Compressing Web geodata for real-time environmental applications

Claudio Cavallaro **, Roman Fedorov *, Carlo Bernaschina *, and Piero Fraternali *

Politecnico di Milano,
Dipartimento di Elettronica, Infomazione e Bioingegneria
Piazza Leonardo da Vinci, 32, Milan, Italy
*{first.last}@polimi.it
**{first.last}@mail.polimi.it

Abstract. The advent of connected mobile devices has caused an unprecedented availability of geo-referenced user-generated content, which can be exploited for environment monitoring. In particular, Augmented Reality (AR) mobile applications can be designed to enable citizens collect observations, by overlaying relevant meta-data on their current view. This class of applications rely on multiple meta-data, which must be properly compressed for transmission and real-time usage. This paper presents a two-stage approach for the compression of Digital Elevation Model (DEM) data and geographic entities for a mountain environment monitoring mobile AR application. The proposed method is generic and could be applied to other types of geographical data.

Keywords: Data Compression, Augmented Reality, Environment Monitoring

1 Introduction

Outdoor augmented reality applications exploit the position and orientation sensors of mobile devices to estimate the location of the user and his/her device Field Of View (FOV) so as to overlay such view with information pertinent to the user's inferred interest and activity. These solutions hold the promise of engaging users to collect environmental data, exploiting the physical presence of the user in a position of interest and the possibility of sending his/her contextual, geo-referenced data to explain and support the data collection process.

A possible use case could be a mountain-specific Augmented Reality (AR) application [9], in which the user is steered towards collecting observations about specific mountains by overlaying his/her camera view with the identification, tracking distance, degree of interest, etc. of each mountain. The main challenges of outdoor mobile AR applications include providing an accurate estimation of

This work was partially funded by the CHEST FP7 project of the European Commission.

user's information like current position, FOV and delivering the most relevant contextual meta-data, adapted to the changing view in real-time. Commercial applications, which operate mainly in the tourism field, simplify the problem by estimating the user's context based only on the device position and orientation sensors, irrespective of the content actually in view. Examples are sky maps, which show the names of sky objects based on the GPS position and compass signal. These approaches may provide information that does not match well what the user is seeing, due to errors in the position and orientation estimation or to the presence of objects partially occluding the view. These limitations prevent the possibility for the AR application to create *augmented content*. If the overlay of the meta-data onto the view is imprecise, it is not possible for the user to save a copy of the augmented view, e.g., in the form of an image with captions and meta-data associated to each object in view. Such augmented content could be useful for several purposes: archiving the augmented outdoor experience and the collected data, indexing visual content for supporting search and retrieval of the annotated visual objects, and even for the extraction of semantic information from the augmented content for environment analysis purposes. Furthermore, outdoor mobile AR applications must support field work in areas with low or absent network connectivity; this requires supporting offline usage, by letting the user download the potentially relevant meta-data before the data collection campaign. This requirement further advocates for intelligent data pre-processing to minimize the wait time, data storage, and data transmission time and cost, which could discourage citizens from using the application.

This paper describes the approach to real-time data management for the *SnowWatch* [8, 2] outdoor mobile AR application, which supports mountain data collection campaigns, by contextualizing the user's view with the automatic overlay of geographical meta-data (peak name, altitude, distance from viewer, etc). Unlike other systems (e.g., PeakFinder[1]), SnowWatch exploits a content-based reality augmentation algorithm [10], which takes as input not only the position and orientation of the user's device but also the content of the current view and meta-data about the region of interest. Such meta-data derive from a Digital Elevation Model (DEM), which is a 3D representation of the Earth's surface, stored at the server side. First, the DEM, the position and the orientation of the user are exploited to estimate a bi-dimensional projection of the panorama that should be in sight and to match it to the image currently captured by the camera. Second, meta-data about mountain peaks are transferred from the DEM to the camera view and superimposed in real time to it, so that the user can save an augmented image that integrates the contextual meta-data and the captured image. The augmentation process is executed in real-time and requires a policy for meta-data compression, transmission, decompression and caching, compatible with the capacity of the mobile device and able of ensuring the fluidity of the user's experience. The contributions of the paper can be summarized as follows:

– We introduce the problem of reality augmentation, specifically for mountain environment data collection purposes, and highlight the challenges of geo-

[1] `www.peakfinder.org`

graphic meta-data management in a mobile AR context, in terms of latency, data transfer cost, and support to disconnected usage, with a specific focus on DEM data.
- We describe multiple encoding and compression pipelines, which take advantage of the semantics of DEM data, for addressing such challenges. The best approach yields a compression ratio of 5.46 in average conditions and of 11.26 in favorable conditions (few peaks).
- We illustrate the application of the algorithms to a mobile AR application that supports real-time peak identification and augmented content generation.
- We report on the preliminary results of evaluating alternative data compression solutions in real outdoor experimental conditions.

The rest of the paper is organized as follows: Section 2 overviews previous work in the areas of data compression, outdoor AR mobile applications, and environmental monitoring applications; Section 3 states the problem of data management for outdoor AR application and introduces alternative algorithms addressing the challenges of this class of applications; Section 4 evaluates the algorithms for real-time DEM data management in different conditions; Section 5 concludes by discussing the generalization of the algorithms designed for DEM data to other environmental data management problems, presents the outcome of using annotated mountain images for the resolution of a real-world environmental problem, and provides an outlook about the next research objectives.

2 Related Work

Data compression is a prominent task in data science, for which many techniques have been developed over the years. Recently, the need of managing data in real time (e.g, for online applications) shifted the focus from the pure compression ratio optimization also to the minimization of the compression/decompression time. The leading data representation method for compression purposes is Huffman coding, which forms the basis of most subsequent approaches [20]. Compression techniques can be broadly classified in four major categories: derivatives of Lempel-Ziv-Welch [23], approaches based on statistical model prediction [4], on characters permutations [1], and on arithmetic coding [16]. In the case of DEM data, the seminal work [12] advocated the possibility of reducing data size with an initial data simplification stage, followed by compression with Huffman coding. Along this line, the work in [22] applied 3-points prediction, which yielded up to 80% compression performance improvements; [14] applied an 8-points prediction, based on the Lagrange multipliers method, reaching 4% compression performance improvements in comparison to [22]; [13] used a 24-points prediction algorithm in the pre-processing phase, achieving 40-60% improvement in comparison to gzip [6]. On the performance side, the work in [18] compares execution time of several lossless image compression algorithms, and applies them DEM data. Compression ratios and execution times for data in different geographic areas are collected and the optimal solution exploits a PNG compression method that supplements a deflate algorithm [5]. The work in [3] applies integer wavelet theory to pre-process data with both loss and loss-less methods, followed by arithmetic coding, which results in 80% compression ratio, although with data loss.

In our work, we focus on the lossless compression of DEM data and follow the two-phase approach of previous methods, enhanced with heuristics in the data pre-processing phase for taking advantage of the knowledge about the data distribution; specifically, we exploit the terrain heterogeneity typical of DEM data in mountain ranges, to optimize both compression ratio and execution time in mountain regions, while preserving performance in other types of geographical areas.

3 Geodata Compression Algorithms

3.1 Problem Statement

The *SnowWatch* application addresses the problem of mobile AR for the enrichment of outdoor natural objects, specifically, mountain peaks. Restricting the focus to devices that support a bi-dimensional view, the mobile application receives as input a representation of the reality in which the user is embedded: a sequence of camera frames captured at a fixed rate, and the position and orientation of the device, acquired by the GPS and orientation sensors (e.g. magnetic sensor, gravity sensor, ...) ; the second input is the information about the possible objects present in a region of interest, as provided by the DEM of the geographical area surrounding the user; such information is used to render a 360° cylindrical virtual terrain view, which is aligned w.r.t. the current camera frame, to estimate the on-screen mountain peak positions. One of the biggest challenges is the requirement to operate in mountain regions, where internet connection may be unreliable. This calls for smart DEM pre-caching techniques, which can reduce consumption of both bandwidth and storage. The second challenge is the reduced computation power of mobile devices and the need to process DEM data in real time. The key to successfully address both problems is an efficient algorithm for DEM compression and decompression, guaranteeing a high level of compression and low decompression time, so that data can be transmitted faster, occupy less space in the device, and can be decoded quickly.

The DEM used in this work, provided by NASA, is the outcome of Shuttle Radar Topography Mission (SRTM) [7]; data can be seen as a regular grid overlaid on the Earth surface, where each grid point reports the height of the terrain in that position. We consider two different versions of SRTM DEM: SRTM1 at 1 arcsec resolution (which corresponds to $\approx 30m$ in areas relatively far from the poles), and SRTM3 at 3 arcsec resolution ($\approx 90m$). Both DEMs are provided as a series of tiles of $1'' \times 1''$, thus including $3601 \times 3601/1201 \times 1201$ points and having the size of $25,327KB$ and $2,818KB$, respectively in case of SRTM1 and SRTM3. Data are saved as 16 bits in Little Endian encoding. Another intuitive way to see a DEM is a gray scale satellite image, where the color of every pixel is associated to the average height of the terrain in that area. Although several efficient lossy compression techniques can be used [11], [17], DEM precision is vital for the correct outcome of peak identification (especially given other sources of error, such as device compass and GPS error). For this reason, we focus on lossless compression approaches, which allow us to compress a DEM and decompress it to its exact original state. Furthermore, to avoid selective downloading (e.g., ignoring data for areas of scarce mountaineering interest) and ensure full geographical coverage, the solution should guarantee high compression in areas with both high and low altitude variance.

3.2 Pre-Compression Data Encoding

The goal of our work is to design an encoding technique applicable before compression (and symmetrically a decoding technique after decompression) [13], which exploits the specificity of DEM data and the purpose of the peak recognition application to improve compression ratio with minimal runtime overhead. SRTM data can be regarded as a 3D surface with maxima and minima that correspond to the mountain peaks and depressions. Although in optimal conditions (e.g., flat land) each point can be interpolated from its neighbors, in a realistic and common scenario, simple interpolation does not work well. A better method is to apply a predictive rule that estimates the value of a point from its neighbors [13]. Once estimated, the computed point value can be expressed as the difference between the original and the predicted value. Since difference values are smaller than absolute ones, the amount of information to compress is reduced. Although the idea is simple, it involves two non-trivial decisions: 1) which point to pick at start ; 2) given a point, how many neighbors to use for predicting the value and how to select them. Differently from the described state-of-the-art methods, the proposed algorithm encodes DEM data recursively, as follows.

The procedure starts from the highest altitude point of the DEM tile to be encoded, and recursively calculates the value for the adjacent points in the possible eight directions. Each new value is calculated as the difference between the original point value and the average between the point treated in the preceding recursive step and another neighbor[2] (conventionally chosen as the next point in the current exploration direction). This transformation may produce both positive and negative values, which, due to two's complement encoding, are compressed inefficiency (e.g., the bit-to-bit difference between $+1$ and -1 is very high). Thus, a second step normalizes all values to positive, by adding to them the global minimum. This guarantees that elements close in terms of absolute value are also close in terms of bit encoding. Finally, the last step of the pre-processing exploits number encoding to create areas of the file with similar values. Each value in the file is a 16 bit (2 bytes) integer number. The high-byte of each cell is similar to its neighbors high-bytes, on the contrary the low-byte of each cell tend to be different from its neighbors low-bytes. Reorganizing the

[2] Trial-and-error experiments proved that increasing the number of neighbors for the average calculation decreases the total compression rate.

file putting all the first high-bytes first and all the low-bytes next we creates the above mentioned areas. Note that data decoding after decompression applies the same steps in reverse order (byte reorganization, de-normalization and recursive original value computation) and that the encoding is lossless.

This step allows to identify and remove resonance in the data, based on the terrain characteristic. The compression algorithm instead considers the data as a sequence of bytes and tries to exploit the remaining correlation of the data for a better compression. It is important to notice that the two techniques (proposed pre-compression step and the compression algorithm) are not interchangeable, because once the compression algorithm is applied, the data loose the characteristics needed to apply the pre-compression step.

3.3 Data Compression

The encoding described in the preceding section is generic and can be executed before any efficient compression algorithm; in this Section we describe the compression algorithms that produced the best results based on our experiments.

Currently, there are two classes of algorithms that perform optimally:

Lempel-Ziv compression methods (e.g., LZ77 [23]) work by substituting the occurrences of a specific string with a new value, composed by the distance from the first occurrence and the length of the string to copy. To further reduce the size of the new values, a dictionary table can be used, which associates values with shorter symbols. Such table can be stored as a header of the file and encoded. LZ77-based algorithms work well in presence of many repetitions, which are likely to occur after the described pre-processing phase. Furthermore, this family of algorithms provide a very fast decompression.

Prediction by Partial Matching (PPM) [4] algorithms consider the correlation between values (which, for example, could be linearly growing). They use N past values to predict the next one, trying to find the best relationship. During the prediction, an algorithm creates a ranking table in which more relevant repetitions are placed on top. Since our data values are already partially related (especially when the terrain is constant or linear) this method exploits its full potential. For several reasons PPM algorithms, however, tend to provide a worse performance, above all the decoding phase consuming a large amount of memory for past values and their frequency storage.

In our experiments, we compare two methods based on LZ77, namely, Deflate [5] and LZMA2 [15] and a PPM-based method called PPMd [21]. In addition, we also analyze Bzip2 [19], an algorithm based on Huffman coding [20], and the Burrows-Wheeler block sorting algorithm [1]. Bzip rearranges values inside a block to produce a more uniform sequence. Then the obtained sequences are substituted with symbols with Huffman coding.

4 Experimental Evaluation

4.1 Datasets

In compression problems, performance varies greatly with the nature of the data. For this reason, we experimented with different SRTM DEM examples, with variable characteristics:

- **ALPS1/ALPS3**: 34 square degree tiles (N43-N47, E5-E15) covering the whole Alpine region, extracted from SRTM1/SRTM3 DEM, for a total size of 98/881 MB respectively. Characterized by massive presence of mountains and high altitude variance.
- **North America (NA)**: 48 square degree tiles (N52-N59, W97-W102) covering a part of Canadian Manitoba province, extracted from SRTM3 DEM, for a total size of 138 MB. Characterized by relatively low altitude variance.
- **ANDES**: 44 square degree tiles (S13-S16, W67-W77) covering the whole Andes region, extracted from SRTM3 DEM, for a total size of 127 MB. Characterized by mixed low and high altitude variance regions.

For each dataset we computed the compression ratio in two sessions: 1) using the four compression algorithms alone (PPMD, Bzip2, LZMA2 and DEFLATE); 2) using them after our pre-compression encoding step. We also measured the compression and decompression times for each test session. For the fairness of comparison, when the pre-compression encoding is used, its contribution is included in the total execution time. All experiments are performed on a Windows 10 desktop workstation, i7, 8 GB RAM.

Let C and C^* denote, respectively, the compression ratio (i.e., original size of data / compressed size) obtained by the algorithm alone and with pre-compression encoding; $\Delta C = (C^* - C)/C$ represents the improvement obtained by using the pre-compression encoding. We also define T^c and T^d as the compression and decompression time required by the algorithms. The star symbol $(*)$ is applied also to execution times, to highlight the usage of pre-compression encoding; in this case, $\Delta T^c / \Delta T^d$ represent the improvement of using pre-compression encoding for the compression/decompression time, respectively.

Table 1 shows the obtained results on ALPS3 dataset, while Table 2, Table 3 and Table 4 show the results respectively on ALPS1, NA and ANDES datasets.

Table 1. Experimental results for the ALPS3 dataset.

Algorithm	C	C*	ΔC **(%)**	Tc (s)	Tc* (s)	ΔTc **(%)**	Td (s)	Td* (s)	ΔTd **(%)**
PPMD	2.7	3.66	+35.56	23.16	13.90	+39.98	22.6	15.00	+33.62
Bzip2	2.96	3.29	+11.30	10.60	8.26	+22.07	4.00	3.30	+17.50
LZMA2	2.48	3.22	+29.83	23.20	11.50	**+50.43**	24.60	13.00	**+47.15**
DEFLATE	1.70	2.79	**+64.20**	7.30	7.00	+4.11	0.80	1.10	-37.50

Table 2. Experimental results for the ALPS1 dataset.

Algorithm	C	C*	ΔC (%)	Tc (s)	Tc* (s)	ΔTc (%)	Td (s)	Td* (s)	ΔTd (%)
PPMD	5.44	6.83	+25.55	96.5	88.1	+8.81	95.2	80.5	+15.96
Bzip2	5.12	5.65	+10.35	117.9	56,2	**+52.32**	29.8	22.5	+24.49
LZMA2	4.19	5.44	+29.91	161.1	102,3	+36.50	17.02	13.1	+23.03
DEFLATE	2.29	4.72	**+106.06**	43.0	74,5	**-73.26**	6.02	4.1	**+31.89**

Table 3. Experimental results for the NA dataset.

Algorithm	C	C*	ΔC (%)	Tc (s)	Tc* (s)	ΔTc (%)	Td (s)	Td* (s)	ΔTd (%)
PPMD	10.39	11.26	+8.42	3.7	4.8	**-29.73**	3.6	3.7	**-2.78**
Bzip2	9.22	9.84	+6.74	25.2	9.5	**+62.30**	6.46	4.2	**+34.98**
LZMA2	7.74	9.37	+21.04	35	21.5	+38.57	2.3	1.9	17.39
DEFLATE	5.57	8.39	**+50.62**	15	22	**-46.67**	0.8	0.7	+12.5

The results show that the proposed pre-compression step provides a consistent improvement in compression efficiency in every scenario, varying from 6% to 106%, depending on the compression algorithm and the dataset. Although the best improvement occurs in regions with high mountain density and sensible altitude variance, the regions with low altitude variance still benefit from the pre-compression encoding. The execution time is generally improved too, with few exceptions (highlighted in red inside the result tables). However, it must be noted that the time increase occurs usually in cases where the pre-processing step achieves a high compression gain.

Since the goal of this work is to present a compression algorithm that could be efficiently executed also on mobile terminals, Table 5 and Table 6 show the time performance measured on a mobile device (namely, Galaxy S5 with Snapdragon 801 @2.5Ghz processor and 2GB of Ram) on ALPS3 dataset (clearly, the compression performance remains invariant). As one may see, the benefits of using the proposed approach are even higher than on the desktop configuration. Although, due to the space constraints, we do not report the analogous performance on other datasets - they are similar to the ones presented in Table 5 and Table 6.

Table 4. Experimental results for the ANDES dataset.

Algorithm	C	C*	ΔC (%)	Tc (s)	Tc* (s)	ΔTc (%)	Td (s)	Td* (s)	ΔTd (%)
PPMD	2.71	3.57	+31.71	30.7	19.1	+37.78	35.6	21.2	**+40.50**
Bzip2	2.81	3.22	+14.56	12.34	10.01	+18.96	6.1	5.05	+17.20
LZMA2	2.98	3.20	+7.46	25.1	15.04	**+40.07**	5.0	3.4	+32.00
DEFLATE	1.79	2.89	**+61.65**	13.1	20.0	**-53.84**	1.1	1.0	+9.09

Table 5. Experimental results for the ALPS3 dataset on a mobile device.

Algorithm	Tc (s)	Tc* (s)	ΔTc (%)	Td (s)	Td* (s)	ΔTd (%)
PPMD	6.5	5.1	+21.53	10.36	4,63	**+55.31**
Bzip2	1.2	0.7	**+41.67**	0.48	0.35	+27.08
LZMA2	1.4	1.03	+28.57	1.12	0.71	+36.61
DEFLATE	1.1	0.9	+18.18	0.18	0.2	**-11.11**

Table 6. Experimental results for the NA dataset on a mobile device.

Algorithm	Tc (s)	Tc* (s)	ΔTc (%)	Td (s)	Td* (s)	ΔTd (%)
PPMD	2.33	2.2	+5.58	1.91	1.90	+0.5
Bzip2	1.5	0.7	**+53.33**	0.30	0.19	**+36.67**
LZMA2	1.9	1.2	+36.84	0.49	0.46	+6.1
DEFLATE	1.6	1.1	+31.25	0.06	0.08	**-33.33**

5 Conclusions and Future Work

We presented an approach for the compression of DEM data and geographic entities for a mountain environment monitoring mobile AR application. We showed that a pre-compression encoding can be effectively applied, regardless of the underlying compression algorithm, and boosts the overall performance of the system both in terms of compression ratio and execution time. We tested the proposed approach with several state-of-art compression algorithms and DEM datasets with various characteristics. Although, as expected in compression problems, the final results vary sensibly w.r.t. the dataset and the compression algorithm, we obtained a constant improvement of the compression ratio, up to 50%, and an average improvement of both compression and decompression time.

The simplicity and generality of the proposed approach allows one:

– To apply the approach with potentially any compression algorithm.
– To easily integrate the approach in an already existing compression pipelines.
– To process any kind of geographical data (besides the DEM), as long as they assign values denoting the property of interest to terrain models sampled on a regular grid (e.g., snow level, terrain temperature, vegetation health, etc.).

Our future work will address the usage of adaptive predictors instead of static ones, i.e. predictors that adapt flexibly to the terrain type and morphology.

References

1. Burrows, M., Wheeler, D.J.: A block-sorting lossless data compression algorithm (1994)
2. Castelletti, A., Fedorov, R., Fraternali, P., Giuliani, M.: Multimedia on the mountaintop: Using public snow images to improve water systems operation. In: Proceedings of the 24rd ACM international conference on Multimedia. ACM (2016)
3. Chen, R., Li, X.: Dem compression based on integer wavelet transform. Geo-Spatial Information Science 10(2), 133–136 (2007)

4. Cleary, J.G., Witten, I.H.: Data compression using adaptive coding and partial string matching. Communications, IEEE Transactions on 32(4), 396–402 (1984)
5. Deutsch, L.P.: Deflate compressed data format specification version 1.3 (1996)
6. Deutsch, L.P.: Gzip file format specification version 4.3 (1996)
7. Farr, T.G., Kobrick, M.: Shuttle radar topography mission produces a wealth of data. Eos, Transactions American Geophysical Union 81(48), 583–585 (2000)
8. Fedorov, R., Camerada, A., Fraternali, P., Tagliasacchi, M.: Estimating snow cover from publicly available images. IEEE Transactions on Multimedia 18(6), 1187–1200 (2016)
9. Fedorov, R., Frajberg, D., Fraternali, P.: A framework for outdoor mobile augmented reality and its application to mountain peak detection. In: International Conference on Augmented Reality, Virtual Reality and Computer Graphics. pp. 281–301. Springer (2016)
10. Fedorov, R., Fraternali, P., Tagliasacchi, M.: Mountain peak identification in visual content based on coarse digital elevation models. In: Proceedings of the 3rd ACM International Workshop on Multimedia Analysis for Ecological Data. pp. 7–11. ACM (2014)
11. Franklin, W.R., Said, A.: Lossy compression of elevation data pp. 29–41 (1996)
12. Kidner, D.B., Smith, D.H.: Compression of digital elevation models by huffman coding. Computers & Geosciences 18(8), 1013–1034 (1992)
13. Kidner, D.B., Smith, D.H.: Advances in the data compression of digital elevation models. Computers & Geosciences 29(8), 985–1002 (2003)
14. Lewis, M., Smith, D.: Optimal predictors forthe data compression of digital elevation models using the method of lagrange multipliers.
15. Pavlov, I.: 7z format, `http://www.7-zip.org/7z.html`
16. Rissanen, J., Langdon, G.G.: Arithmetic coding. IBM Journal of research and development 23(2), 149–162 (1979)
17. Ruuvzika, J., Ruuvzika, K.: Impact of gdal jpeg 2000 lossy compression to a digital elevation model pp. 205–214 (2015)
18. Scarmana, G.: Lossless data compression of grid-based digital elevation models: A png image format evaluation. ISPRS Annals of the Photogrammetry, Remote Sensing and Spatial Information Sciences 2(5), 313 (2014)
19. Seward, J.: bzip2 and libbzip2, version 1.0.5 a program and library for data compression (1996), `http://www.bzip.org/1.0.5/bzip2-manual-1.0.5.html#intro`
20. Sharma, M.: Compression using huffman coding. IJCSNS International Journal of Computer Science and Network Security 10(5), 133–141 (2010)
21. Shkarin, D.: Ppmd-fast ppm compressor for textual data (2001)
22. Smith, D.H., Lewis, M.: Optimal predictors for compression of digital elevation models. Computers & Geosciences 20(7), 1137–1141 (1994)
23. Ziv, J., Lempel, A.: A universal algorithm for sequential data compression. IEEE Transactions on information theory 23(3), 337–343 (1977)