Spatiotemporal Sampling for Trajectory Streams

Mohamed K. El Mahrsi, Christine Potier, Georges Hébrail, Fabrice Rossi
Télécom ParisTech, BILab, 46 rue Barrault, Paris 75013, France
{khalil.mahrsi | christine.potier | georges.hebrail | fabrice.rossi}@telecom-paristech.fr

ABSTRACT

In this paper, we present a summarization technique adapted to trajectory streams. The Spatiotemporal Stream Sampling (STSS) algorithm is a single-pass sampling technique that takes advantage of both the spatial and temporal dimensions inherent to these streams in order to reduce their processing and storage costs.

Categories and Subject Descriptors

H.2.8 [**Database Management**]: Database Applications

General Terms

Algorithms

Keywords

Trajectory streams, spatiotemporal, sampling, summaries.

1. INTRODUCTION

The advent of location-aware devices (GPS, PDAs, etc.) makes it possible to easily track and monitor the current position of individuals, vehicles, etc. This process, however, generates a tremendous amount of transient, unbounded *data streams* [1] of time-stamped positions, leading to computational and storage challenges. Therefore, efficient compression techniques are needed in order to construct good quality summaries of incoming data on-the-fly and within affordable computational costs.

Progress in trajectory compression is mainly inspired by advances in the field of line simplification, cartographic generalization and time series compression. Existing work includes, mainly, the *TD-TR* and *OPW-TR* algorithms (both proposed by Meratnia et al. in [2]) as well as the *Thresholds* and *STTrace* algorithms (presented in [3]). The aforementioned techniques are either resource-efficient (mainly CPU and memory) but have no control over the compression quality or vice versa . Based on this observation, we introduce a new sampling technique, called *Spatiotemporal Stream Sampling* (*STSS*), that combines both aspects by fastly and intelligentely discarding items from the original stream while providing deterministic compression error bounds.

SAC'10 March 22-26, 2010, Sierre, Switzerland.

2. PROBLEM FORMULATION

The *trajectory* T of an object moving over the Euclidean space is a sequence of time-stamped positions $P_i\,(t_i, x_i, y_i)\,, i \in \{1, ..., n\}$. The pair (x_i, y_i) is the position of the object and t_i is the time instant at which it was recorded.

A *compressed trajectory* (or *summarized trajectory*) T_c, on the other hand, is a subseries of the point series belonging to the original trajectory T. In other words, we are interested in compression techniques that downsample the original trajectory by deleting unnecessary data points while preserving its essential movement trends and features.

The following aspects should be taken into consideration while designing trajectory streams compression algorithms: (i) *Online processing*: in a streaming context, only algorithms capable of processing data incrementally on-the-fly can be used; (ii) *Time complexity*: very fast algorithms are needed in order to cope with very large data streams; (iii) *Memory complexity*: preferably, only a small memory window is kept on the data; and (iv) *Error bounds*: errors due to the lossy compression process should be small and, preferably, parametrically adjustable.

3. THE STSS ALGORITHM

The *Spatiotemporal Stream Sampling* (STSS) algorithm is based on the intuitive idea of linear prediction. The algorithm tries to capture the currently observed motion pattern of the moving object. As long as this predicted motion pattern is respected, data points can be dropped. This is put into practice using the following *motion function*:

$$t \mapsto (x, y) = (a_x t + b_x, a_y t + b_y) \qquad (1)$$

Given two points, say P_j and P_k ($t_j < t_k$), a_x and b_x (similarly a_y and b_y) are calculated as follows:

$$a_x = \frac{x_k - x_j}{t_k - t_j};\;\; b_x = x_j - a_x t_j = x_k - a_x t_k \qquad (2)$$

Initially, the motion function is determined from the first two points in the streamed trajectory (first point is automatically selected to be part of the compressed trajectory). For each new data point P_i, a predicted point P'_i is calculated by applying the motion function to the timestamp of P_i. If the distance between P_i and P'_i is inferior to a given threshold d_{Thres}, the point is considered as well-predicted. In case P_i is inaccurately predicted, the last well-predicted point (P_{i-1}) is inserted into the compressed trajectory, and the motion function is updated using P_{i-1} and P_i. The pseudo code of STSS is given in Alg.1.

Algorithm 1 STSS(Trajectory S, Threshold d_{Thres})

insert P_1 in T_{C}
calculate the motion function f using P_1 and P_2
for all $P_i = (t_i, x_i, y_i)$ in S **do**
  calculate P'_i {Predicted position}
  if $d(P_i, P'_i) > d_{\text{Thres}}$ **then**
    insert P_{i-1} in T_{Comp}
    update f using P_{i-1} and P_i
  end if
  keep P_i in buffer for the next iteration
end for

Since STSS acts as a simple filter, it benefits from low computational and memory costs: time complexity (per positional update) and memory complexity are $O(1)$. Interestingly enough, the algorithm also succeeds in providing a guaranteed error bound (of $2d_{\text{Thres}}$) between the compressed trajectory and the original one:

PROOF. Let $[P_s, P_e]$ be a segment that STSS constructed (Fig. 1).The interpolation P''_i of each intermediate point P_i (where $t_s < t_i < t_e$) on $[P_s, P_e]$ obeys the following equation:

$$P''_i = \left(t_i, \frac{x_e - x_s}{t_e - t_s}(t_i - t_s), \frac{y_e - y_s}{t_e - t_s}(t_i - t_s)\right) \qquad (3)$$

Since prediction was conducted using P_s and P_2, the predicted position P'_i is expressed as:

$$P'_i = \left(t_i, \frac{x_2 - x_s}{t_2 - t_s}(t_i - t_s), \frac{y_2 - y_s}{t_2 - t_s}(t_i - t_s)\right) \qquad (4)$$

However, since P'_e is also a linear prediction (of P_e) using P_s and P_2, P'_i can also be expressed as:

$$P'_i = \left(t_i, \frac{x'_e - x_s}{t_e - t_s}(t_i - t_s), \frac{y'_e - y_s}{t_e - t_s}(t_i - t_s)\right) \qquad (5)$$

Since $\frac{t_i - t_s}{t_e - t_s} \leq 1$, we can easily derive the following equation from (3) and (5):

$$d(P'_i, P''_i) \leq d(P_e, P'_e) \qquad (6)$$

According to the triangle inequality, and using (6):

$$d(P_i, P''_i) \leq d(P_i, P'_i) + d(P'_i, P''_i) \leq d(P_i, P'_i) + d(P_e, P'_e) \qquad (7)$$

Both P_e and P_i being well-predicted by STSS:

$$d(P_i, P'_i) \leq d_{\text{Thres}} \text{ and } d(P_e, P'_e) \leq d_{\text{Thres}} \qquad (8)$$

Equations (7) and (8) yield the final result:

$$d(P_i, P''_i) \leq 2d_{\text{Thres}} \qquad (9)$$

□

4. EXPERIMENTAL EVALUATION

We compared the STSS, OPW-TR and STTrace algorithms using the trucks dataset[1] which is composed of 276

[1]available at: http://www.rtreeportal.org/

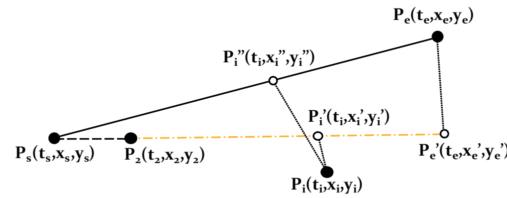

Figure 1: Proof of the guaranteed error bound.

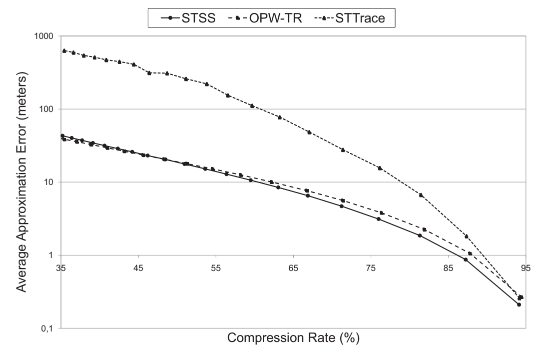

Figure 2: Average approximation error vs. compression rate

real trajectories. Due to space limitations, we only present the comparison of approximation errors at different compression rates. As shown in both Fig. 2 and Fig. 3, STSS outperforms STTrace (which is of comparable complexities) and provides results comparable to those of OPW-TR (which has the, higher, complexity of $O(n)$ per incoming point).

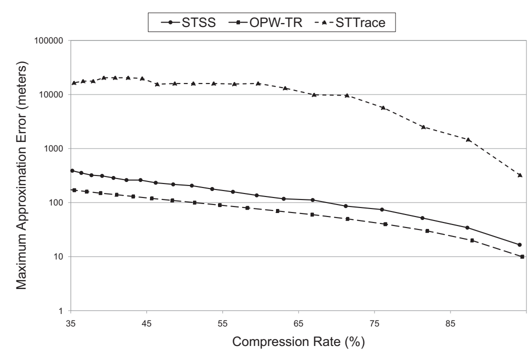

Figure 3: Maximum approximation error vs. compression rate

5. REFERENCES

[1] L. Golab and M. T. Özsu. Issues in data stream management. *SIGMOD Record*, 32(2):5–14, 2003.

[2] N. Meratnia and R. A. de By. Spatiotemporal compression techniques for moving point objects. In *9th Int. Conf. on Extending Database Technology (EDBT'04)*, pages 765–782, 2004.

[3] M. Potamias, K. Patroumpas, and T. Sellis. Sampling trajectory streams with spatiotemporal criteria. In *Proc. of 18th Int. Conf. on Scientific and Statistical Database Management (SSDBM'06)*, pages 275–284, 2006.