*	10*	n/d*	0
ECW	n/d*	10*	15*	90%*	n/d
ZSTD PRED1	1.7	2.12	4	52.12%	0

* = Closed-source benchmarks provided by proprietors. Unable to validate independently.

Compared by compression ratio and mean average error (MAE), GPGC significantly outperforms its direct competitors. The median compression ratio is nearly triple that of GEDACS, a similar NASA algorithm for geospatial rasters. The compression ratio is more than double that of ECW, a proprietary format owned by Hexagon AC for their autonomous vehicle project. Note the median compression ratio for ECW has not been quantified in any peer-reviewed articles and is cited only from information that Hexagon itself provides.



This box chart only examines algorithms which could be independently tested; ECW and MrSID are not included as their proprietary encoding applications could not be used for exact comparisons. As the box plot suggests, **GPGC compresses data at a greater accuracy than any other lossy algorithm, while being comparable the extreme compression ratios of JPEG**. Note, the wide IQR of JPEG and GPGC draws from their use of entropy coding, with each algorithm being highly sensitive to the type of data it is compressing.

Implementation

I have been funded by the international research consortium VAIL in partnership with NASA and the FAA to implement the GPGC algorithm into a new open-source automated ground collision avoidance system (Auto-GCAS) for small civilian aircraft like those deployed on American fighter aircraft. The algorithm has been largely used to supplant the GEDACS implementation

As implemented, the novel method has shown exceptional results for aircraft encoding. It benefits largely from its simplicity and reliability, with a planar system that makes it intuitive to calculate collision vectors without any additional assumptions or checks.

VAIL
Vehicle Autonomy and Intelligence Lab
Image courtesy of VAIL

Furthermore, the project is attempting to create a specification for GDAL and QGIS, the two most popular open source applications for interacting with geospatial data, allowing data to be read, analyzed, and written with the same ease as any other format such as GeoTIFF. Finally, if possible, steps have been taken to modify GPGC slightly to allow for implicit encoding into TIFF files, making the compression extremely accessible without the need for any additional binary formats.

Conclusion

Over the large testing data set, GPGC has shown to supersede its nearest competitor GEDACS in compression ratio, accuracy, encoding speed, and decoding speed. Although testing was not possible for ECW and MrSID, it encodes data with an even greater efficiency than their advertised benchmarks. Furthermore, the deterministic planar encoding system ensures that there are no flagrant artifacts in the encoded rasters. As such, all of the engineering objectives have been exceeded.

- A method for partitioning the data was devised through the creation of a recursive quad-tree structure.
- Considerations of entropy allowed for intelligent encoding of raster data.
- Data was encoded in such a way that it was guaranteed to never produce artifacting.
- The quad tree structure was decoded using a novel depth-first-search algorithm specifically designed for GPGC
- The codec was evaluated compared to similar and competing algorithms.

Future Research

Possibly the most exciting part of GPGC is the massive room it still leaves for improvement. The provided implementation does its best to maximize efficiency, but there are several entropy coding steps that were omitted from the method and could be the subject of future research. Immediately, the mosaicing system stands out as being a particularly inefficient method for allowing odd-number divisions. There is much room to allow minimal encoding strain while not also forcing data into unnecessarily small bands.

Possibly countering the inefficiency of the band solution for mosaicing, geospatial compression as described would be benefited enormously from a Huffman Coding system post processed into the quad tree. The Huffman Code allows for more efficient encoding of structures by representing leaf nodes as pointers to another binary tree structure