Efficient Terrain Heightmap Compression

Shaye Garg

***Abstract*— This paper presents a simple and efficient technique for compressing terrain heightmap data, *'hcomp'*, by specializing image compression techniques for use with heightmaps. Both compression and decompression retain high-performance characteristics, suitable for use in real-time applications.**

I. Introduction

Terrain heightmaps are widely used in various areas of computing. They are used for rendering 3D terrain in games and other visualizations, and used to render relief and contour maps, depicting the topography of the Earth [1, 2]. Due to the prevalence of maps, heightmaps are frequently used in constrained environments, with low memory, storage, and bandwidth, such as mobile phones. In the case of games, heightmaps can be a large part of the installed size, depending on required visual fidelity, reaching upwards of 50 GB in some cases. Thus, an optimal algorithm for compressing heightmaps with high decoding and encoding performance can have many benefits. This paper presents a simple algorithm that has high compression ratios and fast compression and decompression. The reference implementation is implemented in less than 500 lines of Rust code and can compress data with no training or dictionary generation required.

II. Overview

The algorithm consists of a sequential series of passes that modify the input data, reducing the entropy of the data in the process. Running these passes in reverse decompresses the data. The input for compression is an image where each pixel is one 16-bit unsigned integer representing the height of the pixel from some constant reference position. The first pass is a prediction pass. The value of each pixel is predicted using surrounding, already decoded pixels, and only the delta of the actual value from this value is stored. If these values fit within 8 bits, they are all stored as 8-bit integers. If not, then a palette generation pass is run. If the number of unique deltas allows for, a palette of deltas is created, and the image is transformed such that each pixel has an 8-bit index into the palette. Finally, an entropy coding pass is run to exploit the reduction in entropy and compress the data.

III. Prediction

Prediction is the primary method of reducing entropy. It uses previously decompressed values in scanline order to predict the value of a neighboring value, storing only the delta in the compressed output. The higher the prediction accuracy, the smaller the stored deltas will be, further reducing the entropy in the output.

The first pixel must be stored as-is since there is no other data to predict its value. The pixels immediately to the right, and immediately to the bottom are predicted to be the same height as the first pixel. Then, the entire first row and first column are predicted using a linear predictor. The linear predictor takes 2 pixels and predicts the next one to lie on the same line as them – keeping the deltas constant. Finally, the rest of the image is predicted using a plane predictor. The plane predictor takes the pixels directly above, to the left, and diagonally top left of the target pixel, and predicts that the target pixel is on the same plane as them.

The prediction outputs an image of signed deltas, but these are not ideal, since the two's complement representation stores negative numbers with most bits in the most significant byte set. Having data distributed evenly between both bytes of the number can impact entropy coding, so instead, the minimum delta is stored separately, and the image is encoded as deltas from this minimum delta. This clusters the data in the least significant bits, improving entropy coding, and helping the next step, byte compression.

IV. Byte Compression

If all the values excluding the first pixel and minimum delta fit within one byte, the most significant byte of each pixel in the entire image is discarded, since it consists entirely of 0s. This has substantial gains because the entropy coding algorithm used works in one-byte units.

V. Palette Generation

If byte compression fails, palette generation is attempted. If there are 256 or fewer unique values in the prediction output, each pixel can instead become an 8-bit index into a palette. However, if there are lesser than 512 values in the heightmap, no palette is generated as there is a low chance of value repetition. To further reduce entropy, the palette has some extra processing done to it. Since the order of values in the palette doesn't matter, it is sorted, and each term is stored as a positive delta from the previous. Since the palette is deduplicated, the deltas cannot be 0, so 1 is also subtracted from each of the deltas stored.

VI. Entropy Coding

The Zstandard entropy coding algorithm to entropy code the preprocessed data. Zstandard uses both Huffman coding, and entropy coding based on asymmetric numeral systems [3, 4]. Since it works on 8-bit units, palette generation and byte compression can have a large impact on the final compression ratio. Since no metadata is stored, the size of

* Shaye Garg is with Head Start Educational Academy, Bangalore, India.

the output is used to determine the successful passes run on the dataset.

VII. Results

For testing, the ALOS Global Digital Surface Model (AW3D30) dataset, sourced from OpenTopography, was used. It has a resolution of 1 arcsecond per pixel, where each 1-degree by 1-degree tile on the Earth is divided into an image with a resolution of 3600 pixels by 3600 pixels. Each pixel stores a signed 16-bit height in meters from sea level. Thus, the raw size of each tile is 25.92 MiB. The tiles are stored in the GeoTIFF format, with LZW compression. To convert the signed values to the unsigned values *hcomp* requires, a value of 500 meters was added to each pixel, ensuring all values are positive.

Several widely used formats such as WebP, raw Zstandard, and XZ were used for comparison. The highest compression ratios were used for all algorithms.

Table I compares one of the largest tiles of the dataset, 'ALPSMLC30_N027E086_DSM.tif,' which represents the area starting from 27º N, 86º E. The terrain in this tile is highly varied, with mountains and valleys. *hcomp* comfortably beats all other algorithms in compression ratio, as well as compression speed.

Table II compares one of the smallest tiles of the dataset, 'ALPSMLC30_N000W030_DSM.tif,' which covers the area starting from 0º N, 30º W. This tile has very flat terrain, making it very easy to compress. However, WebP outperforms *hcomp* in compression ratio, due to the entropy coding algorithm's (Zstandard) output format metadata dominating the output size.

In general, decompression time also suffers compared to other algorithms. This is because the reference implementation is not as heavily optimized as the other algorithms, which are widely used, with a lot of work put into making them as fast as possible.

TABLE I. COMPARISION OF ALGORITHMS ON THE LARGEST TILE

Algorithm	**Absolute**				**Relative to *hcomp***		
	Size (MiB)	***Compress Time (s)***	***Decompress Time (s)***	***Compression Ratio***	***Size***	***Compress Time***	***Decompress Time***
hcomp	7.23	8.39	0.18	28%	-	-	-
WebP	9.47	41.53	0.12	37%	131%	495%	69%
ZStandard	14.01	11.74	0.11	54%	194%	140%	66%
XZ	11.85	13.52	0.75	46%	164%	161%	423%

TABLE II. COMPARISION OF ALGORITHMS ON THE SMALLEST TILE

Algorithm	**Absolute**				**Relative to *hcomp***		
	Size (KiB)	***Compress Time (s)***	***Decompress Time (s)***	***Compression Ratio***	***Size***	***Compress Time***	***Decompress Time***
hcomp	0.40	0.22	0.15	0.00%	-	-	-
WebP	0.30	3.87	0.01	0.00%	76%	1757%	8%
ZStandard	2.19	0.05	0.02	0.01%	537%	22%	13%
XZ	3.90	1.13	0.08	0.02%	957%	514%	52.4%

VIII. Further Work

hcomp doesn't perform very well for small input with low entropy, due to Zstandard's frame format. Zstandard also works on 8-bit units, instead of 16-bit. Replacing Zstandard with a custom entropy encoder working on 16-bit units can rid the need for byte compression and reduce the frame size.

Using a better predictor can also substantially improve compression. An example includes the ODETCOM model (overdetermined compression) [5]. However, this requires training on the dataset, while the current predictor works without any training required. Requiring training is a trade-off between the compression ratio and the compression performance, with little effect on decompression performance.

IX. References

[1] E. Yusov, and M. Shevtsov, "High-performance terrain rendering using hardware tessellation," Journal of WSCG, vol. 19, no. 1-3, pp. 85-92, 2011.

[2] B. Horn, "Hill shading and the reflectance map," Proceedings of the IEEE, vol. 69, pp. 14-47, 1981.

[3] D. Huffman, "A Method for the Construction of Minimum-Redundancy Codes," *Proceedings of the IRE*, vol. 40, pp. 1098-1101, 1952.

[4] J. Duda, "Asymmetric numeral systems: entropy coding combining speed of Huffman coding with compression rate of arithmetic coding," arXiv, 2013.

[5] M. Inanc, "Compressing Terrain Elevation Datasets," Ph.D. dissertation, Dept. Computer Sci., Rensselaer Polytechnic Institute, Troy, NY, 2008.