Extended Delta Compression Algorithm for Scanning LiDAR Raw Data Handling

Conference Paper · November 2019

3 authors:

Ievgeniia Maksymova
Infineon Technologies

Christian Steger
Graz University of Technology

Norbert Druml
Infineon Technologies

Extended Delta Compression Algorithm for Scanning LiDAR Raw Data Handling

Ievgeniia Maksymova [1,2], Christian Steger [2] and Norbert Druml [1]

***Abstract*— LiDAR sensors are widely used in robotic applications for depth image acquisition, and generate tens of frames per second that are transformed into a 3D point cloud. Each of these frames contains a pixel matrix of an entire field of view. In scanning LiDAR a frame is acquired sequentially and requires big on-chip memory arrays or a high speed interface for data to be transferred to ECU. In this paper we compare efficiency of existing lossless algorithms when applied to raw LiDAR data, and propose a lossless compression algorithm that is intended to reduce a required on-chip memory and/or relax speed requirement on an interface. The simulation results showed that achievable compression rate is $> 38\%$ independently from distance to target or receiving circuit resolution.**

I. Introduction

Due to advances and minituarization of the LiDAR technology, the sensor's application field has been expanded to robotics and unmanned aerial vehicles (UAV) [1]. LiDARs are widely used in mobile robots for a corridor mapping in a railway track inspections [2], pedestrian detection in urban scenarios [3], collision avoidance in hazardous and difficult accessible environments [4]. In recent years, also then automotive industry turned its attention to this technology in attempt to reach higher autonomous driving levels [5].

Along with an application field expansion, technical requirements of LiDAR sensors have also changed by addressing longer range ($> 200m$), higher resolution and frame rate, and lower production costs ($<$ \$200). A scanning LiDAR that uses MEMS mirrors for beam steering could potentially fulfill all aforementioned requirements. However, scanning long-range LiDAR sensors acquire frame data sequentially and require an intermediate memory before an entire frame can be processed into a 3D point cloud. The size of this memory is driven by the sensor's parameters such as range, resolution, an oversampling factor etc. [6]

3D point cloud computing is typically done using a complex DSP within the LiDAR sensor. This approach allows to produce results faster, requires big memory arrays for an intermediate data storage, custom implemented special functions and a long development cycle. Nevertheless, there are scenarios (e.g. sensor fusion) when raw data from several sensors shall be fused with the help of an external electronic computing unit (ECU). In this case the frame data shall be transferred from the sensor to ECU with a very high speed interface (Gb/s) and the internal memory array requirements could be relaxed. Both approaches have the same issue - high power consumption caused by either a big memory, or a very high speed interface.

In this paper we compare various low-level lossless compression algorithms that could be used in LiDAR sensors for a memory size reduction or improved bandwidth utilization. The algorithm comparison performed using several factors such as an implementation complexity, compression speed and effectiveness. Finally, a simple, yet effective compression algorithm is proposed that could be beneficial for battery-powered robots, e.g. UAV, and systems with a high frame rate requirements such as autonomous vehicles.

II. State-of-the-art

There are two main components in raw LiDAR data. First, an echo signal with a certain amplitude that depends on the target reflectance and the distance to a target. Second, a noise signal that can have several sources, e.g. an ambient noise or a shot noise of a receiving circuitry. Figure 1 shows two sets of measured LiDAR raw data that are done with different objects in the field of view with a distance up to 150 meters.

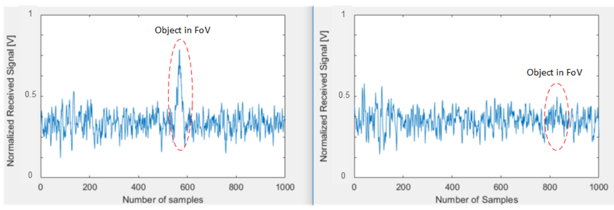

Fig. 1. Measured raw LiDAR data of targets with a high reflectance (left) and a low reflectance (right)

The dictionary-based compression algorithms require multiple iterations for building up a dictionary that later has to be saved or sent along with the compressed data for a successful data decompression. Thus, these algorithms are not effective for the LiDAR raw data compression.

[1]Infineon Technologies Austria AG {ievgeniia.maksymova; norbert.druml}@infineon.com

[2]Institute of Technical Informatics, TU Graz steger@tugraz.at

The authors would like to thank the Austrian Federal Ministry for Transport, Innovation and Technology which funded the ODeLiA project under the grant agreement no 860032.

2nd Workshop on Proximity Perception
2019 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS 2019), Macau, China

Golomb-Rice Coding (GRC) compresses a data stream X of a length N using a tunable parameter M that is chosen to be a power of 2. The coding is performed by representing every value X_i as a sequence of 0's followed by 1 [7]. The number of 0s is defined by an integer result value of X_i divided by M. The reminder of this division is converted to $log_2(M)$ bits and adjusted to the 0's sequence.

In Delta Encoding (DE), the first value in a data stream is always written out the same as the original value. Following values are representing the difference between the current and the previous value in the data stream (delta). The codeword is S bits long to allow a delta ranging from -2^{S-1} to $2^{S-1}-1$. When a delta is too large to be represented by S bits, the original current value is stored as a codeword.

Symmetric Segmented Delta encoding algorithm (SSD) is a modified version of the delta encoding algorithm that computes deltas as the absolute value of the difference, which is then segmented in one of four segments [8]. Each of four segments has a defined $base$ that is used as a threshold for a proper segment placement, and an $offset$ that represents the reminder between the delta and the $base$. The output data is then stored as a delta sign bit, followed by a segment number and an offset.

As it can be observed in Figure 1, LiDAR data contains a DC component. Thus, removing this DC component through a careful selection of algorithm specific parameters (M, S, $base$ and $offset$) might improve resulting compression ratios, as less bits will be required to represent the same data.

III. Extended Delta Encoding Algorithm

The proposed Extended Delta Compression algorithm (EDC) is a derivative of the delta encoding algorithm that is used for data stream compression. The compression flowchart of EDC is shown in Figure 2. Every sample in the original data stream is a bw bit wide positive number. The maximum bit width reduction of a data stream sample bx is defined before the compression begins. The first value is always written out the same as the original value, while all computed delta values are compared with the 2^{bw-bx} value. In case a signed delta value is outside of the range, an overshoot is detected; the position of this overshoot is stored along with the full 2^{bw-1} bits wide delta value. Once all data has been encoded, the overshoot information is prefixed to the deltas as shown in Figure 3, where K is the number of occurred overshoots, $P-array$ are their positions, and $r-array$ are delta values.

The decompression flowchart is shown in Figure 4. First, the number of overshoots that occurred during compression (K) and their positions (P array) are extracted from the data stream. Second, the current sample's index is compared with values in P-array and the bit width of the current delta is defined. Then the difference between adjacent samples is calculated, resulting in lossless reconstruction of sampled data. As compressed data is a data stream, the pointer $plsb$ is used for tracking the position of the next sample and is incremented on a cycle base with the bitwidth of the current sample.

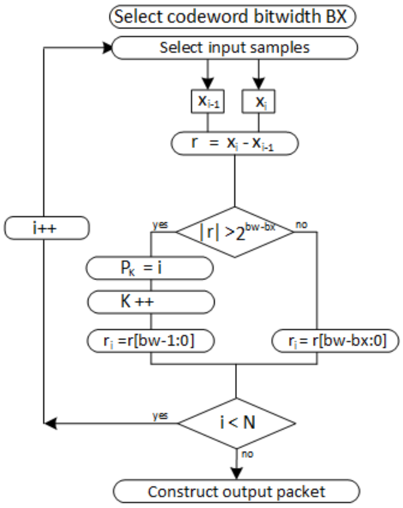

Fig. 2. Compression flowchart of the extended delta encoding

Fig. 3. Compressed data packet format.

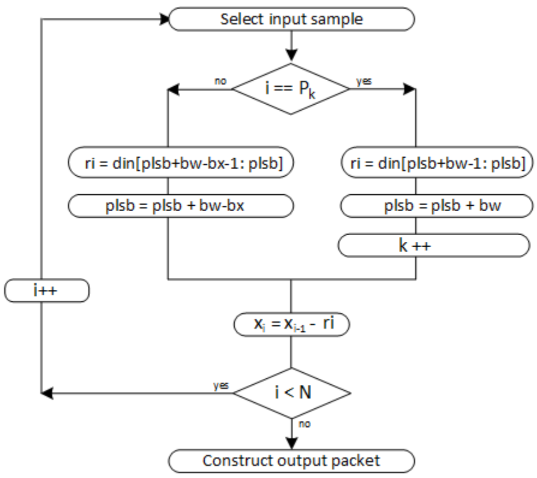

Fig. 4. Decompression flowchart of the extended delta encoding

IV. Simulation Results

Aforementioned algorithms are implemented in MATLAB and evaluated through a set of characteristics: compression ratio, compression speed, and algorithm complexity. The simulations were done using datasets acquired with a LiDAR demonstrator in a laboratory setup for collision avoidance. The following conditions were used: distance to a target ranging from 1m to 100m, weather factors set to no light,

2^{nd} Workshop on Proximity Perception
2019 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS 2019), Macau, China

bright light and rain, targets with reflectance levels ranging from 10% to 95%. The data has been digitized using ADC with two arbitrary ADC bit width values, 6 bits and 10 bits, to evaluate algorithms efficiency. Table I presents the evaluation results.

The compression ratio is used to evaluate algorithm performance and can be defined as

$$CompressionRatio = (1 - \frac{CompressedSize}{OriginalSize}) * 100\%$$

The speed of compression is evaluated in cycles required to complete the compression process of a data stream with N samples, assuming a cycle per operand. The complexity is defined by the amount of required hardware (e.g. buffers, adders, multipliers etc.) for an algorithm implementation.

TABLE I
COMPRESSION ALGORITHMS EVALUATION RESULTS

Algorithm	Speed in cycles	Complexity	C.Ratio ADC 6b	C.Ratio ADC 10b
DE	7*N	+++	19%	19%
GRC	7*N	++	-1.2%	1%
SSD	14*N	+	16%	28%
EDC	9*N	++	47%	38%

The Golomb-Rice Compression algorithm showed a very poor performance as the data DC value is far distanced from any of power of 2. The Delta Encoding showed a steady compression ratio independently from the data bit width. The Symmetric Segmented Delta encoding showed a tendency to have a better performance with high resolution data. The Extended Delta compression algorithm showed a good compression results ($> 38\%$) independently from the data bit width or the data set.

V. DISCUSSIONS AND CONCLUSIONS

In recent years, mobile robots, UAV and autonomous vehicles are more often using LiDAR sensors for depth imaging and range detection. LiDAR data acquisition, processing and transfer are resource demanding, which is a limiting factor for many compact and battery-powered applications. Through incorporating a raw data compression in a scanning LiDAR sensor, the sensor's memory demands could be optimized and complex high speed interfaces for data transfer between the sensor and the ECU could be eliminated. In this work the Extended Delta Compression algorithm was presented that is capable to compress LiDAR raw data by more than 38%, and the effectiveness of several low-level compression algorithms was compared.

As a continuation of this work, the proposed EDC algorithm will be used for a design optimizations of our LiDAR demonstrator, such as optimizing its interface bandwidth utilization. Moreover, as it has a good theoretical compression ratio and it is possible to calculate the maximum length of a compressed stream, it might be used to reduce on-chip memory needed to store a frame.

REFERENCES

[1] R. Moss, P. Yuan, X. Bai, E. Quesada, R. Sudharsanan, "Low-cost compact MEMS scanning LADAR system for robotic applications," Proc. of SPIE, vol.8379, 2012

[2] M.T. Andani, A. Mohammed, A. Jain, M. Ahmadian, "Application of LIDAR technology for rail surface monitoring and quality indexing," Journal of Rail and Rapid Transit, vol.232, 2018

[3] C. Premebida, O. Ludwig, U. Nunes, "LIDAR and vision-based pedestrian detection system," Journal of field of robotics, vol.26, 2009

[4] P.-L. Richard, N. Pouliot, S. Montambault , "Introduction of a LIDAR-based obstacle detection system on the LineScout power line robot," IEEE/ASME, ISBN: 978-1-4799-5736-1, 2014

[5] Corporate Partnership Board, "Automated and autonomous driving. Regulations under uncertainty," OECD/ITF 2015

[6] I. Maksymova, C. Steger, N. Druml, "Review of LiDAR Sensor Data Acquisition and Compression for Automotive Applications," Proceedings 2018, vol.2, pp.852-856, 2018

[7] P. Kalode and R. Khandelwal, "Test Data Compression based on Golomb Coding and Two-value Golomb Coding," SIPIJ, vol.3, no.2, pp.171-184, April 2012

[8] Shu-Ping Liang, Yi-Yu Liu, "Symmetric Segmented Delta Encoding for Wireless Sensor Data Compression," SASIMI Proceedings, 2016

2nd Workshop on Proximity Perception
2019 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS 2019), Macau, China