Multithreaded approach for lossless LiDAR data compression

Domen Mongus, Denis Špelič, Borut Žalik
Faculty of Electrical Engineering and Computer Science, University of Maribor,
Smetanova ul. 17, SI 2000 Maribor, Slovenia
{domen.mongus, denis.spelic, zalik}@uni-mb.si

Abstract

Light detection and ranging (LiDAR) has the capability of capturing a huge amount of highly accurate spatial data. However, the size of the data causes a lot of problems associated with its exchange and storage. In this paper, a method for lossless LiDAR data compression is presented. Although, efficient methods in terms of compression ratio have already been proposed, their time efficiency can be improved. For this purpose, a multithreading schema was developed to increase the use of the computer resources (i.e. multi-core central processor unit and direct memory access). In this way, the overall compression time has been reduced over 70%.

Keywords: *LiDAR, multithreading, lossless compression, predictive coding.*

1. INTRODUCTION

In the recent years, light detection and ranging (LiDAR) has become one of the prime remote sensing technologies [1, 2, 3]. Most of its capabilities arise from the use of the laser light to measure the range from the distant objects. The range is calculated based on the time delay between the transmission of the laser pulse and detection of its reflection [4]. Since a short wavelength signal is used, LiDAR achieves high accuracy and performs the measurements extremely fast [5]. As such, it is able to capture large amount of highly accurate and dense data in a short time. Because of this, it becomes one of the most widely used techniques within a wide range applications [6, 7, 8].

LiDAR is especially important in geosciences, where relatively large Earth' s surface can be gathered by airborne LiDAR systems[1, 2, 3, 6, 7, 8]. The airborne LiDAR systems are mounted on aircrafts and obtain geographical position of measured points by the use of the global positioning system (GPS) and inertial measurement unit (IMU) [5]. GPS is used to define the position of the aircraft, while IMU measures the roll, the pitch, and the heading of the aircraft. By that, and by measuring the scan angle, the angular orientation of each point is established and thus, the position of the point can be defined (see Fig. 1). Furthermore, such systems are capable to distinguish between different reflections of the emitted laser pulse. In this way, they can retrieve some points from the Earth's surface even below the vegetation [4].

The gathered points are usually saved in a LAS file, which represents the industrial standard for storing and exchanging LiDAR data [9]. In a LAS format, points are represented by *xyz*-coordinates. Scalar values are associated with each point thet represents, for example, an intensity, a reflection number or user specified data. Exact specification depends on the version of the LAS format. Because LiDAR systems can perform over 200.000 measurements per second (which allows retrieving over 35 points per square meter), such files become extremely large. They often contain several tens of millions of points and their size can easily reach more than a few gigabytes. Therefore, to store such files is difficult, while their exchange over the local networks and internet is practically impossible. Because of this, the compression of LiDAR data is of great relevance to the remote sensing community [5]. However, as huge files have to be compressed (and decompressed), the time for compression represents another important factor for efficient maintaining of the data.

In this paper, we present a multithreaded approach for LiDAR data compression, which exploits multi-core central processor units to speed up the compression process. In section 2, related work on LiDAR data compression is briefly presented. Brief description of the used compression method is given in section 3. In section 4, the multithreaded approach is explained in details. The results are presented in section 5. Section 6 concludes the paper.

2. RELATED WORK

Early works on point compression were closely related to compressing the topology of the triangular meshes. Thus, one of the earliest methods presented in [10] uses triangular mesh, beside which the geometry of points is compressed. The method uses a prediction of the next points position. For this purpose, the triangular mesh is divided into stripes. Points are then compressed based on the order of their appearances in such stripes, where the position of the next point is encoded using linear prediction schema [11]. In this way, only the differences between predicted and actual positions of points are stored. Since this approach compresses the topology as well as the geometry of triangular mesh, it is not suitable for LiDAR data, where neighbouring relations of points are not known (we are talking about unstructured points). The efficient compression of unstructured point-cloud usually requires to find the points that are close enough in the space, to enable utilization of the prediction paradigm. The method described in [12] establishes a correlation between the points by using an octree-based space partitioning schema. This algorithm predicts the location of the next point with regard to the approximated planes in the octree cells. A similar spatial hierarchy is used in the algorithm of Huang et al. [13]. It also supports the compression of scalar values attached to the points. Both, the point coordinates and the attached scalar values are represented by using an average value in the surrounding tree cells. Other approaches apply prediction vectors to estimate the coordinates of the next point in the stream. For example, the algorithm proposed in [13] predicts the location of the next point using the vector between the last two points before the observed one. Besides this, they introduced additional rules for rotating of the prediction vector.

The algorithm proposed in [14] is one of the earliest, aimed for LiDAR data compression. It utilizes the Delaunay triangulation [15] for planes approximation and the wavelet transform to update the values of the points coordinates. This method is very efficient with regard to the compression ratio but, simultaneously, it is slow and lossy. The algorithm proposed by Isenburg [16] is also intended to compress LiDAR data. At the moment, it is accessible only as a demo program, while the implementation details are not known to the public. However, in terms of compression ratio, even more efficient algorithm was presented by Mongus and Žalik in [17] (execution version is available at [18]). To seed-up the compression process, a multithreaded implementation is suggested in this paper.

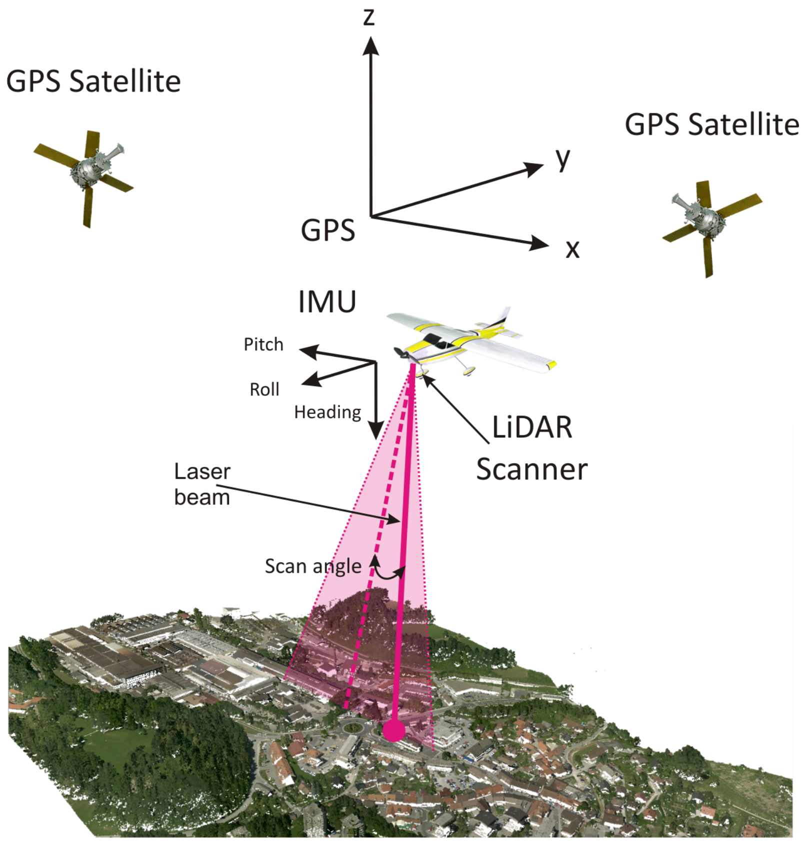

Figure 1: Gathering of airborne LiDAR data.

Because of this, this method will be briefly explained in the next section.

3. LOSSLESS LIDAR DATA COMPRESSION ALGORITHM

The considered method for compression of LAS files uses domain-specific information about the LiDAR data gathering. In this way, the correlation between points can be established without an additional space partitioning. The method works in three steps:

1. Points are encoded with a predictive coding scheme.
2. The errors in the prediction are coded with the variable-length-coding (VLC).
3. VLC values are compressed with arithmetic coder (AC) and stored in the output file.

In the prediction model, three prediction rules are used to estimate the positions of the points, while constant prediction rule is used do predict their scalar values (details are given in [17]):

- Constant prediction rule presumes that the values of the same attribute of two successive points are the same. Because the consequent scalar values of LiDAR points are often very similar, the constant prediction rule is highly efficient. On the other hand, the positions of successive points are never the same. Therefore different prediction rules are proposed for them.
- Prediction rule for *x*-coordinate uses the average distance between *x*-coordi-nates of the successive points. In addition, the deviation of the last few samples (100 have been used in our case) are used, and by the help of the linear interpolation, included in the final prediction.
- Prediction rule for *y*-coordinate exploits the difference between successive points *x*-coordinate to predict the *y*-coordinate of the coded point. Usually, the large distance in *x*-coordinate results in a large distance in *y*-coordinate, too. Thus, the history of the coded points is searched to find two successive points with a similar difference in *x*-coordinate. If such points are found, their differences in *y*-coordinate should match, too. If the match is not found, the linear approximation between previous *y*-coordinates is used for prediction.
- Prediction rule for *z*-coordinate applies a similar concept as the prediction for *y*-coordinate. However, it uses both *x* and *y* coordinates.

Because predictions made by described prediction model are accurate, the absolute values of prediction errors are small and the VLC can be efficiently applied. In the VLC step, the description byte is added to each value, where the information about sign and length (in bytes) of each value are stored. Thus, the zero bytes can be removed from each value. Furthermore, description bytes, as well as non-zero bytes of the same importance, are arranged in the separated byte streams. Each of those streams is then independently compressed by arithmetic coding (AC) [11] and stored to output file. This independency makes the algorithm suitable for multiprocessor programming, introduced in the next section.

4. MULTITHREAD APPROACH FOR LIDAR DATA COMPRESSION

As already stated, most of the LiDAR point attributes are coded independently and therefore they can be processed simultaneously. Thus, by the use of multithreading, multi-core central process units (CPU) and direct memory access (DMA) [19] can be exploited for a considerable reduction of the time needed for LiDAR data compression. In the Fig. 2, the scheme of the proposed approach is presented.

A set of LiDAR points, which represents an input in our method, is rearranged into a set of data-streams. Each data-stream contains the values of the same LiDAR point attribute. Because only predictions of *xyz*-coordinates are mutually dependent (prediction for *y* is dependent on *x* values and prediction for *z* is dependent on *x* and *y* values), one thread is created for each data-stream and one for processing *xyz*-coordinates. It is obvious that most of the CPU time is used to predict *xyz*-values and thus, corresponding thread should have a higher priority. In this way, the processing of the data-streams is more synchronized and consecutively less CPU time is lost in the last synchronisation step. However, after the predictive coding step for the *xyz*-coordinates is completed, VLC can be preformed for *xyz*-coordinates simultaneously as well. Therefore, two more threads are created (for *y*-coordinates and *z*-coordinates), while the priority of the current one (*x*-coordinates) is set back to normal. Furthermore, in the VLC step, each data-stream is further split into four byte-streams. Because each byte-stream can be handled by AC independently, additional threads are created (see Fig. 2).

In the last step, the threads are synchronised and compressed data is stored in the output file. Since it is time consuming to write the data to the out file, a pipeline is created for using DMA to speed up the data storing. Thus, in the first step, each of the threads enters the queue for the AC. One after another, byte-streams are accepted by the AC and the corresponding threads are killed. When processing of a byte-stream is terminated by AC, the data is passed to the DMA. Because DMA does not use the CPU for writing, the CPU can process the next byte-stream. In this way, the arithmetic coding and the data storing are performed simultaneously, reducing the overall data compression time. However, because we cannot predict which of the threads finishes the first, the order of the byte-streams is stored in the output file, too. Nevertheless, the number of threads is limited and the space needed to store the order of the byte-streams is irrelevant in regards to the total file size.

5. RESULTS

The presented multithreading approach was tested against the original method accessible at [18]. Both implementations were done in C# under MS Windows 7 Profession. The time efficiency of multithreading approach is demonstrated on five LAS files that contain different number of points. Tests were performed on computer system with Intel Core 2 Quad CPU Q6600 2.45 GHz, 4 GB of RAM. The results are presented in Table 1.

From the results we can see that the proposed multithreaded approach is, for over 70%, faster than the original approach. The CPU utilization is increased from 24% to 94%. The original method has not been designed for multi-core CPUs. Because of this, its utilisation cannot be higher than *1/N*, where *N* is the number of CPU cores. Thus, in our case (*N=4*) the maximal utilization is 25%. When dealing with multi-core CPUs utilization, the distribution of computational workload should be as equal as possible. With the presented approach, the CPU utilization confirms that almost optimal workload distribution has been achieved. The difference be-

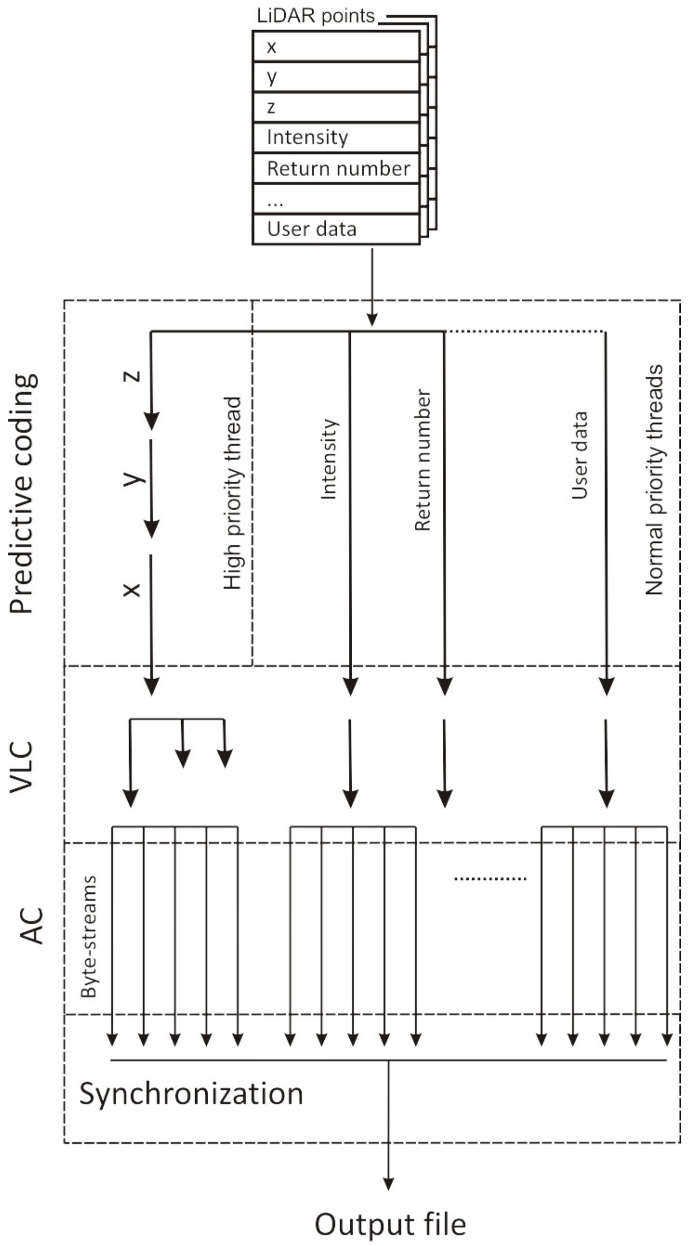

Figure 2: Multithread aproach for LiDAR data compression.

Table 1: Time efficiency of lossless LiDAR data compression algorithms

	LAS File 1	LAS File 2	LAS File 3	LAS File 4	LAS File 5
Original file size (kB)	64.194	101.649	108.037	175.304	429.743
Compressed file size(kB)	9.887	9.289	14.339	20.250	40.455
Number of points	2.345.998	3.581.247	3.951.030	6.411.089	15.716.290
Original method	14.85 sec.	19.95 sec.	24.35 sec.	36.82 sec.	142.80 sec.
Multithreaded method	4.30 sec.	5.66 sec.	7.18 sec.	10.35 sec.	39.60 sec.
Time reduction	**71.0%**	**71.6%**	**70.55%**	**71.9%**	**72.3 %**

tween optimal utilisation and the achieved utilisation is due to the additional calculations needed for synchronizations of threads and thread scheduling. Because the thread synchronisation time is equal regardless to the file size, slight increase in the time reduction can be noticed when larger files are compressed. The only exception is LAS File 3, where data density is particularly low and worse compression ratio is achieved as well.

6. CONCLUSION

A new multithread compression schema for lossless LiDAR data compression has been presented in this paper. The proposed approach exploits multi-core CPU and DMA of modern computer systems to speed up the data compression. It has been shown that in these ways, the total compression time has considerably decreased, due to the multithreading. When dealing with multiple threads, different threads share the same cash. Thus, by proper synchronisation, less time-costly data transfers are required [19]. Furthermore, if one thread gets a lot of cache misses, other threads can still employ free computer resources that would otherwise be idle. In this way the overall compression time has been reduced over 70%.

7. REFERENCES

[1] F. Ackermann, "Airborne laser scanning - present status and future expectations," *ISPRS Journal of Photogrammetry and Remote Sensing*, vol. 54(2-3), pp. 64–67, 1999.

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

[3] M. R. Belmont, "Application of non-uniform to uniform data mapping to: Shallow angle lidar with the introduction of independent variable techniques," *Signal Processing*, vol. 87(10), pp. 2461–2472, 2007.

[4] A. Wehr and U. Lohr, "Airborne laser scanning - an introduction and overview," *ISPRS Journal of Photogrammetry and Remote Sensing*, vol. 54(2-3), pp. 68–82, 1999.

[5] D. F. Maune, *Land development handbook (3th ed.)*, chapter Aerial mapping and surveying, pp. 877–910, McGraw-Hill Professional, 2008.

[6] H. Lee, K. C. Slatton, B. E. Roth, and W.P. Cropper, "Prediction of forest canopy light interception using three-dimensional airborne lidar data," *International Journal of Remote Sensing*, vol. 30(1), pp. 189–207, 2009.

[7] S. Coveney, A. S. Fotheringham, M. Charlton, and T. McCarthy, "Dual-scale validation of a medium-resolution coastal dem with terrestrial lidar dsm and gps," *Computers & Geosciences*, vol. 36(4), pp. 489–499, 2010.

[8] J. L. Guerrero-Rascado, B. Ruiz, G. Chourdakis, S. A. Raymetrics, G. Georgoussis, and L. Alados-Arboledas, "One year of water vapour raman lidar measurements at the andalusian centre for environmental studies (ceama)," *International Journal of Remote Sensing*, vol. 29(17-18), pp. 5437–5453, 2008.

[9] American Society for Photogrammetry and Remote Sensing (ASPRS), "Las specification," avaliable at http://www.asprs.org/.

[10] G. Taubin and J. Rossignac, "Geometric compression through topological surgery," *ACM Transactions on Graphics*, vol. 17(2), pp. 84–115, 1998.

[11] D. Salomon, M. Giovanni, and D. Bryant, *Data Compression: The Complete Reference*, Springer, 2009.

[12] R. Schnabel and R. Klein, "Octree-based point-cloud compression," in *Eurographics Symposium on Point-Based Graphics*, M. Botsch and B. Chen, Eds. 2006, pp. 147–156, Eurographics Association.

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

[14] B. Pradhan, S. Kumar, S. Mansor, A. R. Ramli, and A. R. B. M. Sharif, "Light detection and ranging (lidar) data compression," *KMITL Journal of Science and Technology*, vol. 5, pp. 515–526, 2005.

[15] B Žalik, "An efficient sweepline delaunay triangulation algorithm," *Computer-Aided Design*, vol. 37, pp. 1027–1038, 2005.

[16] M. Isenburg, "Lastools: converting, viewing, and compressing lidar data in las format," avaliable at: http://www.cs.unc.edu/∼isenburg/lastools/.

[17] D. Mongus and B. Žalik, "Efficient method for lossless lidar data compression," *International Journal of Remote Sensing*, vol. 32(9), pp. 2507 – 2518, 2011.

[18] D. Mongus and B. Žalik, "Las compression algorithm," avaliable at: http://gemma.uni-mb.si/lascompression/.

[19] M. Herlihy and N. Shavit, *The Art of Multiprocessor Programming*, Morgan Kaufmann, 2008.

ABOUT THE AUTHORS

Domen Mongus is a Ph.D. student at University of Maribor, Faculty of Electrical Engineering and Computer Science,Department of computer science. His contact email is `domen.mongus@uni-mb.si`.

Ph.D. Denis Špelič is a researcher at University of Maribor, Faculty of Electrical Engineering and Computer Science,Department of computer science. His contact email is `denis.spelic@uni-mb.si`.

Borut Žalik is a professor at University of Maribor, Faculty of Electrical Engineering and Computer Science,Department of computer science. His contact email is `zalik@uni-mb.si`.