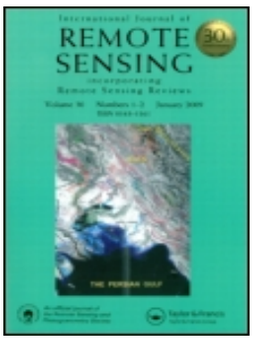

International Journal of Remote Sensing

ISSN: 0143-1161 (Print) 1366-5901 (Online) Journal homepage: https://www.tandfonline.com/loi/tres20

Efficient method for lossless LIDAR data compression

Domen Mongus & Borut Žalik

To cite this article: Domen Mongus & Borut Žalik (2011) Efficient method for lossless LIDAR data compression, International Journal of Remote Sensing, 32:9, 2507-2518, DOI: 10.1080/01431161003698385

To link to this article: https://doi.org/10.1080/01431161003698385

International Journal of Remote Sensing
Vol. 32, No. 9, 10 May 2011, 2507–2518

Taylor & Francis
Taylor & Francis Group

Efficient method for lossless LIDAR data compression

DOMEN MONGUS* and BORUT ŽALIK
University of Maribor, Faculty of Electrical Engineering and Computer Science, Smetanova 17, SI-2000 Maribor, Slovenia

(*Received 10 August 2009; in final form 21 December 2009*)

Light Detection and Ranging (LIDAR) has become one of the prime technologies for rapid collection of vast spatial data, usually stored in a LAS file format (LIDAR data exchange format standard). In this article, a new method for lossless LIDAR LAS file compression is presented. The method applies three consequent steps: a predictive coding, a variable-length coding and an arithmetic coding. The key to the method is the prediction schema, where four different predictors are used: three predictors for x, y and z coordinates and a predictor for scalar values, associated with each LIDAR point. The method has been compared with the popular general-purpose methods and with a method developed specially for compressing LAS files. The proposed method turns out to be the most efficient in all test cases. On average, the LAS file is losslessly compressed to 12% of its original size.

1. Introduction

Light Detection and Ranging (LIDAR) is an attractive remote-sensing technique that uses scattered light (a laser beam) to obtain information about distant objects. Similar to radar technology, a time difference between transmission of a laser pulse and detection of its reflection is used to calculate the distance to the observed object. As LIDAR uses electromagnetic signals with much shorter wavelengths (near-infrared) than Radio Detection and Ranging (RADAR), much higher precision is achieved (Wehr and Lohr 1999).

LIDAR data scanners can execute over 100 000 measurements per second (Maune 2008). Usually, they are mounted on aircrafts, and a vast amount of spatial data with high density can be captured (Lienert *et al.* 1999, Briese *et al.* 2002, Kerry and Amnon 2002, Gamba *et al.* 2003, Nayegandhi *et al.* 2009). The determination of the position and the altitude of the LIDAR scanner are necessary to determine coordinates of sampled points. The position of the LIDAR scanner is defined by a global positioning system (GPS), while the inertial measurement unit (IMU) measures the roll, pitch and heading of the aircraft (Maune 2008). In this way, an angular orientation of the LIDAR sensor is established. Furthermore, the LIDAR sensor measures the scan angle of the laser pulses and, combined with the IMU, the angular orientation of each emitted laser pulse is defined. Using angular orientation and the range measurement between the LIDAR scanner and surveyed points, the positions of sampled points are obtained, as shown in figure 1. Recorded points are projected onto one of the local geographic coordinate systems (e.g. Gauss–Krüger coordinate system) or a global

*Corresponding author. Email: domen.mongus@uni-mb.si

International Journal of Remote Sensing
ISSN 0143-1161 print/ISSN 1366-5901 online
http://www.tandf.co.uk/journals
DOI: 10.1080/01431161003698385

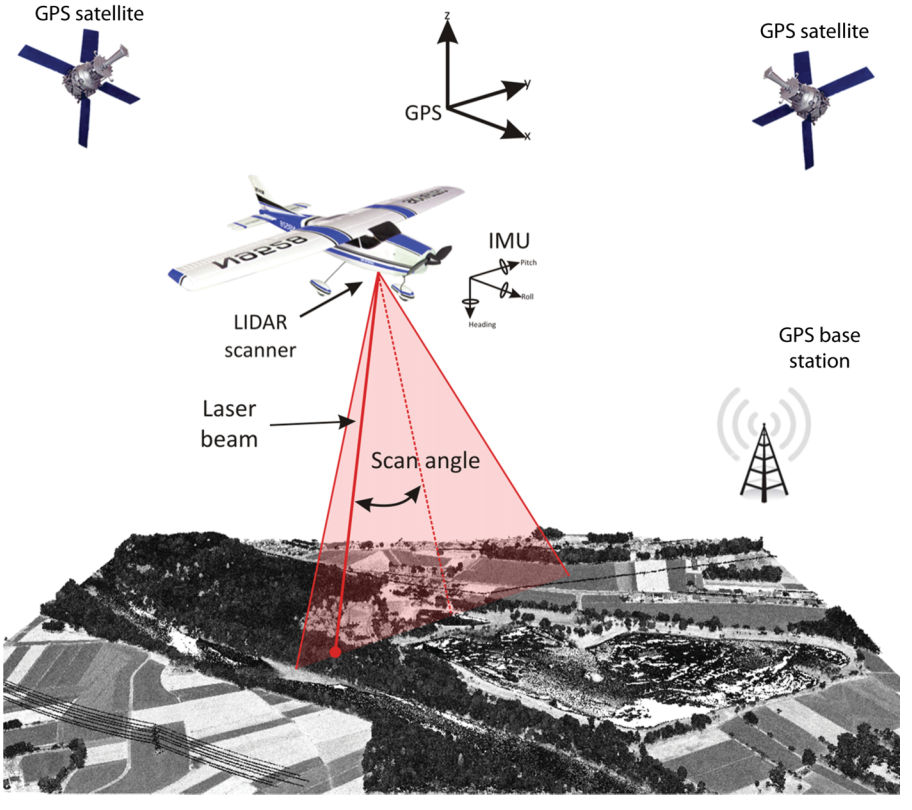

Figure 1. Collecting LIDAR data.

geographic coordinate system is used (e.g. Universal Transverse Mercator coordinate system).

The industrial standard for storing spatial data collected by LIDAR is a LAS file – an open binary LIDAR data exchange format standard (ASPRS 2008). Although there are slight differences between different versions of LAS formats, they all consist of:

- a header, where general information about the LIDAR dataset is stored, such as the number of points, version of LAS file format, or coordinate bounds;
- variable-length records with additional information such as projection, meta-data and user data; and
- point-data records, which store points with associated scalar values.

LAS files are huge as they may contain several tens of millions of points. In practice, this represents considerable problems. Their storage is expensive, it is difficult to deliver them to the users, sending them through the internet takes a long time or it is even impossible. Because of this, compression of LIDAR data has recently become an important issue (Pradhan *et al.* 2005). In this article, we present a new method for lossless compression of LIDAR data stored in LAS format. The article is organized as follows: in section 2, a brief overview of basic concepts for point-geometry

compression is presented; section 3 explains the proposed method; the results are presented in section 4, and section 5 concludes the article.

2. Related work

The pioneering work of compressing geometrical data was carried out by Taubin and Rossignac (1998), who published a method for compressing triangular meshes. Their algorithm divided the triangle mesh into triangular strips. The vertices were then arranged according to their appearances in the triangular strips and coded with a linear prediction schema. In this way, instead of storing the absolute coordinates, only differences between the predicted and the actual positions of vertices were stored. Due to a need for the topology, this algorithm cannot be directly applied to LIDAR datasets.

A lossless compression method for densely sampled surfaces (point clouds) was presented by Schnabel and Klein (2006). Their method was based on the octree space subdivision, and the point cloud was encoded according to the octree cells. To compress the points, a predictive schema was applied based on a local surface approximation. A similar method, also based on the octree space subdivision, was proposed by Huang *et al.* (2008). They proposed a generic point-cloud encoder that provided a framework for compression of three-dimensional (3D) points. The coding process was controlled by the octree space subdivision, where at each level of subdivision the positions of points were approximated by the geometrical centres of the octree cells. Normals, colours and other attributes were also approximated by the statistical average within the octree cell.

Another predictive method based on a spanning tree was presented by Gumhold *et al.* (2005). The method constructed a prediction tree that specified which previously encoded points should be used for predicting the position of the next point in the sequence. The spanning tree was constructed using a greedy point-by-point method that tried to minimize the prediction error. The prediction was based on the location of points in the spanning tree, using two predictors. The new point could either be placed at the same position as the parent or its position was predicted from the parent and grandparent (its distance from the parent was the same as the distance between the parent and the grandparent). Because this method can be adapted for streaming, large datasets can be compressed. A very similar technique for compression of point clouds was presented by Merry *et al.* (2006). This method used linear and lateral predictors. In the case of the linear prediction, the previous edge was used to predict the position of a new point. If the lateral prediction was applied, the previous edge was rotated by 90° with regards to the estimated normal. According to the authors, this technique was particularly efficient when applied to the regularly sampled points.

The lossy compression schema for LIDAR data was presented by Pradhan *et al.* (2005). The method was based on the second generation wavelet where the interpolation wavelet filter was applied. A random set of points was firstly used for the surface approximation by applying Delaunay triangulation. In the splitting step, a triangle was divided into several sub-triangles, and the elevation step was used to modify the point values (point geometry) after the splitting.

Isenburg developed a lossless algorithm for LIDAR data compression. At the moment of writing, the details of the algorithm were not yet known. However, Isenburg (2009) provided a demonstration program, which we have used for comparison (see section 6).

3. Method overview

The majority of presented methods use techniques for space partitioning to establish a correlation between points in order to obtain better predictions. In our case, this is not needed because we exploit the information about how the LIDAR data points are collected. LIDAR points are successively recorded and stored in a series of LAS point records. Each point is represented by a triple (x, y, z), with associated scalar values such as intensity, classification index, scan-angle rank, edge of flight line, point-source identification (ID) and user-specific data. The algorithm arranges attributes of LAS point records into streams, where the same attribute records are bound together and compressed independently. The entire file is compressed in one pass, which is the characteristic of so-called single-rate methods (Taubin and Rossignac 1998, Lee *et al.* 2002, Merry *et al.* 2006). Each stream is compressed into three steps (figure 2):

- predictive coding with determination of an error between the predicted and the actual value;
- coding of obtained error values by variable-length codes (VLCs); and
- compression of VLCs by arithmetic coding.

As arithmetic coding is well-known (Langdon 1984, Salomon 2006), only the first two steps are explained in the following.

3.1 *Predictive coding*

As mentioned in section 2, predictive coding has proved to be an efficient approach for compression of geometric data. Let s_i be the actual value being coded and suppose the prediction s_i' for this value is calculated by a prediction function $f(s)$, where s contains already seen values. Theoretically, all seen values can be used by f, but usually only the last few (possible only the last) of them are applied. As a result of prediction, the error e_i between the predicted and the actual value is calculated according to

$$e_i = s_i - s_i', \tag{1}$$

$$s_i' = f(s_{i-1}, s_{i-2}, ..., s_0) = f(s).$$

In our method, four different predictors are used.

Predictor for scalar attributes: the majority of scalar attributes (i.e. source ID, scanned angle, classification) in LAS files rarely change. Because of this, they can be predicted by a constant predictor (Sayood 2005, Salomon 2006) defined by

$$s_i' = s_{i-1}, \tag{2}$$

where s_{i-1} is the previous value in the stream of the particular scalar attribute and s_i' is the predicted value. On the other hand, points are rarely at the same position and

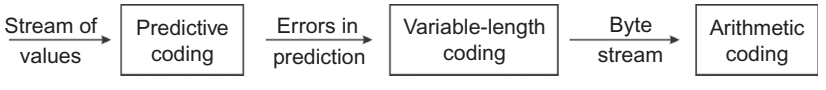

Figure 2. The single-rate compression scheme of LIDAR data.

therefore, the constant predictor would be highly inefficient. Even more, it has proven that it is sensible to develop different predictors for each point coordinate.

Predictor for *x* coordinates: because LIDAR points are collected almost uniformly, the distance between two successive points $\boldsymbol{p}_{i-1}$ and $\boldsymbol{p}_i$ is usually invariant with respect to the coordinate x. However, large deviations in the distances can appear when more reflections of a single laser pulse occur (e.g. when the laser beam penetrates the vegetation cover). Therefore, two predictors are used to predict the x coordinates of a given point $\boldsymbol{p}_i$:

- if the deviation in the distances between consecutive points is smaller than a predefined value δ, prediction s_i' is estimated as $\overline{\Delta x_{i-1}}$, representing an average distance of k_x previous x values; and
- when a large deviation in the x distances is noticed, the prediction of the deviation is included by subtracting the residual of a sample $\widehat{\varepsilon}_{i-1}$.

The prediction model for the x coordinates is summarized by:

$$x_i' = \begin{pmatrix} x_{i-1} + \overline{\Delta x_{i-1}} & ; & d \leq \delta \\ x_{i-1} + \overline{\Delta x_{i-1}} - \widehat{\varepsilon}_{i-1} & ; & d > \delta \end{pmatrix}, \tag{3}$$

where

$$\Delta x_i = x_i - x_{i-1},\ \overline{\Delta x_i} = \frac{\sum_{n=i}^{n=i-k_x} \Delta x_n}{k_x},\ \widehat{\varepsilon}_i = x_i - \overline{\Delta x_i},$$

$$d = \sqrt{(\Delta x_i - x_{i-1})^2},\ \delta = 0.05 \text{ and } k_x = 10.$$

The value of δ has been determined through experiments and set to 5 cm. Similarly, through experiments, the constant k_x was determined as a trade-off between compression and time efficiency.

Predictor for *y* coordinates: because LIDAR data are collected at an angle, a geometric correlation in point positions exists (see figure 3). Therefore, the difference Δx_i between x coordinates of successive points $\boldsymbol{p}_{i-1}$ and $\boldsymbol{p}_i$ can be exploited to predict the y coordinates y_i' of $\boldsymbol{p}_i$. Unfortunately, the positions of successive points are dependent on the flight direction and the scan angle, which constantly change over time. The points are represented in one of the geographic coordinate systems which is, of course, not dependent on the flight direction. The flight direction is also not predictable and slightly varies. Thus, it is impossible to estimate an exact correlation between point positions. Therefore, two prediction rules are used for the approximation:

- among the last k_y samples, two successive points $\boldsymbol{p}_n$ and $\boldsymbol{p}_{n-1}$ are tried to be found, where Δx_n is similar to Δx_i; if such points are found, then y_i' is estimated using the difference Δy_n between y coordinates of $\boldsymbol{p}_n$ and $\boldsymbol{p}_{n-1}$ and
- if such points do not exist, a linear prediction based on Δx is applied.

The predictor for the y coordinates is formally defined by:

$$y_i' = \begin{pmatrix} y_{i-1} + \Delta y_n & ; & \Delta x_n \doteq \Delta x_i \text{ and } i - k_y \leq n < i \\ y_{i-1} + \Delta_{x_i} * k_{\Delta x} & ; & \text{else} \end{pmatrix}, \tag{4}$$

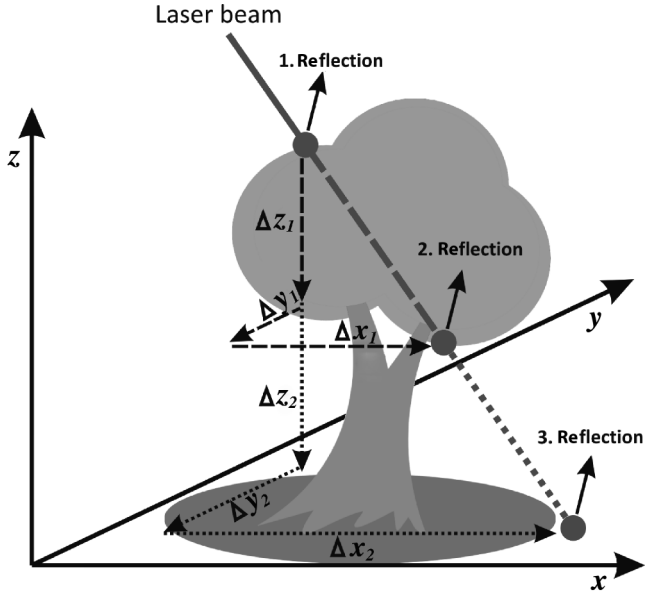

Figure 3. An example of geometric correlations between positions of successive points. When a laser beam penetrates the vegetation cover, three reflections of a laser pulse occur, and three linear dependant points are recorded. This shows that Δx, Δy and Δz are correlated values.

where

$$k_{\Delta x} = \frac{\Delta y_{i-1} - \Delta y_{i-2}}{\Delta x_{i-1} - \Delta x_{i-2}},\ \Delta_{x_i} = \Delta x_i - \Delta x_{i-1},$$

$$\Delta x_i = x_i - x_{i-1},\ \Delta y_i = y_i - y_{i-1} \text{ and } k_y = 100.$$

The value of k_y is a constant that is set through experiments, similar to before for k_x.

Predictor for z coordinates: the concept presented for the prediction of y coordinates is also applied to z coordinates. Three predictors are used in this case:

- the last k_x samples are searched in order to find two of successive points $\boldsymbol{p}_n$ and $\boldsymbol{p}_{n-1}$, where Δx_n is similar to Δx_i and Δy_n is similar to Δy_i; if such points are found, then z_i' is estimated, based on Δz_n calculated from $\boldsymbol{p}_n$ and $\boldsymbol{p}_{n-1}$;
- in the case that such points do not exist, we try to estimate z_i' with linear prediction based on Δx; or
- with linear prediction based on Δy.

To determine, which linear prediction rule will be applied (linear prediction based on Δx or those based on Δy), two statistical variables to monitor the accuracy of both prediction rules are used. The predictor for the z coordinates is therefore formally defined by:

$$z_i' = \begin{pmatrix} z_{i-1} + \Delta z_n & ; & (\Delta x_n, \Delta y_n) \doteq (\Delta x_i, \Delta y_i) \text{ and } i - k_z \leq n < i \\ z_{i-1} + \Delta_{x_i} * k_{\Delta x} & ; & error X \leq error Y, \\ z_{i-1} + \Delta_{y_i} * k_{\Delta y} & ; & error X > error Y, \end{pmatrix}, \quad (5)$$

where

$$\Delta_{x_i} = \Delta x_i - \Delta x_{i-1},\ \Delta_{y_i} = \Delta y_i - \Delta y_{i-1},$$

$$k_{\Delta x} = \frac{\Delta z_{i-1} - \Delta z_{i-2}}{\Delta x_{i-1} - \Delta x_{i-2}},\ k_{\Delta y} = \frac{\Delta z_{i-1} - \Delta z_{i-2}}{\Delta y_{i-1} - \Delta y_{i-2}},$$

$$\Delta x_i = x_i - x_{i-1},\ \Delta y_i = y_i - y_{i-1},\ \Delta z_i = z_i - z_{i-1} \text{ and } k_z = 100.$$

The value *error X* is an average error made by linear prediction based on the x coordinate and *error Y* is an average error made by linear prediction based on the y coordinate. Similar to before, k_x was determined through experiments. It is clear that the x coordinates should be decoded before the y coordinates. To decode the z coordinates, both x and y coordinates have to be known already.

3.2 *Variable-length coding*

With the predictive coding scheme presented, the radical reduction of the absolute values of point coordinates and associated attributes is achieved. In order to reduce the number of bytes required for their storage, a VLC is applied in three steps:

(i) Each value s_i from the input stream $s = (s_0, s_1, \ldots, s_n)$ is associated with a description byte d_i. It contains the information about the sign and the actual size in bytes about s_i (see figure 4). In this way, the input stream contains only the absolute values $|s| = (|s_0|, |s_1|, \ldots, |s_n|)$. As seen, the first 4 bits in each description byte are not used and are set to 0. In the last step of our algorithm, these leading zeros are easily eliminated by the arithmetic coding. Therefore, a stream of description bytes $d = (d_0, d_1, \ldots, d_n)$ and $|s|$ are coded independently.

(ii) Originally, each $|s_i|$ is stored in 4 bytes. In this step, the size of each $|s_i|$ is reduced according to the information stored in d_i. Each $|s_i|$ is considered as a string of 4 bytes $|s_i|^3, |s_i|^2, |s_i|^1, |s_i|^0$, where zero bytes are removed (see figure 5).

(iii) The non-zero bytes form byte streams based on their significance (see figure 5), where the left-most byte $|s_i|_n$ is the most significant byte (MSB) and the right-most byte $|s_i|_0$ is the least significant byte (LSB). Each stream is compressed individually by arithmetic coder.

Decoding of byte streams is the inverse procedure, where bytes are obtained from the byte streams. Based on the description byte d_i, zero bytes are added to get the numeric values, and finally the correct sign is added to the numerical values to get the original values.

4. Results

The presented algorithm has been implemented in C++ under the Microsoft Windows operating system. The efficiency of the algorithm is demonstrated on the

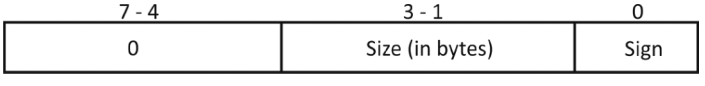

Figure 4. Description byte d_i related to the encoding value s_i.

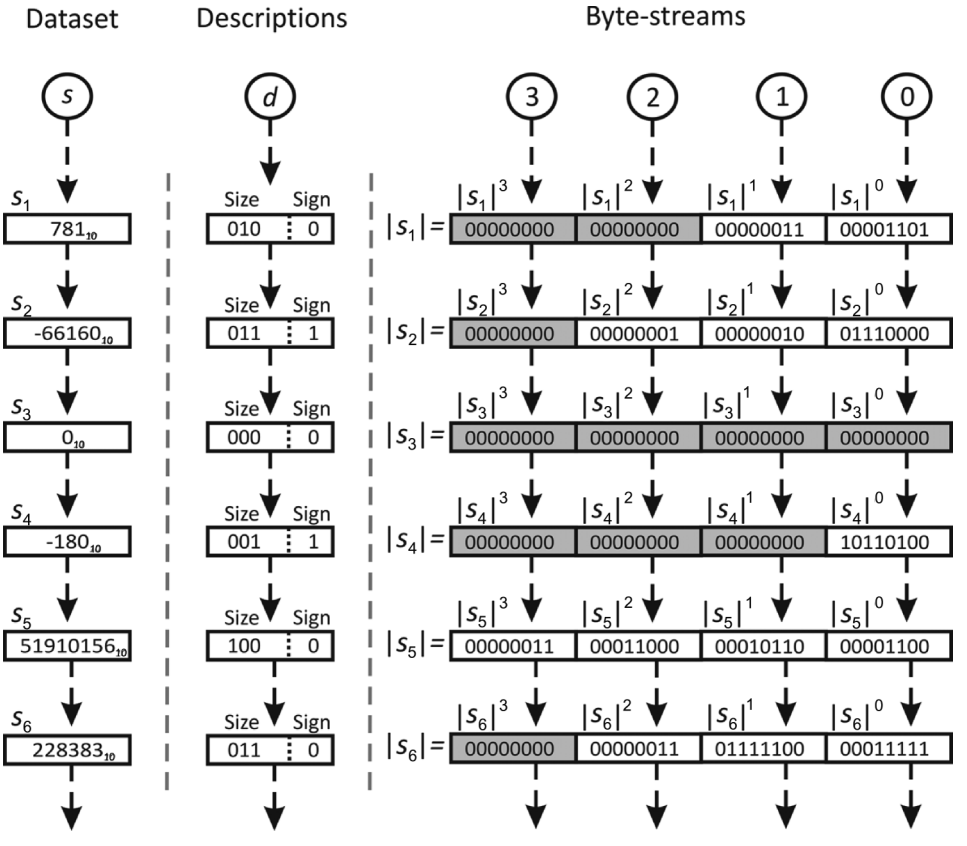

Figure 5. An example of adjusting the size of values $s\{781, -6616, 0, -180, 51910156, 228383\}$, where all the remaining values are represented using binary notation. First, d is constructed containing the sign and the size of s_i. According to d_i, s_i is considered as a string of bytes $|s_i|^3$, $|s_i|^2$, $|s_i|^1$, and $|s_i|^0$. Zero bytes, marked darker, are removed.

Table 1. Description of test LIDAR LAS files.

File	No. points	Size (m × m)	Density	Description	Size (kB)
1	3 581 247	1 000 × 1 000	3.58	Hilly terrain	97 925
2	3 582 203	1 000 × 1 000	3.58	Hilly terrain	97 947
3	3 652 365	2 104 × 1 975	0.88	Flat terrain	99 870
4	3 717 437	2 104 × 1 975	0.89	Flat terrain	101 649
5	6 411 098	3 288 × 2 163	0.90	City terrain	175 304
6	12 967 073	1 872 × 1 779	3.89	Hilly terrain	354 569
7	13 806 773	1 872 × 1 779	4.15	Hilly terrain	377 530
8	15 716 290	4 000 × 4 454	0.88	Flat terrain	429 743
9	15 902 042	4 000 × 4 454	0.89	Flat terrain	434 822
10	27 027 335	1 964 × 2 965	4.64	Hilly terrain	739 029
11	29 032 708	1 964 × 2 965	4.99	Hilly terrain	793 864
12	33 326 065	5 000 × 7 454	0.89	Flat terrain	911 260
13	34 000 000	5 000 × 7 454	0.91	Flat terrain	929 688

test dataset. For this purpose, 13 different LIDAR LAS files containing different terrains, number of points, sizes and densities have been tested (see table 1).

This dataset has been compressed by four different algorithms for lossless data compression:

- **7-Zip**, which is believed to be one of the most efficient open-source algorithms for lossless data compression (Pavlov 2009);
- **RAR**, which is another widely used algorithm for lossless data compression (RAR Lab 2009);
- **LASZip**, which is a highly efficient algorithm, implemented by Isenburg (2009), especially adopted for LIDAR data compression; and
- **LASComp**, as we named our algorithm, presented in this article.

Although one can obey the usage of general-purpose 7-ZIP and RAR compression algorithms for comparison against domain-specific algorithms for LAS files as LASZip and LASComp, 7-ZIP and RAR are widely used in practice. We believe that the reader can get a better impression about the properties of the proposed approach when referenced to 7-ZIP and RAR. While testing, we measured the size of compressed files and calculated the compression ratio defined by (Salomon 2006):

$$\text{Compression ratio} = \frac{\text{Size of compressed file}}{\text{Size of uncompressed file}} \times 100. \tag{6}$$

In our case, the LAS point-record format version 1.1 is used, where 224 bits-per-point (bpp) is needed (ASPRS 2008). Table 2 summarizes the compression performances of the tested algorithms. It is observed that the proposed algorithm achieves the best compression ratio in all cases and compresses LAS files generally to less than 12% of their original size. Figure 6 shows the results of compression algorithms in bpp.

The efficiency of all lossless compression algorithms is data dependent (Sayood 2005, Salomon 2006). From table 1, it can be noticed that a better compression ratio is achieved in the case of flat terrains. In datasets that present hilly regions (files 1, 2 and

Table 2. Comparison of efficiency between algorithms on the test dataset.

	Compressed size (kB)				Compression ratio (%)			
File	7-Zip	RAR	LASZip	LASComp	7-Zip	RAR	LASZip	LASComp
1	22 157	17 033	13 145	11 550	22.63	17.39	13.42	11.79
2	24 511	19 749	15 397	13 596	25.02	20.16	15.72	13.88
3	20 093	13 981	10 659	9 302	20.12	14.00	10.67	9.31
4	20 310	13 658	10 490	9 289	19.98	13.44	10.32	9.14
5	38 475	30 636	22 774	20 250	21.95	17.48	12.99	11.55
6	81 866	65 505	53 069	47 420	23.09	18.47	14.97	13.37
7	94 301	76 733	62 362	53 447	24.98	20.33	16.52	14.16
8	87 132	61 443	46 849	40 445	20.28	14.30	10.90	9.41
9	88 016	61 516	46 994	40 926	20.24	14.15	10.81	9.41
10	187 457	154 528	129 806	111 359	25.37	20.91	17.56	15.07
11	207 793	172 461	142 789	129 459	26.17	21.72	17.99	16.31
12	182 997	127 359	97 131	84 916	20.08	13.98	10.66	9.32
13	195 400	145 004	111 515	99 644	21.02	15.60	11.99	10.72
Average					22.38	17.07	13.42	11.80

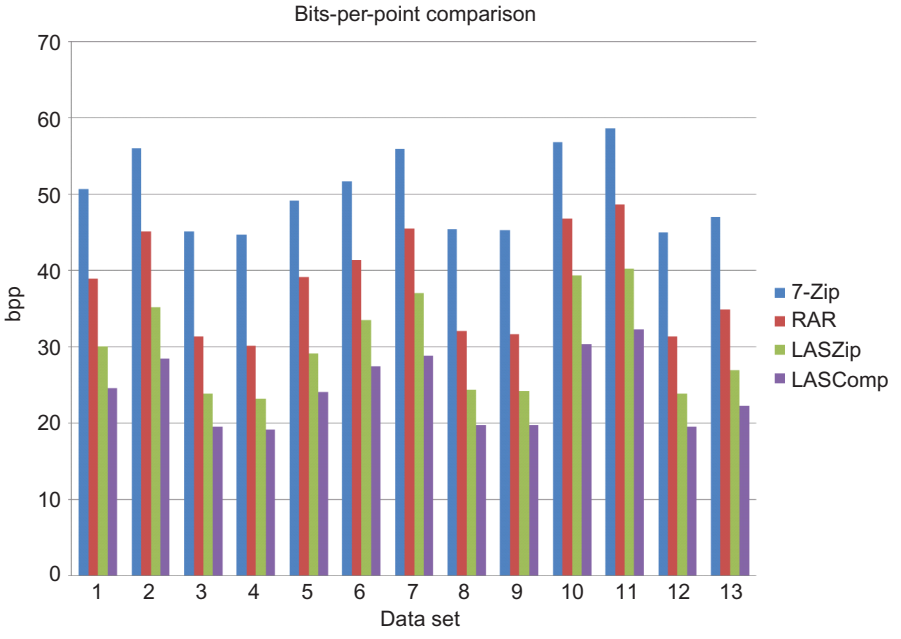

Figure 6. Bits-per-point (bpp) comparison between tested algorithms.

6), geometry prediction is slightly less efficient. Furthermore, the compression ratio also depends on the data density: the higher the density the worse the compression ratio (files 1, 2, 6, 7, 10 and 11). The reason for this is that dense datasets usually contain a lower percentage of the redundancy. This is particularly noticeable in the case of forested areas, where a higher penetration ratio occurs. However, compared to other tested algorithms for lossless data compression, the data dependency is smaller.

Using various segmentation methods, the amount of the LIDAR dataset can be reduced by eliminating needless points. Such segmented datasets can be further compressed by the proposed method. However, to preserve the efficiency of the algorithm, the data-abstraction techniques should not significantly change the order of the points.

5. Conclusion

A new method for single-rate lossless LIDAR data compression has been presented in this article. The method is based on the prediction paradigm. We have proposed four different predictors; three for coding the geometry of points and one for other parameters associated with the points. In the case of geometry coding, the change in the position between two successive points in the x direction is used to predict the changes in the y direction. Similarly, for the z coordinates, the information about the x and y coordinates are used. The errors in the predictions are coded with a VLC, minimizing the number of required bytes. Finally, the redundancy in the obtained stream of bytes is removed using arithmetic coding. With the presented algorithm, we are able to reduce the size of datasets generally to less than 12% of their original size. As confirmed through experiments, we achieve the best compression ratio among the

tested algorithms. Because the output data are represented by different streams, the presented algorithm can also be parallelized.

The executable version of the test application, for use on a Microsoft Windows operation system, and some examples of datasets tested are available in Mongus and Žalik (2009).

Acknowledgements

This work was supported by the Slovenian Research Agency under grants 1000-08-310105, P2-0041 and L2-3650.

References

American Society for Photogrammetry and Remote Sensing (ASPRS), 2008, LAS 1.2 format standard. Available online at: www.asprs.org (accessed 07 August 2009) and www. liblas.org (accessed 07 August 2009).

Briese, C., Pfeifer, N. and Dorninger, P., 2002, Applications of the robust interpolation for DTM determination. *International Archives of the Photogrammetry, Remote Sensing and Spatial Information Sciences*, **34**, pp. 55–61.

Gamba, P., Dell'Acqua, F. and Houshmand, B., 2003, Comparison and fusion of LIDAR and InSAR digital elevation models over urban areas. *International Journal of Remote Sensing*, **24**, pp. 4289–4300.

Gumhold, S., Karni, Z., Isenburg, M. and Seidel, H.P., 2005, Predictive point-cloud compression. In *International Conference on Computer Graphics and Interactive Techniques*, 31 July–4 August 2005, LA, USA, article no. 137 (New York, NY: ACM).

Huang, Y., Peng, J., Kuo, C.C.J. and Gopi, M., 2008, A generic scheme for progressive point cloud coding. *IEEE Transactions on Visualization and Computer Graphics*, **14**, pp. 440–453.

Isenburg, M., 2009, LASZip algorithm homepage. Available online at: www.cs.unc.edu/isenburg (accessed 07 August 2009).

Kerry, M. and Amnon, K., 2002, Integration of laser-derived DSMs and matched image edges from generating an accurate surface model. *ISPRS Journal of Photogrammetry and Remote Sensing*, **56**, pp. 167–176.

Langdon, G.G., 1984, An introduction to arithmetic coding. *IBM Journal of Research and Development*, **28**, pp. 135–149.

Lee, H., Alliez, P. and Desbrun, M., 2002, Angle-analyzer: a triangle-quad mesh codec. *Computer Graphics Forum*, **21**, pp. 383–392.

Lienert, B.R., Porter, J.N. and Sharma, S.K., 1999, Real time analysis and display of scanning Lidar scattering data. *International Journal of Remote Sensing*, **22**, pp. 259–265.

Maune, D. F., 2008, Aerial mapping and surveying. In *Land Development Handbook*, 3rd edn, S.O. Dewberry and L.N. Rauenzahn (Eds), pp. 877–910 (New York, NY: McGraw-Hill Professional).

Merry, B., Marais, P. and Gain, J., 2006, Compression of dense and regular point clouds. In *Proceedings of the Afrigraph 2006: the 4th International Conference on Computer Graphics, Virtual Reality, Visualisation and Interaction in Africa*, 29–30 July 2006, Cape Town, South Africa, pp. 15–20 (New York, NY: ACM).

Mongus, D. and Žalik, B., 2009, LAS compression algorithm homepage. Available online at: gemma.uni-mb.si/algorithms.phtml (accessed 7 August 2009).

Nayegandhi A., Brock J.C. and Wright C.W., 2009, Small-footprint, waveform-resolving lidar estimation of submerged and sub-canopy topography in coastal environments. *International Journal of Remote Sensing*, **30**, pp. 861–878.

Pavlov, I., 2009, 7-Zip algorithm homepage. Available online at: www.7-zip.org (accessed 07 August 2009).

Pradhan, B., Kumar, S., Mansor, S., Ramli, A.R. and Sharif, A.R.B.M., 2005, Light Detection and Ranging (LIDAR) data compression. *KMITL Science and Technology Journal*, **5**, pp. 515–526.

RAR Lab, 2009, Win RAR homepage. Available online at: www.rarlab.com (accessed 07 August 2009).

Salomon, D., 2006, *Data Compression: the Complete Reference*, 4th edn (New York, NY: Springer).

Sayood, K., 2005, *Introduction to Data Compression*, 2nd edn (San Francisco, CA: Morgan Kaufmann Publishers Inc.).

Schnabel, R. and Klein, R., 2006, Octree-based point-cloud compression. In *Proceedings of the Symposium on Point-Based Graphics 2006*, 29–30 July 2006, Boston, MA, pp. 147–156.

Taubin, G. and Rossignac, J., 1998, Geometric compression through topological surgery. *ACM Transactions on Graphics*, **17**, pp. 84–115.

Wehr, A. and Lohr, U., 1999, Airborne laser scanning – an introduction and overview. *ISPRS Journal of Photogrammetry and Remote Sensing*, **54**, pp. 68–82.