Euler Histogram Tree: A Spatial Data Structure for Aggregate Range Queries on Vehicle Trajectories

Hairuo Xie
Department of CIS
University of Melbourne
xieh@unimelb.edu.au

Egemen Tanin
Department of CIS
University of Melbourne
etanin@unimelb.edu.au

Lars Kulik
Department of CIS
University of Melbourne
lkulik@unimelb.edu.au

Peter Scheuermann*
Department of EECS
Northwestern University
peters@eecs.
northwestern.edu

Goce Trajcevski†
Department of EECS
Northwestern University
goce@eecs.
northwestern.edu

Maryam Fanaeepour
Department of CIS
University of Melbourne
mfanaeepour@unimelb.
edu.au

ABSTRACT

This work addresses the problem of aggregation of trajectories data. Specifically, we propose a tree-based data structure for counting vehicle trajectories by mapping them into a set of spatial histograms with different granularities. We also present an approach for processing spatio-temporal range queries by aggregating the histograms in the query rectangles. The proposed methodology can be used for preserving the privacy of vehicle drivers by maintaining aggregated trajectory data. In addition, as we show, it can be used to handle the well-known distinct counting problem. Experimental results show that the new data structure achieves a high level of accuracy in query results and consistently outperforms the leading histogram-based approach.

Categories and Subject Descriptors

E.1 [**Data**]: Data structures; H.2.8 [**Database applications**]: Spatial databases and GIS

General Terms

Algorithms

Keywords

aggregate query, spatial histogram, hierarchical data structure

1. INTRODUCTION

This paper addresses the problem of processing a variant of range queries in spatial databases for vehicle trajectories. More specifically, we tackle the, so called, *distinct vehicle* query – which is, a query which returns the number of individual/distinct trajectories inside a given rectangle. We develop an efficient approach for the settings in which the monitored space can be partitioned into a rectangular grid and the query rectangles have axes-parallel boundaries, aligned with the boundary of the cells.

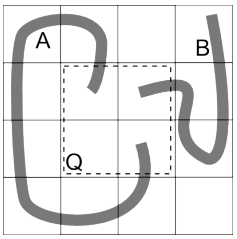

Figure 1: Two trajectories intersecting with query rectangle Q.

An example query over a 4×4 grid is shown in Figure 1. The query rectangle **Q** covers four cells. Since trajectory *A* and *B* overlap with **Q**, the answer to the range query is 2. The key observation which motivated this work is that if a particular trajectory exists which has multiple parts inside the query rectangle – i.e., multiple intersections – such trajectory should still be counted only once by the distinct vehicle query. As shown in Figure 1, trajectory *A* has two disjoint parts in *Q* – however, according to the intended semantics, it should only be counted once for the answer.

Distinct vehicle queries can be extremely useful in the domain of traffic analysis and management. Consider, for instance, a scenario in which the number of distinct vehicles in an area with observed high traffic congestion is significantly lower than the design capacity of the roads in that area. This may indicate that there is a possibility for the congestion being caused by vehicles that circle around the area – e.g., looking for parking places. By running distinct vehicle query against the congested area, traffic management authority can narrow down the possible causes of the congestion problem.

One straightforward solution for answering range queries is to construct an index that keeps a list of the unique object IDs for each cell. Using this approach, for a given a query region **Q**, we can compute the union of the IDs of the vehicles from within all the cells intersected by **Q** and then eliminate duplicate IDs. However, in addition to the fact that duplicate elimination may be an un-necessarily expensive task, the ID-based approaches may not be

*Research Supported by the NSF grant CNS 0910952
†Research Supported by the NSF grants CNS 0910952 and III 1213038, and ONR grant N00014-14-10215

Permission to make digital or hard copies of all or part of this work for personal or classroom use is granted without fee provided that copies are not made or distributed for profit or commercial advantage and that copies bear this notice and the full citation on the first page. Copyrights for components of this work owned by others than ACM must be honored. Abstracting with credit is permitted. To copy otherwise, or republish, to post on servers or to redistribute to lists, requires prior specific permission and/or a fee. Request permissions from Permissions@acm.org.

SIGSPATIAL '14, November 04-07 2014, Dallas/Fort Worth, TX, USA
Copyright 2014 ACM 978-1-4503-3138-8/14/11...$15.00
http://dx.doi.org/10.1145/2674918.2674921

suitable for privacy-sensitive applications. For example, in a traffic management application, one may not be allowed to store unique identifiers such as licence plates numbers. Our approach guarantees a high level of privacy protection as it does not need to store object IDs.

Existing research results [5] have shown the effectiveness of spatial histograms which maintain object counts for exhaustive non-overlapping areas. By answering queries using simple counts, spatial histograms are not only cost-efficient but are also privacy-aware. However, the histogram-based approaches are challenged by the well-known *distinct counting (multiple count) problem* [6, 9, 10] for spatial-temporal data sets, where an object may be counted more than once for a given query region. An object with extent is counted as many times as it intersects (or overlaps) with a query rectangle. Since histograms are mainly designed to store counts, it is difficult to identify which counts have originated from the same object so that their multiple-considerations can be eliminated from the result.

As mentioned in our example-scenario in Figure 1, trajectory *A* would be counted twice by a histogram-based approach as it has two disjoint parts inside the query rectangle. It is difficult to eliminate the redundant count because one cannot differentiate the counts from *A* and the counts from *B* during aggregation. As shown in our experiments, the distinct counting problem can significantly affect the accuracy level of the results for distinct vehicle queries.

To address this problem, we introduce a novel spatial data structure – the *Euler Histogram Tree (EHT)* which enables representing a given histogram with different levels of granularity. EHT essentially uses a hierarchy of histograms, whereby the area of a histogram at a higher level consists of multiple sub-areas corresponding to the histograms at a lower level. To answer a particular query, we traverse from the top of the hierarchy and try to use as much as possible the high(er)-level histograms. Only when a cell at a given level partially overlaps with the query rectangle, we access histograms at a lower level such that the areas of the histograms fill up the query region. This is due to the fact that for a given query region, the expectation is that there is a smaller probability that an object exists with disjoint parts at the higher level(s) of the hierarchy. This, in turn, reduces the redundant counts (which is unavoidable if histograms are aggregated at a fine granularity only).

In summary, the main contribution of this work are as follows:

- We propose the *Euler Histogram Tree* (EHT) – a novel hierarchical data structure for efficient management of aggregate trajectory data.
- We propose an efficient methodology for obtaining an answer to the distinct count variant of a range query over a given set of trajectories.
- We present experimental evaluations demonstrating that the proposed EHT offers a significant advantage over the leading spatial histograms in terms of accuracy levels.

The rest of this paper is organize as follows. On Section 2 we overview the existing related results. Section 3 discusses the details of the proposed EHT structure along with its use for processing the distinct count variant of the range query. Our experimental results are presented in Section 4 and in Section 5 we summarize the paper and outline directions for future work.

2. RELATED WORK

Hierarchical data structures for aggregate queries over spatial data have been studied for many years. For example, a tree-based

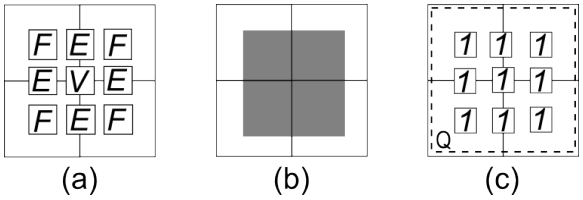

Figure 2: An example of EHs.

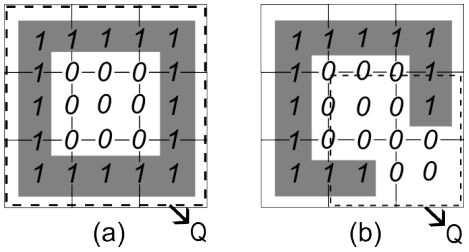

Figure 3: Errors with EHs for a query rectangle Q (each sub-figure contains an object in grey color).

structure maintaining the count of objects in a given area is proposed in [7]. The structure is suitable for an application scenario where the minimal bounding rectangles of road segments are indexed using a R-tree. Each node in the tree stores the count of cars that are in the corresponding area of the node. Queries are solved by summing up the counts while finding the nodes whose corresponding areas are within the query rectangles. CRB-tree [3] indexes aggregate information of objects using a B-tree. The aP-tree [11] aggregates spatial and temporal information, e.g., the lifespan of objects.

Our approach is closely related to a type of spatial histograms, *Euler Histograms* (EHs). EHs are grid cell partitions of a 2D space [1, 9] and maintain buckets for the inner faces of grid cells, the edges between adjacent cells and the vertices (i.e., edge intersections). Each bucket stores a count of objects. Figure 2 shows an example of EHs constructed in a space partitioned by a 2×2 grid. Sub-figure (a) shows the buckets for 4 inner faces (F), 4 edges (E) and 1 vertex (V). Sub-figure (b) shows a rectangular object in grey color on the grid. The object overlaps with all the faces, edges and vertex in the EHs. Therefore, it affects the counts in all the buckets. As shown in sub-figure (c), the counts in all the buckets are 1. Edge buckets and vertex buckets enable the connectivity for objects spanning across multiple adjacent cells. The answer to a range query can be computed as $F - E + V$, where F is the count from inner faces, E is the count from edges and V is the count from vertices. For example, the answer to the query Q in sub-figure (c) of Figure 2 can be computed as $4 - 4 + 1 = 1$. There are scenarios where EHs cannot provide accurate answers (cf. Figure 3). The left sub-figure in Figure 3 contains an object with a hole. There is no vertex count affected by the object and the query-answer is $8 - 8 + 0 = 0$. In the right sub-figure, the object leaves two parts in the query rectangle. The result from EHs is $2 - 0 + 0 = 2$. It is higher than the accurate answer, 1, and shows the effect of the distinct counting problem with EHs. Our proposed approach reduces these errors by using a hierarchy.

Distributed Euler Histograms (DEHs) [12] have been proposed as a variation of EHs. Different to EHs, DEHs can count the number of non-simple (i.e., self-intersecting) curves. Therefore, DEHs are particularly suitable for counting trajectories of moving objects.

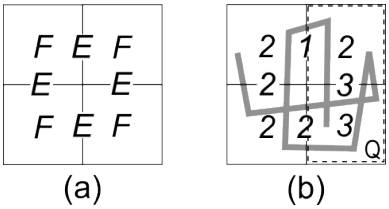

Figure 4: The left sub-figure shows the positions of faces (F) and edges (E) in DEHs. The right sub-figure shows the face counts and edge counts given a trajectory with 8 line segments. Q is a query rectangle.

If the whole trajectory is cut into disjoint sub-trajectories by the query rectangle, each of the sub-trajectories inside the query rectangle is regarded as an *entry* to the area and is counted independently. DEHs keep accurate information of entries by incrementing a face count or an edge count whenever an object reaches a face or crosses an edge. Assuming that a moving object does not cross a vertex, the number of distinct entries can be computed as $F - E$, where F is the sum of face counts and E is the sum of edge counts. Figure 4 shows an example of DEHs. The answer to the query shown in the right sub-figure can be computed as $5 - 3 = 2$, which is correct as the object makes two entries to the query rectangle **Q**. DEHs still cannot address the distinct counting problem as different entries may come from the same objects. Our experiments compare the accuracy between our approach and DEHs for answering distinct vehicle queries.

Duplicate counts in spatial-temporal databases can be reduced using sketches, which store hash values of object identifications [10]. If the hash value of an object identification exists in multiple sketches, it will not be duplicated when the sketches are combined. The approach maintains sketches in a tree-based structure, which is traversed to compute the combination of the sketches in query regions. The combined sketches are used to estimate the number of distinct objects in the regions. The methodology in [6] – similar to DEHs in the sense of maintaining information about the objects crossing borders between adjacent regions – shows advantage of histogram-based approaches over the sketch-based approaches in terms of accuracy. Thus to the best of our knowledge, DEH is the leading approach that we will compare our approach against.

An approach similar in spirit to [10] in the sense of retaining the ID information, and to our work, in the sense of hierarchical partitioning of the space – and geared towards handling the multiple count problem, was presented in [13]. However, that work was considering objects corresponding to spatial regions (with polygonal boundaries), whereas we focus on trajectories data. In addition [13] was using a hierarchical partition based on Quadtrees [8].

3. EULER HISTOGRAM TREE

The manner in which an Euler Histogram Tree (EHT) is built is very similar in fashion to a pyramid. As shown in the example EHT (Figure 5), the granularity of EHs increases as the level of the tree deepens. At each level of the EHT, counts are stored at the grid cells. In our implementation, we store four counts associated with each cell, corresponding to: *face, right edge, top edge and top-right vertex* respectivelly.

Given a particular trajectory, we update the Euler Histograms in all levels of EHT independently. A specific face count is incremented by 1 if that face overlaps with the object. Similarly, an edge count is incremented by 1 if both faces on either side of the edge

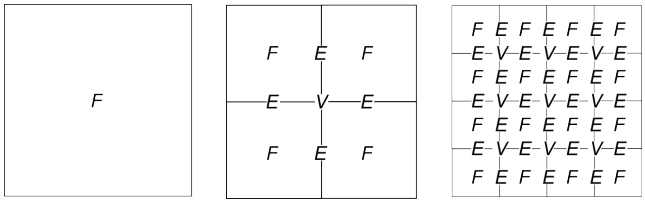

Figure 5: EHT with three levels. Face, edge and vertex counts are labelled as F, E and V, respectively.

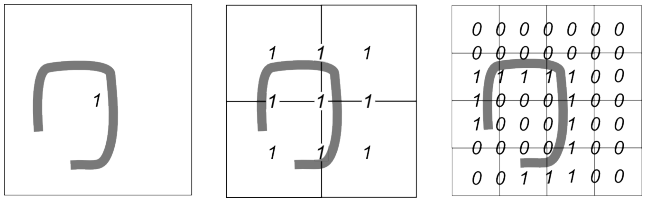

Figure 6: Three levels of EHT – the counts in each level are determined by the same trajectory.

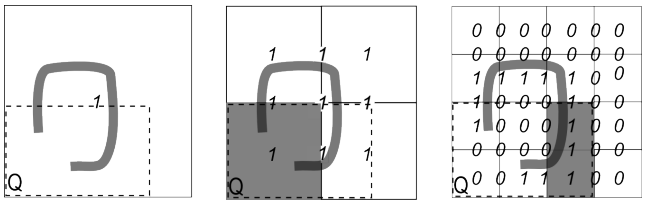

Figure 7: The progress of retrieving counts in query rectangle Q. EHT is traversed from the top level (left) to the lowest level (right). The grey areas show the cells where counts are retrieved and aggregated.

overlap with the object. A vertex count, on the other hand, is incremented by 1 if all the four faces surrounding the vertex overlap with the object. Figure 6 shows an example how the values are calculated for a given trajectory. The vertex count at the center of the whole area in the second sub-figure is 1 because the trajectory appears at all four cells surrounding the vertex. But the count for the same vertex in the third sub-figure is 0 because the trajectory does not appear in the bottom-left cell next to the vertex. It is important to note that at the very first level we can see that the initial count of 1 is the correct number of trajectories in the underlying space – this is an illustration of the key observation for basing our work.

3.1 EHT and Distinct Count Range Query

Given a query-rectangle **Q**, we use it to start a traversal of the respective EHT from the top of the pyramid. If a cell is contained in the query rectangle, we add its face count to the total face count, F. If the cell's right edge or top edge is contained in **Q**, we add the corresponding edge count to the total edge count, E. If the cell's top-right corner is contained in **Q**, we add the cell's vertex count to the total vertex count, V. We do not need to visit any child of the cells which are completely contained inside **Q**, thus reducing the effects of distinct counting problems. For cells that partially overlap with the query area, we visit their children in the pyramid and repeat the same progress. After collecting counts from all the parts in the query area, we compute the final answer as $F - E + V$. Figure 7 shows the steps of querying EHT based on the example in Figure 6. We traverse from the top of the EHT, shown in the

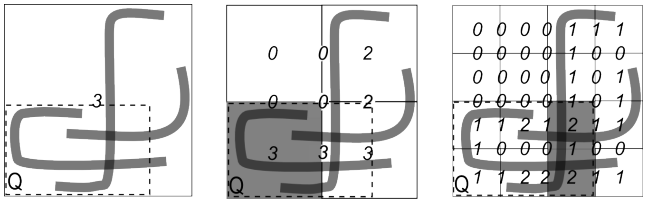

Figure 8: The progress of answering query Q in EHT with three trajectories. EHT is traversed from the top level (left) to the lowest level (right). The grey areas show the cells where counts are retrieved and aggregated.

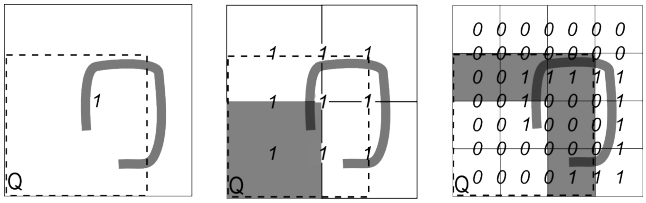

Figure 9: A scenario where EHT cannot accurately count objects in query rectangle Q. The object is a trajectory. The grey areas show the cells where counts are retrieved and aggregated.

first sub-figure. The query rectangle **Q** is matched against the cell corresponding to the whole area. Since **Q** does not contain the cell, we move to the next level, shown in the second sub-figure. On this level, the cell in grey color is completely contained in **Q**. We add the cell's face count, 1, to the total face count. Since the right edge of the cell is completely contained in **Q**, we add its count, 1, to the total edge count. The remaining part of **Q** partially overlaps with another cell. We visit this cell's children on the next level, shown in the third sub-figure. On this level, the remaining part of **Q** contains two cells, where we collect two face counts and one edge count. Each of the face counts is 1 and the edge count is 1. We add the two face counts to the total face count, which becomes 3. We add the edge count to the total edge count, which becomes 2. The total vertex count remains at 0. The final answer is computed as $3-2+0=1$.

EHT addresses the distinct counting problem in various scenarios where EHs and DEHs do not work well. This is due to the fact that an object is less likely to exist with separate parts in a query rectangle at coarse granularities. EHT uses aggregate information at coarse granularities as much as possible while EHs and DEHs only use aggregate information at fine granularities. For example, EHT can give the correct answer to the query in Figure 7. If we use EHs to solve the same query, we aggregate the counts shown in the third sub-figure of Figure 7. The answer is computed as $4-2+0=2$. Similarly, if we use DEHs, the answer is also 2. As EHs and DEHs always aggregate counts from the histograms at the finest granularity, they cannot use histograms at coarse granularities to reduce duplicate counts from the same object.

3.2 Multiple Trajectories and Imprecisions

When there are multiple trajectories in the same area, the counts in EHT are incremented from all the trajectories. Figure 8 shows an EHT with three trajectories. As the figure shows, the count for a face, edge or vertex is the total number of distinct trajectories that overlap with the face, edge or vertex.

Although EHT addresses the distinct counting problem to a large extent, there are cases where EHT cannot yield the correct answer. We show an example scenario in Figure 9, where EHT computes the answer as $4-3+1=2$. The error comes from the lowest level of EHT as the information about the connectivity between certain parts of the object is lost. However, our experimental results show that EHT can achieve high accuracy in various scenarios.

No. of Objects	% of Multi -trip Objects	No. of Trips	QRS	No. of Cells
$100-2000$	10%	2	1%	4^5
1000	$10\%-100\%$	2	1%	4^5
1000	10%	$1-10$	1%	4^5
1000	10%	2	$1\%-100\%$	4^5
1000	10%	2	1%	4^3-4^7

Table 1: Experimental settings for synthetic trajectory data.

4. EXPERIMENTS

We compare EHT with DEHs using synthetic trajectory data and real trajectory data. To simulate different scenarios, we define certain parameters in the experiments. The effects of each parameter on the accuracy of the query results is tested independently from the other parameters. For each parameter, we run tests on a range of settings. Each setting in the range is tested with 100 queries. The positions of the queries are generated uniformly at random in the whole area. We compute the accuracy of EHT and DEHs based on a benchmark approach using object IDs. The average accuracy and the deviation of accuracy are recorded for each setting.

4.1 Synthetic Trajectories

4.1.1 Experimental Environment

The synthetic trajectory data is generated from a network-based moving object generator [2]. The trajectories are generated based on a road network covering a $20km^2$ densely populated area of Melbourne, which is imported from OpenStreetMap [4].

The parameters are shown in Table 1. The *number of objects* is the number of vehicles generated in the whole area. By default, a vehicle takes one trip by following a shortest path between two points on the road network. We let some of the vehicles do multiple trips to simulate the movements of certain vehicles, e.g., taxi. For a multi-trip vehicle, the end point of a trip is the start point of the next trip. The *percentage of multi-trip objects* controls the number of vehicles that take multiple trips. We also control the number of trips taken by the multi-trip objects through the parameter, *number of trips*. The *Query Rectangle Size (QRS)* is the ratio between the size of the query rectangle to the size of the whole area. Finally, the *number of cells* controls the maximum granularity of the histograms. This value is the number of cells in DEHs and the lowest level in EHT.

4.1.2 Results

Number of objects: When the number of objects changes between 100 and 2000, the average accuracy of EHT varies between 94.5% and 98.9% (Figure 10). The accuracy of EHT is always higher than DEHs, whose accuracy is between 83.5% and 86.2%. The deviation of DEHs' accuracy is higher than that of EHT. For example, DEHs' deviation is 16.2% for 2000 objects while EHT's deviation is 3%.

Percentage of multi-trip objects: When there are more multi-trip objects, the effects of the distinct counting problem are more obvi-

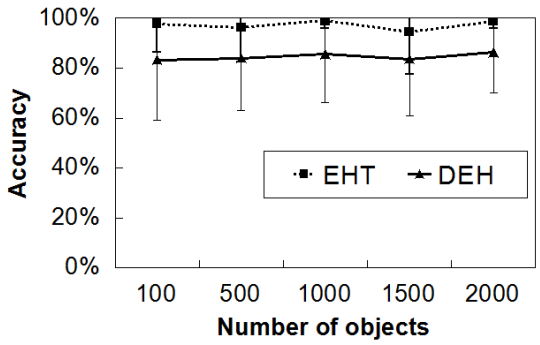

Figure 10: Accuracy of EHT and DEH over the number of objects using synthetic trajectory data.

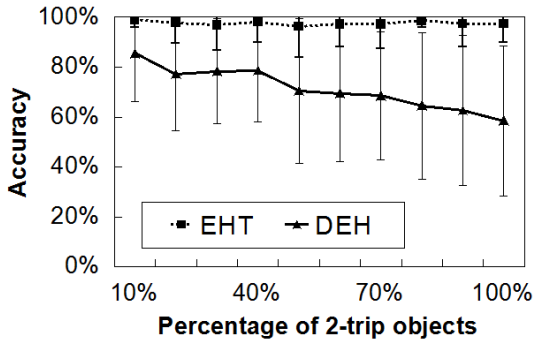

Figure 11: Accuracy of EHT and DEH over the percentage of 2-trip objects using synthetic trajectory data.

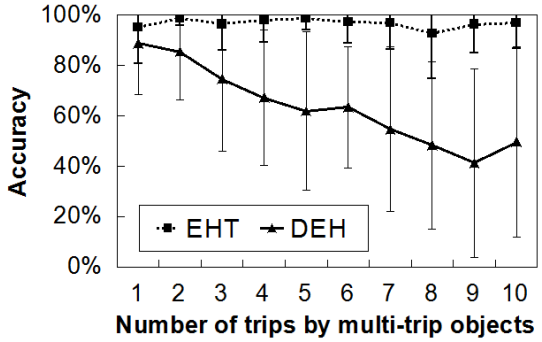

Figure 12: Accuracy of EHT and DEH over the number of trips taken by multi-trip objects using synthetic trajectory data.

ous on DEHs because there tends to be more trajectories that exist with separate sub-trajectories in the query rectangle. Figure 11 shows that the accuracy of DEHs drops from 85.5% to 58.5% with the percentage of 2-trip objects increases from 10% to 100%. The deviation of DEHs' accuracy is high – it reaches 30.2% when all objects take two trips. EHT maintains the accuracy level between 96.3% and 98.9%, and a deviation between 3% and 12.6%.

Number of trips: When we increase the number of trips taken by the multi-trip objects, DEHs' accuracy drops significantly – the lowest accuracy being 41.4% when each multi-trip object takes 10 trips (Figure 12). EHT maintains the accuracy level between 92.7% and 98.9%.

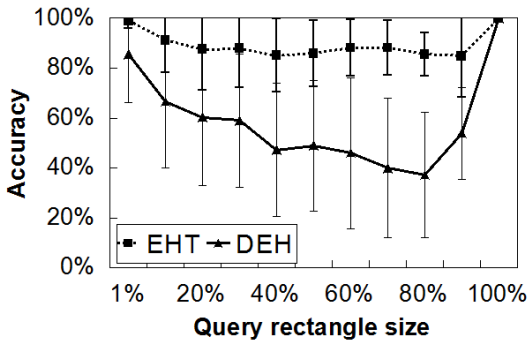

Figure 13: Accuracy of EHT and DEH over the query rectangle size using synthetic trajectory data.

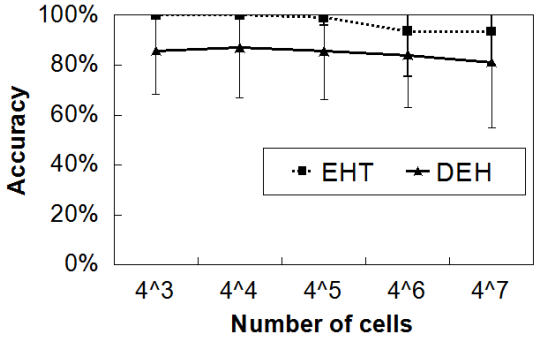

Figure 14: Accuracy of EHT and DEH over the number of cells using synthetic trajectory data.

Query rectangle size: The query rectangle size has interesting effects on the accuracy levels. As shown in Figure 13, DEHs' accuracy drops significantly as QRS increases from 1% of the whole area but jumps from 37.2% at 80% QRS to 100% at 100% QRS. The drop of accuracy is due to the fact that more objects are involved in the queries when QRS increases. However, when QRS keeps increasing, separate sub-trajectories of the same object join each other in larger query rectangles. EHT's accuracy is also slightly affected by QRS – it drops from 98.9% to 85% then returns to 100% at 100% QRS.

Number of cells: EHT's accuracy changes from 100% to 93.4% when the number of cells increases from 4^3 to 4^7 (Figure 14). The slight drop of accuracy is due to the fact that there is a higher chance that EHT needs to collect counts from low-level histograms near the boundary of the query rectangles, when the maximum granularity increases. Consequently, EHT tends to collect more redundant counts from the query rectangles. The accuracy of DEHs also drops slightly with the increase of the number of cells. It varies between 81% and 86%.

4.2 Real Trajectories

4.2.1 Experimental Environment

We use the real GPS trajectories from T-Drive [14, 15] sample dataset to test EHT and DEHs. The dataset contains GPS traces from trajectories of taxicabs in Beijing. The trajectories were collected over a period of 7 days with an average sampling interval of 3 minutes. Table 2 details these experimental settings. We tested the two algorithms against the *number of trajectories*, the *query*

No. of Trajectories	QRS	No. of Cells
2000 – 10000	1%	4^5
2000	1% – 100%	4^5
2000	1%	$4^3 - 4^7$

Table 2: Experimental settings for real trajectory data.

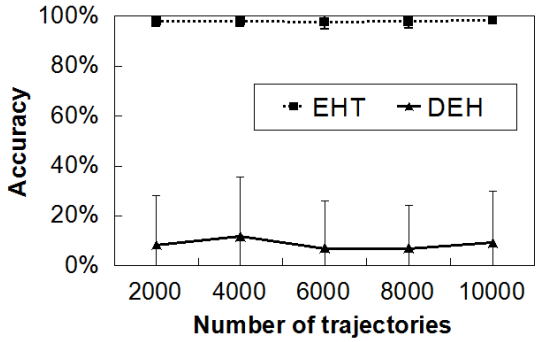

Figure 15: Accuracy of EHT and DEH over the number of trajectories using real trajectory data.

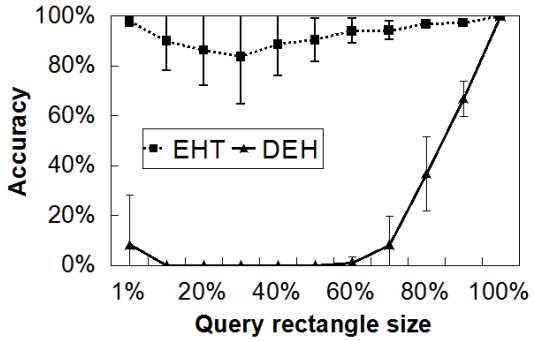

Figure 16: Accuracy of EHT and DEH over the query rectangle size using real trajectory data.

rectangle size and the *number of cells*.

4.2.2 Results

Firstly, we observe that, since it is highly likely that a taxicab has a large number of short trips to certain areas during the data collection period, the effects of the distinct counting problems on DEHs are significantly higher than using synthetic trajectory data.

Number of trajectories: When the number of trajectories changes between 2000 and 10000, the average accuracy of EHT varies between 97.7% and 98.4% (Figure 15). DEHs' average accuracy is between 7% and 11.7%. The deviation of EHT's accuracy is below 2.9% while the deviation of DEHs' accuracy is above 17.2%.

Query rectangle size: Similar to the results on synthetic trajectories, the accuracy of both approaches drops when the value of *QRS* is increased beyond 1% (Figure 16). The accuracy of both approaches increases after *QRS* reaches certain values. Eventually both approaches achieve 100% accuracy level when *QRS* is 100%. However, even in this context, the average acuracy of EHT is consistently higher than 83.7%. DEHs' accuracy is below 66.9% when *QRS* is below 90%.

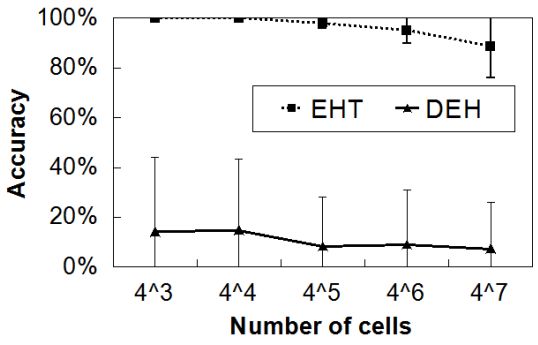

Figure 17: Accuracy of EHT and DEH over the number of cells using real trajectory data.

Number of cells: Figure 17 shows that EHT's accuracy drops from 100% to 88.7% when the number of cells is being changed from 4^3 to 4^7. However, it also shows that the accuracy of DEHs drops from 14.2% to 7.3% as the number of cells changes. This result once again demonstrates that EHT can maintain a higher accuracy level on real trajectory data while DEHs cannot achieve satisfactory accuracy level.

5. CONCLUSION

We introduced EHT – a histogram-based data structure for managing aggregate (count) values for moving objects trajectories, along with an efficient approach to retrieve the number of distinct trajectories inside a given query rectangle. EHT's hierarchical multi-granularity structure can address the distinct counting problem in various scenarios. Compared with a leading histogram-based approach, the proposed approach exhibits significant advantages in terms of accuracy in the experiments with with both synthetic trajectories and real trajectories dataset. EHT offers a good alternative to ID-based approaches, since it achieves a high accuracy level without the need for ID information.

As part of our future work, we will firstly attempt to extend the applicability of EHT to broader categories of range queries, such as non axes-parallel rectangles and arbitrary polygons (convex, concave, non-simple). As for a longer-term objective, we will focus on balancing the trade-offs between improving the precision of the aggregate distinct count values for the scenarios where EHT yields incorrect counts vs. the potential overheads incurred in terms of the additional information/processing used by the EHTs.

6. ACKNOWLEDGMENTS

This research was partially supported under Australian Research Council's Discovery Projects funding scheme (project number DP110100757 and DP130103705).

7. REFERENCES

[1] R. Beigel and E. Tanin. The geometry of browsing. *Third Latin American Symposium on Theoretical Informatics*, 1998.

[2] T. Brinkhoff. A framework for generating network-based moving objects. *GeoInformatica*, 6(2), 2002.

[3] S. Govindarajan, P. K. Agarwal, and L. Arge. CRB-tree: an efficient indexing scheme for range-aggregate queries. In *ICDT*, 2003.

[4] M. Haklay and P. Weber. Openstreetmap: User-generated street maps. *IEEE Pervasive Computing*, 7(4), 2008.

[5] J. Jin, N. An, and A. Sivasubramaniam. Analyzing range queries on spatial data. In *ICDE*, 2000.

[6] L. Leonardi, S. Orlando, A. Raffaetà, A. Roncato, C. Silvestri, G. Andrienko, and N. Andrienko. A general framework for trajectory data warehousing and visual OLAP. *GeoInformatica*, 18(2), 2013.

[7] D. Papadias, P. Kalnis, J. Zhang, and Y. Tao. Efficient OLAP operations in spatial data warehouses. In *SSTD*, 2001.

[8] H. Samet. *Foundations of multidimensional and metric data structures*. Morgan Kaufmann, 2006.

[9] C. Sun, D. Agrawal, and A. El Abbadi. Selectivity estimation for spatial joins with geometric selections. In *EDBT*, 2002.

[10] Y. Tao, G. Kollios, J. Considine, F. Li, and D. Papadias. Spatio-temporal aggregation using sketches. In *ICDE*, 2004.

[11] Y. Tao and D. Papadias. Range aggregate processing in spatial databases. *IEEE TKDE*, 16(12), 2004.

[12] H. Xie, E. Tanin, and L. Kulik. Distributed histograms for processing aggregate data from moving objects. In *IEEE MDM*, 2007.

[13] A. Yaagoub, X. Liu, G. Trajcevski, E. Tanin, and P. Scheuermann. Materialized views for count aggregates of spatial data. In *ADBIS*, 2012.

[14] J. Yuan, Y. Zheng, X. Xie, and G. Sun. Driving with knowledge from the physical world. In *ACM SIGKDD*, 2011.

[15] J. Yuan, Y. Zheng, C. Zhang, W. Xie, X. Xie, G. Sun, and Y. Huang. T-drive: driving directions based on taxi trajectories. In *SIGSPATIAL*, 2010.