Simple Variable-Length Encodings for GPS Trajectory Compression

Hirohito Sasakawa[1] **and Hiroki Arimura**[1]

[1]Graduate School of IST, Hokkaido University
N14, W9, Sapporo 060-0814, Japan

Large amounts of GPS trajectory data from mobile sensor devices have appeared in recent years. In this paper, we propose simple variable-length encoding schema for lossless compression of GPS-trajectory data. The first frequency-based coding, called the *truncated ETDC*, is built on ETDC. The second group of encodings, called the *signed difference encoding*. These are the extention of the integer sequence encoding for inverted lists. They encode trajectories as pairs of non-monotonic integer sequences combined with fixed prefix codings, e.g. γ-coding and Rice coding. Experimental results show that our proposed methods have usefulness for compression of trajectory data in compression ratio.

1 Introduction

By the rapid progress of sensor devices, massive amount of trajectory data have emerged for the last decade. Enormous volumes of trajectory data brings about heavy burdens for storage. To overcome this difficulty, many trajectory compression algorithms have attracted increasing attention [3]. In this paper, we propose simple variable-length encoding schema for GPS-trajectory data of moving objects for lossless compression.

The first method is proposed based on a frequency-based symbol coding, called the *truncated ETDC*, base on the ETDC (end-tagged dense code) method (Brisaboa *et al.* [1]). In this method, we first partition a given space domain of trajectory $m \times m$ grid and encodes each grid by any frequency-based compression, e.g. Huffman or ETDC, and then successively encode each points' location in the grid by binary number of a fixed number of bits.

The second method is proposed based on a popular difference-based coding for integer sequences, called the *signed difference encoding*. This coding can be combined with various fixed prefix codings, such as γ-coding [7] and Rice coding [7], resulting in a range of point sequence encodings. We also generalize this encoding to the difference of differences of point sequences, and refer to it as the *2nd-order signed difference encoding*.

Finally, we ran experiments on real world trajectory data. In the experiments, the proposed methods are competitive in compression ratio to existing solutions.

2 Variable-Length Encodings for Trajectory Data

In this section, we propose our variable-length encodings for trajectory data. The first one is a frequency-based point encoding, called the truncated ETDC, and the second one is a difference-based sequence encoding, called the signed difference encoding. For $i \leq j$, the notation $[i, j]$ denotes the discrete interval $\{i, i+1, \ldots, j\}$.

2.1 Trajectory data

In the original GPS-data, the coordinates of a point are usually represented by a pair (lat, lng) of real numbers, called the *latitude* and *longitude*.

Throughout this paper, we assume that the coordinates of points are normalized to points in the grid $U = [0, u]^2$ for some integer $u \geq 0$ and each coordinate (x, y) are transformed to the r-digit integers, using predetermined *resolution* parameter r of coordinates where $r > 0$.

Then, a *trajectory* of length $n \geq 0$ on the grid U is a sequence

$$s = (x_i, y_i)_{i=1}^{n} = \langle (x_1, y_1), \ldots, (x_n, y_n) \rangle \in [0, u]^2 \tag{1}$$

of two-dimensional points, where $p_i = (x_i, y_i)$ is a two-dimensional point on U for every $i = 1, \ldots, n$.

2.2 Frequency-based point encoding

Frequency-based encoding method, e.g. Huffman coding [4] is the compression method that assigns the codeword to symbol whose length is shorter to more frequent symbols. ETDC (End-Tagged Dense Code) [1], is also a frequency-based coding method, achieves a good compression ratio on natural language text. However, a potential problem to apply frequency-based coding to trajectory data is the number of distinct points in a set of trajectories S. If the set of distinct points is too large, it makes difficult to capture the frequency distribution and the storage for the codeword dictionary may consume large space (See [2] for similar arguments).

From this observation, we propose the following frequency-based encoding, called the *truncated ETDC*. Suppose that each original point in an input set S is encoded as a pair of binary numbers $x = b_1 \cdots b_\ell$ and $y = c_1 \cdots c_\ell \in \{0, 1\}^\ell$.

- We first form a single binary number $z = b_1 c_1 \cdots c_\ell b_\ell \in \{0, 1\}^{2\ell}$ of length 2ℓ bits by splicing two bit-sequences x and y.
- For a fixed integer $1 \leq k \leq \ell$, we split the bit-sequence z into the prefix $p = b_1 c_1 \cdots c_k b_k$ of length $2k$ bits and the suffix $q = b_{k+1} c_{k+1} \cdots c_\ell b_\ell$ of length $m = 2(\ell - k)$ bits.
- We encode the prefixes p's by a ETDC encoding for points, while we use the suffixes q's as it is, that is as equi-length codewords of length $2(\ell - k)$ bits.

The parameter k controls the balance between the dictionary size and the encoded text length. If k is larger, then the encoded text will become succincter, but the dictionary will become larger, or *vice vasa*.

2.3 Signed difference-based sequence encoding

In the application of compression of ascending order integer sequence, such as inverted list compression, we use the property that the monotonicity of the sequence. An ascending sequence of integers $s = \langle x_1, \ldots, x_n \rangle$ is transformed into the *gap sequence* $t = \langle d_1, \ldots, d_n \rangle$, where the i-th gap $d_i = x_i - x_{i-1}$ is the *difference* for every i, and $d_0 = 0$. Then, the gap sequence is encoded by *fixed prefix codes* such as Elias γ-code and δ-code for positive integer encoding [7]. Such a fixed prefix code is independently determined from data, and sometimes is optimal for some data distribution. For any positive integer $x \in \{1, 2, \ldots\}$, we denote by $unary(x)$ and $mbin(x)$ the *unary encoding* [5] and *minimum binary enconding* [5] of x, respectively.

The *γ-code* represents a number x as a unary code for $len = \lfloor \log x \rfloor + 1$ followed by a binary code x' for x from which the leading one bit is removed, that is, $\gamma(x) = unary(len) \cdot x'$, where the total length is $\ell \simeq 1 + 2\lfloor \log x \rfloor$ bits. The *δ-code* is a extension of γ-code in which the prefix (unary) part is

Table 1: Description of the `TaxiInt` dataset.

data	number of elements	range of x (min, max)	range of y (min, max)	average number in x	average number in y
`pos`	9,936,666	(1, 10999)	(1, 12499)	7714.56	9208.80
`dif`	9,936,666	(-8848, 10998)	(-11323, 12499)	111.85	166.37
`dif2`	9,936,666	(-14610, 10998)	(-19511, 12499)	22.66	33.52

coded by the γ-code and the rest is the same in binary, that is, $\delta(x) = \gamma(len) \cdot x'$, where total length is $\ell \simeq 1 + \lfloor 2 \log \log x \rfloor + \lfloor \log x \rfloor$ bits.

The *Golomb-code* encodes input number x using parameter b as follows. The first step is to compute the quotient $q = \lceil x \rceil / b$ and the remainder $r = x \mod b$. Then, we encode the quotient q coded by an unary code followed by the remainder r coded by a minimum binary code, that is, $Golomb(x) = unary(q) \cdot mbin(r)$. The *Rice-code* is a special case of the Golomb-code, in which the parameter b is chosen to be 2^k for some integer k.

To apply this idea with fixed prefix code to trajectory data, a problem is that the difference of two consecutive values can be *negative*. Thus, we propose two different sign encoding method.

The first is the *signed encoding* method. In this method, we encode a value x_i into the *sign* $s_i \in \{0, 1\}$ followed by the absolute value $t_i = abs(x)$, where the value $x_i \geq 0$ is represented by $s_i = 1$ and $x_i < 0$ is by $s_i = 0$. That is, the signed version of the codeword for x_i is $c(x_i) = s_i \cdot \gamma(t_i)$ for the γ-code. The second is the *sign run-length encoding* method. In this method, we encode a sequence of signs s_i by the run length encoding followed by sequence of absolute value t_i.

Now, we propose a difference-based trajectory encoding, called the *1st-order signed difference encoding*. We encode a trajectory $s = (x_1, y_1) \cdots (x_n, y_n)$ into the bit-string

$$c(s) = c(d_1^x) \cdot c(d_1^y) \cdots c(d_n^x) \cdot c(d_n^y), \tag{2}$$

where $d_i^x = x_i - x_{i-1}$ and $d_i^y = y_i - y_{i-1}$ for every $1 \leq i \leq n$.

We can further extend this idea of difference sequences into the differences of consecutive difference values. In the *2nd-order point difference encoding*, we encodes the sequence s above into

$$c2(s) = c(dd_1^x) \cdot c(dd_1^y) \cdots c(dd_n^x) \cdot c(dd_n^y) \tag{3}$$

where $dd_i^x = d_i^x - d_{i-1}^x$ and $dd_i^y = d_i^y - d_{i-1}^y$ for every $1 \leq i \leq n$.

We can interpret 1st and 2nd-order difference as the *velocity* and *acceleration* of a moving object. Hence, for GPS-trajectories with small and constant sampling time, we can expect that in a second-order difference sequence, small values will appear frequently.

3 Experimental Results

Datasets and setting: In Table 1, we show the description of the datasets. We used the *Cabspotting* taxi GPS trajectory data[1] [6]. This dataset originally contains approximately 400MB of GPS traces, where points is in a pair of floating numbers, and then preprocessed by us to obtain 105,497,460 byte of GPS traces, contains 9,936,666 points, called `TaxiInt`, in pair of decimal integers in ASCII. The space per point of `TaxiInt` is 85.1163 bits.

The experiments were run on a 3.6GHz Intel Xeon processor with 32GB RAM operating on Ubuntu Linux OS 12.04. The code is in Python and running on Python 2.7.3 interpreter.

[1] `http://cabspotting.org/`

Table 2: Bit length of a point coordinates, prefix and suffix part of truncated ETDC codewords on the `TaxiInt` dataset.

K	total (bit)	suffix (bit)	prefix (bit)
70	34.00	14.78	19.22
100	34.00	13.74	20.26
500	34.00	9.10	24.90
800	34.00	7.74	26.26
1000	34.00	7.10	26.90

Results: We compared the proposed encoding schema on the `TaxiInt` dataset. Signed binary and runlen binary is a baseline method of this experiment. In these methods, we calculate the maximum number m of in the coordinates on the dataset, then we encode the dataset by $\lceil \log m \rceil$ bit fix length binary code. For Rice codes [7], we vary the parameter k for 8, 10, 12, and 14. Truncated ETDC algorithm has parameter $K = 2^k$ for the number of divisions, and K =200, 500, and 800 were used for K, k is the prefix bit length. Table 2 shows the bit length of the prefix and suffix part of truncated ETDC by varying the parameter K. Table 3 shows the comparison of the proposed compression method on `TaxiInt` dataset the previous section in terms of the compression ratio (second column of each data, Ratio) and bits per point (third column of each data, BPP).

For dataset `pos` for absolute positions, the winner was the runlen binary coding, and the runlen Rice12 coding follows, which reduces the bit per point from 85.12 to 27.29 (bits/p) having compression ratio 32.87%. For dataset `dif` for differences, i.e. velocity-like values, the clear winner was signed Rice8 coding, which has 21.39 (bits/p) and compression ratio 25.76%. For dataset `dif2` for 2nd difference, the winner was signed Rice8 coding, which has bit per point 22.05 (bit/p) and compression ratio 26.56%.

These shows differencial approach is effective for trajectory compression. On the contrary, `dif2` approach is not so effective.

4 Conclusion

In this paper, we studied simple variable-length encodings for lossless compression of GPS-trajectory data. Experimental results show that our proposed methods truncated ETDC and the signed difference encoding well perform combined with some existing encoding methods. In future, compressed pattern matching for trajectory data will be an interesting research problem.

Acknowledgements

The authors would like to thank Takuya Kida, Satoshi Yoshida, Miyuki Nakano, Roberto Konow and Susana Ladra for fruitful discussion and comments on trajectory compression.

References

[1] N. Brisaboa, E. Iglesias, G. Navarro, and J. Paramá. An efficient compression code for text databases. In *Proc. 25th European Conference on Information Retrieval Research (ECIR)*, LNCS 2633, pages 468–481, 2003.

[2] N. R. Brisaboa, S. Ladra, and G. Navarro. DACs: Bringing direct access to variable-length codes. *Information Processing & Management*, 2012.

Table 3: Encoded size, for three encoding schema of `TaxiInt` dataset.

Data	`pos`			`dif`			`dif2`		
Method	Size (KB)	Ratio	BPP (bits/p)	Size (KB)	Ratio	BPP (bits/p)	Size (KB)	Ratio	BPP (bits/p)
original data	105,497	1.0000	84.94	75,877	0.7192	61.09	78,170	0.7410	62.93
signed binary	37,161	0.3523	29.25	34,817	0.3300	27.40	37,263	0.3532	29.33
signed gamma	67,458	0.6394	53.09	31,877	0.3022	25.09	33,430	0.3169	26.31
signed delta	51,115	0.4845	40.23	29,149	0.2763	22.94	30,489	0.2890	24.00
signed Rice6	347,100	3.2901	273.18	31,288	0.2966	24.62	34,818	0.3300	27.40
signed Rice8	105,739	1.0023	83.22	**27,174**	**0.2576**	**21.39**	**28,020**	**0.2656**	**22.05**
signed Rice10	49,136	0.4658	38.67	30,186	0.2861	23.76	30,408	0.2882	23.93
signed Rice12	38,757	0.3674	30.50	34,842	0.3303	27.42	34,892	0.3307	27.46
trunc. ETDC 70	40,588	0.3847	31.94	38,290	0.3632	30.15	43,185	0.4093	33.99
trunc. ETDC 100	36,933	0.3501	29.07	40,745	0.3862	32.07	40,550	0.3844	31.91
trunc. ETDC 500	42,539	0.4032	33.48	33,990	0.3222	26.75	33,238	0.3151	26.16
trunc. ETDC 800	46,571	0.4414	36.65	36,018	0.3414	28.35	35,645	0.3379	28.05
trunc. ETDC 1K	48,947	0.4640	38.52	36,890	0.3497	29.03	36,856	0.3494	29.01
runlen binary	**34,677**	**0.3287**	**27.29**	42,015	0.3983	33.07	54,841	0.5198	43.16
runlen gamma	64,974	0.6159	51.14	32,470	0.3078	25.56	35,763	0.3390	28.15
runlen delta	48,631	0.4610	38.27	30,265	0.2869	23.82	34,004	0.3223	26.76
runlen Rice6	344,654	3.2669	271.26	34,029	0.3226	26.78	42,388	0.4018	33.36
runlen Rice8	103,264	0.9788	81.27	31,397	0.2976	24.71	38,439	0.3644	30.25
runlen Rice10	46,654	0.4422	36.72	35,895	0.3402	28.25	43,689	0.4141	34.39
runlen Rice12	36,274	0.3438	28.55	42,041	0.3985	33.09	51,038	0.4838	40.17

[3] M. Chen, M. Xu, and P. Franti. Compression of GPS trajectories. In *Data Compression Conference (DCC), 2012*, pages 62–71. IEEE, 2012.

[4] D. A. Huffman. A method for the construction of minimum-redundancy codes. *Proceedings of the IRE*, 40(9):1098–1101, 1952.

[5] A. Moffat and A. Turpin. *Compression and Coding Algorithms*. Kluwer, 2002.

[6] M. Piorkowski, N. Sarafijanovoc-Djukic, and M. Grossglauser. A Parsimonious Model of Mobile Partitioned Networks with Clustering. In *The 1st International Conference on Communication Systems and Networks (COMSNETS)*, January 2009.

[7] I. H. Witten, A. A. Moffat, and T. C. Bell. *Managing gigabytes: compressing and indexing documents and images*. Morgan Kaufmann, 2 edition, 1999.