Prof. Renée J. Miller
Editor-in-Chief
The VLDB Journal

Dear Prof. Miller,

Attached please find a revised version of our submission to the VLDB Journal, *One-Pass Trajectory Simplification Using the Synchronous Euclidean Distance*.

The paper has been substantially revised according to the referees' comments. In particular, we have (a) added the intuition and key ideas of our solution, (b) elaborated the proofs and clarified the necessities of propositions, (c) added a set of new experiments to compare SED with PED, (d) moved all the proofs to an appendix, and (e) refined the entire paper to improve its readability.

We would like to thank all the referees for their thorough reading of our paper and for their valuable comments.

Below please find our detailed responses to the comments.

Response to the comments of Associate Editor.

[**AE**] *The paper suffers from over-formalization that loses the intuition of the solution. In particular, the crucial sections 3 and 4 are too dense and fail to convey to readers the key technical ideas. The paper also contains numerous typos and grammatical errors. The revision should address these shortcomings. Consider providing an overview/outline of the approach first, rather than laying out a sequence of propositions. Moving some of the proofs to an appendix might be an option to improve readability.*

(1) We have provided the intuition and the key technical ideas of the solution in Sections 3 and 4.

(2) We have fixed typos and grammatical errors raised by reviewers and carefully done proof-readings.

(3) We have also moved all proofs to an appendix.

Response to the comments of Reviewer 1.

[**R1W1**] *Some parts in the paper can be improved for readability.*

Yes, we have elaborated the paper. Please refer to our responses to R1C1, R1C2, R1C3, R1C4, R1C5 and R1C6 for details.

[**R1W2**] *The experimental section needs some clarification on some points.*

Yes, we have clarified the issues you mentioned. Please refer to our responses to R1C7, R1C8 and R1C9 for more details.

[**R1W3**] *The paper has some minor grammatical mistakes.*

Yes, we have fixed these grammatical mistakes, and refined our paper.

[**R1C1**] *On page 2, the first column is a little bit confusing. Going back and forth between SED and PED might confuse the readers. The second paragraph discusses optimal PED-based algorithms that run in $O(n^3)$ and $O(n^2)$. Then it says that the $O(n^2)$ algorithm is not applicable to SED distance. The third paragraph then describes some sub-optimal algorithms for SED-based simplification. The fourth paragraph talks again about PED-based algorithms and mentions some linear-time algorithms but it's clear whether these algorithms are optimal or approximate. It is not clear either why do we need to reiterate over PED-based algorithms if they are not applicable to the addresses problem.*

Yes! We have now removed the introduction of algorithms using PED, and directly discussed the algorithms using SED in the second paragraph. We have also merged the original second and third paragraphs, as they both talk about the algorithms using SED. Thanks for raising this issue!

[R1C2] *Page 2 second column, it was a bit confusing what you mean by effective and efficient. I understood later that effective refers to the compression ratio and efficient refers to the running time. It would help to highlight this earlier or use clearer terms, e.g., faster instead of efficient.*

In our work, *effective* or *effectiveness* refers to the compression ratio, and *efficient* or *efficiency* refers to the running time, respectively. Now we have directly used terms "compression ratio" and "running time" throughout the entire paper. Thanks!

[R1C3] *The last few paragraphs in Section 1 mention too many numbers that the reader cannot digest at that early point (all the comparison numbers to existing algorithms with different datasets.) I suggest mentioning only two or three numbers, e.g., the average speed up or the average compression ratio.*

Yes, we have changed to the average values. Thanks!

[R1C4] *In Section 2.1, it is not clear whether the Directed line segment (L), its length, and angle are in the 2D or 3D space. I assume it is in the 2D space because the angle (theta) is measured in the 2D space (x,y) but you would better highlight this part.*

(1) In the work, a directed line segment $\mathcal{L}$ is defined as $\overrightarrow{P_sP_e}$, which represents the closed line segment that connects the start point P_s and the end point P_e. In this definition, a data point P is in a x-y-t 3D space, hence, the directed line segment (L) can also defined in the x-y-t 3D space.

(2) However, when we talk about the length and angle of a directed line segment ($\mathcal{L}$), it is truly measured in the x-y 2D space. More accurately, when a line segment $\mathcal{L}$ is projected on a x-y 2D space, we also use $|\mathcal{L}|$ and $\mathcal{L}.\theta \in [0, 2\pi)$ to denote the length of $\mathcal{L}$ in the x-y 2D space, and its angle with the x-axis of the coordinate system (x, y). That is, when a directed line segment $\mathcal{L} = \overrightarrow{P_sP_e}$ is treated as a triple $(P_s, |\mathcal{L}|, \mathcal{L}.\theta)$, we are talking about the projection of $\mathcal{L}$ in the x-y 2D space.

We have clarified this in the definition of directed line segment ($\mathcal{L}$) in Section 2.1. Thanks for pointing this out!

[R1C5] *In Section 3.1, the definition of spatio-temporal cones (C) mentions "w.r.t. a Point P_s" is this the starting point P_s or any arbitrary point? If it is always the starting point P_s, then you might better rephrase it to "w.r.t. the starting point P_s."*

Yes, it is "the" starting point. We have fixed it. Thanks!

[R1C6] *In Section 3.4, it could be better if you mention the key idea of the proposed algorithm to help the readers understand it. For example, you can mention that the key observation is that every segment in the two polygons being intersected has to originate from one of the m edges of the regular polygon. Then you can mention how this helps skipping multiple segments at a time.*

Yes, we have rewritten the first two paragraphs of Section 3.4, and highlighted the key idea of the proposed Fast Regular Polygon Intersection algorithm in the second paragraph. Essentially, it is due to "..., every segment in the two polygons being intersected has to originate from one of the m edges of the regular polygons and also the geometric similarity of the regular polygons, we may advance edge $\overrightarrow{A}$ or $\overrightarrow{B}$ multiple steps at a time, instead of one step at a time of the convex polygon intersection algorithm CPolyInter." Then an example follows to illustrate this idea. Thanks for your advice!

[R1C7] *In the experiments section, Figures 12 and 16 are confusing as they compare the compression ratio and error of the proposed algorithm with and without the improved polygon intersection algorithm. It looks like the proposed improvement only affects the running time but produces the same output which makes it unclear how it might affect the quality of the results. It looks like the numbers in the two figures are exactly the same with both intersection algorithms (and they had to be the same). I suggest making two lines only CISED-S and CISED-W similar to figures 13-15 and 17-19.*

The motivations of developing the Fast Regular Polygon Intersection (RPolyInter) algorithm are (1) it should

improve the running time of computing the common intersection of inscribed regular polygons, and (2) it should have the same outputs (compressed trajectories) as the Convex Polygon Intersection (CPolyInter) algorithm.

Tests Exp 1.1, Exp 2.1 and Exp 3.1 are also designed to demonstrate: (1) our algorithm RPolyInter runs faster than algorithm CPolyInter (see Fig.20 and Fig.21), and (2) it has the same compression ratios and average errors as the later (see Fig.12 and Fig.16). Thus, we believe that it would be better to preserve the current lines as shown in Fig.12, Fig.16, Fig.20 and Fig.21.

We have clarified this in tests Exp 1.1, Exp 2.1 and Exp 3.1. Thanks for pointing this out!

[R1C8] *For readability of the results, I suggest using consistent point types in all the figures. Currently, some point types are reused with different algorithms which can be confusing, e.g., CISED-S-C in Figure 12 is the same as CISED-W in Figure 13 and CISED-W-C in Figure 12 is the same as SQUISH-E in Figure 13.*

We have used consistent point types in all the figures. Thanks for your nice advice!

[R1C9] *In Figure 14, it is not clear how the proposed CISED-W algorithm produced a compression ratio that is better than the optimal algorithm. You need to comment on this or correct it.*

By allowing data interpolations, algorithm CISED-W extends the radius of the base circles of the spatio-temporal cones from $\epsilon/2$ in CISED-S to ϵ in CISED-W by Proposition 2. Thus, CISED-W has better compression ratios compared with CISED-S. Its compression ratios are "comparable with" the optimal algorithm, and some times are even better than the latter, *e.g.,* in Fig.14-(1) Service-Car, CISED-W is better than the optimal algorithm.

We have clarified this in Exp 1.3. Thanks!

[R1C10] *The paper has some minor language issues that can be easily corrected with a proofread.*
- "an one" → "a one"
- "to fast the ..." → "to speed up the ..."
- "they forms a ..." → "they form a ..."
- "in the executing of ..." → "in the execution of ..."

Fixed! Thanks!

Response to the comments of Reviewer 2.

[R2W1] *The contribution of the paper seems to be limited; the key idea is to approximate circles intersection with polygons intersection to determine when to start a new line segment in the summarized trajectory*

Theorem 8 clearly tells the contribution of our paper, and the techniques developed are non-trivial at all.

(1) Algorithms CISED-S and CISED-W are the first two one-pass trajectory simplification algorithms using SED. They are fast and have good compression ratios. Hence, this is an important contribution to the community.

(2) From the view of key techniques, it is the first local distance checking method for trajectory compression using SED. A local distance checking method is the key to develop a one-pass trajectory simplification algorithm, and to our knowledge, currently, there are only two effective local distance checking methods, the *sector intersection* mechanism used in SI [40] (or Sleeve [11]) and the *fitting function* approach used in OPERB [15], and these methods are both PED specific, and not applicable to SED. Indeed, *the design of an effective local synchronous distance checking approach is non-trivial*.

(3) We also developed a method that finds the approximate common intersection of n circles in a linear time and a constant space. Though we develop this method for the local synchronous distance checking, it may have potential usages as the circle intersection is a basic functionality.

[11] Z. Zhao and A. Saalfeld. Linear-time sleeve-fitting polyline simplification algorithms. In Proceedings of Auto-Carto, pages 214-223, 1997.

[15] X. Lin, S. Ma, H. Zhang, T. Wo, and J. Huai. One-pass error bounded trajectory simplification. PVLDB, 10(7):841-852, 2017.

[40] C. M. Williams. An efficient algorithm for the piecewise linear approximation of planar curvers. Computer Graphics and Image Processing, 8:286-293, 1978.

[R2W2] *The paper is over formalized with many propositions, some of which may be incorrect.*

Please refer to our responses to R2C1, R2C3-R2C7 for details!

[R2W3] *There are too many typos and grammatical errors.*

We have fixed these grammatical mistakes, and refined our paper.

[R2C1] *Sections 3 and 4, which constitute the core of the paper, are written as a series of propositions, some of which are not needed or are not easy to grasp. The authors could easily remove some and also add some intuition to others.*

(1) We have moved all the detailed proofs to an appendix as suggested by the Associate Editor.

(2) We have removed the original Proposition 1, which has been incorporated into the original Proposition 2 (currently Proposition 1).

(3) We have also added intuitions or explanations to all propositions.

(4) We have also refined our proofs.

Thanks for the suggestion!

[R2C2-1] *In terms of the setting of the problem, the authors did not discuss the device range error which varies from device to device and may affect the algorithms.*

We have discussed this in the second last paragraph of Section 1.

Yes, these devices have range errors, which leads to data quality issues of trajectory data [27, 44]. However, we strongly believe that the problem is beyond the scope of this study. We focus on lossy simplification of trajectory data only in this article as many existing studies do, *e.g.,* [15, 20, 23, 40].

[27] D. Pfoser and C. S. Jensen. Capturing the uncertainty of moving-object representations. In SSD, 1999.

[44] A. Zfle, G. Trajcevski, D. Pfoser, M. Renz, M. T. Rice, T. Leslie, P. L. Delamater, and T. Emrich. Handling uncertainty in geo-spatial data. In ICDE, 2017.

[R2C2-2] *In addition, it seems that the authors assume that the devices move in uniform speed which may not be realistic.*

This indeed questions the rationale of SED, which was proposed in [20]. Yes, SED assumes that the object is moving linearly in a uniform speed *between two adjacent data points* of a compressed trajectory. Your observation is right, and yes, it may not be realistic. However, when data points are sampled at a reasonable rate, the uniform speed assumption is acceptable, and it does alleviate the issue for spatio-temporal queries discussed in the Introduction. Furthermore, SED has become a very important distance metric for trajectory simplification [4, 41].

We have further clarified this in the definition of *synchronized points* in Section 2.1.

[4] M. Chen, M. Xu, and P. Franti. A fast multi-resolution polygonal approximation algorithm for GPS trajectory simplification. IEEE Trans. Image Processing, 21(5):2770C2785, 2012.

[20] Nirvana Meratnia and Rolf A. de By. Spatiotemporal Compression Techniques for Moving Point Objects. EDBT, 2004.

[41] Dongxiang Zhang, Mengting Ding, Dingyu Yang, Yi Liu, Ju Fan, and Heng Tao Shen. Trajectory Simplification: An Experimental Study and Quality Analysis. PVLDB, 11 (9): 934-946, 2018.

[R2C3] *Now looking at some of the propositions. The proof of proposition looks incorrect, take as an example a moving device along the y-axis only over the time axis. In this case,* $p'_{s+i}.x - P_s.x = 0$ *while* $p'_{s+i}.y - P_s.y$ *is not*

zero.

This talks about the proof of the original Proposition 1.

Yes, the proof is not strict enough. We said that "The intersection point P'_{s+i} satisfies that $P'_{s+i}.t = P_{s+i}.t$ and $\frac{P'_{s+i}.t-P_s.t}{Q.t-P_s.t} = \frac{P_{s+i}.t-P_s.t}{Q.t-P_s.t} = \frac{|\overrightarrow{P_sP'_{s+i}}|}{|\overrightarrow{P_sQ}|} = \frac{P'_{s+i}.x-P_s.x}{Q.x-P_s.x} = \frac{P'_{s+i}.y-P_s.y}{Q.y-P_s.y}$. Hence, by the definition of synchronized points, we have the conclusion." As you pointed out, it is possible that the denominator $|\overrightarrow{P_sQ}| = 0$ or $Q.x - P_s.x = 0$ or $Q.y - P_s.y = 0$. We should have discussed the special case that $Q.x - P_s.x = 0$ or $Q.y - P_s.y = 0$, though they do not affect the correctness of the proposition.

The above proof can be fixed as follows: "The intersection point P'_{s+i} satisfies that (1) $P'_{s+i}.t = P_{s+i}.t$, (2) $P'_{s+i}.x = P_s.x + w \cdot (Q.x - P_s.x)$, and (3) $P'_{s+i}.y = P_s.y + w \cdot (Q.y - P_s.y)$, where $w = \frac{P'_{s+i}.t-P_s.t}{Q.t-P_s.t} = \frac{P_{s+i}.t-P_s.t}{Q.t-P_s.t}$. Hence, by the definition of synchronized points, we have the conclusion."

Thanks for raising this issue!

[R2C4] *Except if I am missing something, projecting a cone over a plane is not necessarily a circle.*

In our case, the projection of a *spatio-temporal cone* over a plane $P.t - t_c = 0$ ($t_c > P_s.t$) is certainly a circle.

Recall in the definition, the *synchronous circle* of a data point P_{s+i} *w.r.t.* an error bound ϵ, denoted as $\mathcal{O}(P_{s+i}, \epsilon)$, is a circle on the plane $P.t - P_{s+i}.t = 0$ such that P_{s+i} is its center and ϵ is its radius. Then, given the start point P_s and an error bound ϵ, the *spatio-temporal cone* of a data point P_{s+i} *w.r.t.* the point P_s and the error bound ϵ, denoted as $\mathcal{C}(P_s, \mathcal{O}(P_{s+i}, \epsilon))$, is an oblique circular cone such that point P_s is its apex and the synchronous circle $\mathcal{O}(P_{s+i}, \epsilon)$ of point P_{s+i} is its base. Note that the base, *i.e.,* the synchronous circle $\mathcal{O}(P_{s+i}, \epsilon)$, is on the plane $P.t - P_{s+i}.t = 0$ which is parallel to any other plane $P.t - t_c = 0$. Hence, the projection of a cone $\mathcal{C}(P_s, \mathcal{O}(P_{s+i}, \epsilon))$ on a plane $P.t - t_c = 0$ ($t_c > P_s.t$) is surely a circle.

We have further clarified this in the definition of *cone projection circles* in Section 3.2.

[R2C5] *Looking at Proposition 4, if you take only two polygons with m edges, their intersection could have more than m edges. Or are the authors doing the intersection differently.*

Yes, the intersection of two general m-edges polygons may produce a polygon with more than m edges. However, we approximate a circle by a specialized polygon, *i.e.,* a fixed rotation and m-edges inscribed regular polygon $\mathcal{R}(V, E)$, where (1) $V = \{v_1, \ldots, v_m\}$ is the set of vertexes that are defined by a polar coordinate system, whose origin is the center P of $\mathcal{O}^c$, satisfying $v_j = (r, \frac{(j-1)}{m}2\pi)$, $j \in [1, m]$ and (2) $E = \{\overrightarrow{v_m v_1}\} \bigcup \{\overrightarrow{v_j v_{j+1}} \mid j \in [1, m-1]\}$ is the set of edges that are labeled with the subscript of their start points.

For fixed rotation and m-edges inscribed regular polygons, all edges in the same edge groups E^j ($1 \leq j \leq m$) are in parallel (or overlapping) with each other by the above definition of inscribed regular polygons. This design ensures that the intersection of these polygons has no more than m edges.

We have further refined our proof and explained the motivation and intuition of this design in the first paragraph of Section 3.3.

Thanks for pointing out this!

[R2C6] *Propositions 6 and 7 are not really needed as they follow up from the way your algorithms function.*

Indeed, Propositions 6 and 7 guarantee the correctness of the Fast Regular Polygon Intersection algorithm FastRPolyInter. These two propositions serve as the basis of the extra *advance rules* that we apply to speed up the process of inscribed regular polygons intersection. Hence, they are critical for algorithm FastRPolyInter.

We have rewritten the first two paragraphs of Section 3.4 to clarify this.

[R2C7] *Theorem 8 is not really needed.*

Theorem 8 clearly tells the contribution of this study to the community.

To our knowledge, no one-pass and error bounded line simplification algorithm using SED has been developed in the community yet. Hence, one may wonder whether there exists an effective, one-pass and error bounded line simplification algorithm using SED. Theorem 8 is the positive answer to this question, proved by providing two algorithms CISED-S and CISED-W.

[R2C8] *Referring to Exp-1.1, I wouldn't say that CISED-S is comparable to DPSED, a* 10% *difference is not small.*

In Exp-1.2, the compression ratios of CISED-S are on averaged 109.5% of SQUISH-E and DPSED on all test datasets. For example, when $\epsilon = 40$ meters, the compression ratios of algorithms CISED-S and DPSED are (16.1%, 5.8%, 3.9%, 3.6%) and (14.8%, 5.4%, 3.4%, 3.4%) on the four datasets, respectively. Indeed, the compression ratios of algorithm CISED-S are a bit worse than DPSED. However, when consider the great benefit of running time of CISED-S, we would like to say that the compress ratio of CISED-S is "comparable" with DPSED.

Yes, a 10% difference is not that small. To deal with this controversy, we have replaced the words "comparable with" with "close to" in Exp 1.2 and also the rest of the paper. Thanks for pointing out this!

[R2C9] *In almost all the experiments, the behavior of the different algorithms across all datasets is quite similar; the authors should have commented on this.*

Yes, we have added a comment on this in the summary (the first bullet) in Section 5.2. Thanks for the suggestion!

[R2C10] *In the summary about the experiments, the authors should have discussed, based on compression ratios, average errors, and running time, which of the proposed algorithms should be used in practice under which circumstances. In particular, looking at the average errors, how acceptable are these for the applications that will use the proposed compression algorithms. Being fast with high compression is good, but if the quality is not that good then no one will use these algorithms.*

Yes, a guideline would be good. However, it is extremely hard to provide such a guideline, as different applications many have different requirements to reach a balance among multiple metrics, normally running time, compression ratio and error.

For example, from the experiences of the CarStream [Zhang2017] project (an industrial data-processing system for chauffeured car services that has connected over 30,000 vehicles in more than 60 cities, deployed applications like "tracking the historical trajectories of cars", "mapping trajectories on road networks", "analyzing the trips, mileages and travel patterns of cars", and so on), we find from our scenarios, the importance ranking of the metrics from high to low is running time, compression ratio and error. Say, the running time and compression ratio are more important than the error in these scenarios. Nevertheless, in other applications, one may find that the error is more important. Hence, the choice of compression algorithms is application specific.

Hence, we have only provided a brief guideline for choosing an algorithm from the views of running time, compression ratio and average error, respectively, in Section 5.2.5, as follows.

1) If the running time is the first-level consideration or algorithms are run in resource-constrained devices, then the one-pass algorithms, i.e., CISED-S and CISED-W, are surely the best choices, and they have pretty good compression ratios at the same time.

(2) If the compression ratio is the priority, then algorithm CISED-W and the optimal algorithm are the selections, followed by algorithms DPSED and CISED-S.

(3) When considering error, SQUISH-E is a good choice because it has a relative small average error. Alternatively, we can also use DPSED or CISED-S by setting a smaller error bound compared with SQUISH-E.

Thanks for pointing out this!

[Zhang2017] Mingming Zhang, Tianyu Wo, Tao Xie, Xuelian Lin and Yaxiao Liu. CarStream: An Industrial System of Big Data Processing for Internet-of-Vehicles. PVLDB, 10 (12), 2017.

[R2C11] *Few examples of the typos and the grammatical errors:*
a. In th abstract "algorithms have are been"
b. Abstract "comparable with and 19.6%*"* $\rightarrow$ *The last sentence is a very hard to read.*

c. Introduction (first paragraph) "devices have been using their sensors" → devices use their sensors
d. Page 2 – Column 1 – Last paragraph "to design an one-pass LS algorithm"
e. Page 4 – Column 1 – "is no greater than"
f. Page 9 – Last paragraph – "Algorithm FastRPolyInter. The presented" → it is the first time to talk about the algorithm so how it has been presented.
g. Many single sentence paragraph throughout the paper.

Fixed! Thanks!

Response to the comments of Reviewer 3.

[R3C1] *From the example in Fig. 1, it is really difficult to tell the difference between PED and SED. It seems that SED will use more line segments due to the given constraint enforced by epsilon. However, why is the compression using SED better than that using PED? If PED is good enough, why do we care about SED? Essentially, the motivation of leaning towards SED is never clear from a practical perspective. I think it all depends on the tolerance at the application level. So I would like the authors to evaluate some real applications on top of the compressed trajectory data, to demonstrate the benefits of using SED. At least, the authors should give some convincing arguments on that.*

The second and third paragraphs in Section 1 serve as the purpose of using SED.

When compressing trajectories, the choice of PED or SED depends on the specific application requirements. For example, for a spatio-temporal query application, "*the position of a moving object at time t*", on trajectories compressed by algorithms using SED returns a point P' whose distance to the actual position of the moving object is within the bound ϵ, while the same query on trajectories compressed by algorithms using PED does not has this property. The use of PED brings better compression ratios, but it does not support applications like spatio-temporal query.

We have clarified these in the third paragraph of Section 1, with some extra explains presented in the definitions of SED and PED in Section 2.1. Thanks for raising this issue!

[R3C2] *The authors should give more intuitive explanation of the semantics of "synchronized points." Maybe this has been documented in the literature, but the authors should give sufficient explanation to make this paper self-contained.*

Given a sub-trajectory $[P_s, \ldots, P_e]$ to be simplified, a synchronized data point P'_i, $s < i < e$, is a point on line segment $\overrightarrow{P_sP_e}$ satisfying $|\overrightarrow{P_sP_e}| > 0$ and $\frac{|\overrightarrow{P_sP'_i}|}{|\overrightarrow{P_sP_e}|} = \frac{P_i.t - P_s.t}{P_e.t - P_s.t}$, which means that the object is moving from P_s to P_e at an average speed $\overline{v} = \frac{|\overrightarrow{P_sP_e}|}{P_e.t - P_s.t}$, thus its position at time t_i is the point P'_i on line segment $\overrightarrow{P_sP_e}$ having a distance of $\overline{v} \times (P_i.t - P_s.t)$ to point P_s.

We have clarified this in the third paragraph of Section 1 and in the definition of the *synchronized points* in Section 2.1, and an example of synchronized points has also been added in Example 1 in Section 2.1.

Thanks for pointing this out!

[R3C3] *Examples 1, 2, 3, 4, and 5 are not illustrative: They basically explained what happened in the corresponding figures but did not explain why. So it is not helpful if readers try to understand the algorithms by reading these examples. More details should be given. For instance, in Example 1, you can give specific coordinates to certain points so that readers can verify how the SED points are computed.*

Yes, we have add more details.

As in the response to R3C2, we have given an intuitive explanation of the semantics of "synchronized points", which is helpful for understanding the computing of the synchronized points and SED.

We have added more details for Examples 1, 2, 3, 4, and 5. (1) In Example 1, we have given coordinates to points and demonstrated the computing of a synchronized point. (2) In Example 2, we have described the meaning of

a circle in the Figure and its relation to a sector. (3) In Example 3, we have explained how an *advance rule* is activated so that edge $\overrightarrow{A}$ or $\overrightarrow{B}$ moves on a step. (4) In Examples 4 and 5, we have redrawn the circles and added some comments in the captions.

Thanks for the nice suggestion!

[R3C4-1] *Following my first comment, I would suggest at least comparing with one algorithm based on PED (e.g., your previous work [15]).*

Yes, we have provided additional tests that compare the performance of algorithms using PED vs. SED. We have tested the algorithm Douglas-Peucker using PED and SED, respectively, and the sector intersection algorithm using PED and our spatio-temporal cone intersection algorithm using SED. The results have been presented in Section 5.2.4.

Thanks for your advice!

[R3C4-2] *Also, I am not convinced by just using compression ratio as the single metric for measuring effectiveness. As I mentioned, I recommend using the compressed data to evaluate some real applications.*

Firstly, the application requirements of this work directly come from our CarStream [Zhang2017] project, an industrial data-processing system for chauffeured car services that has connected over 30,000 vehicles in more than 60 cities. In this project, trajectory data is collected from cars, transported to cloud, simplified/compressed and stored in databases, and then used for applications like tracking the historical trajectories of cars, analyzing the trips, mileages and travel patterns of cars, and so on. From this application, the metrics of compression ratios, and errors are good enough to evaluate the qualify of our algorithms.

Secondly, in the area of trajectory simplification, compression ratio, error and running time of algorithm are the major metrics [4, 14, 20]. We follow the these practices in our work.

Finally, different applications may have different requirements to reach a balance among multiple metrics. Hence, here we concentrate on the design of algorithms and the test with common metrics in this work, and let the application developers/users to decide which algorithm to to be used.

[Zhang2017] Mingming Zhang, Tianyu Wo, Tao Xie, Xuelian Lin and Yaxiao Liu. CarStream: An Industrial System of Big Data Processing for Internet-of-Vehicles. PVLDB, 10 (12), 2017.

[4] M. Chen, M. Xu, and P. Franti. A fast multi-resolution polygonal approximation algorithm for GPS trajectory simplification. IEEE Trans. Image Processing, 21(5):2770C2785, 2012.

[15] X. Lin, S. Ma, H. Zhang, T. Wo, and J. Huai. One-pass error bounded trajectory simplification. PVLDB, 10(7):841-852, 2017.

[20] Nirvana Meratnia and Rolf A. de By. Spatiotemporal Compression Techniques for Moving Point Objects. EDBT, 2004.

Your sincerely,

Xuelian Lin, Jiahao Jiang, Shuai Ma, Yimeng Zuo and Chunming Hu