Scalable Generation of Synthetic GPS Traces with Real-life Data Characteristics *

Konrad Bösche[1] Thibault Sellam[2] Holger Pirk[2]
René Beier[1] Peter Mieth[1] Stefan Manegold[2]

[1] TomTom, Berlin, Germany
{*first.last*}@tomtom.com

[2] Centrum Wiskunde & Informatica (CWI), Amsterdam, The Netherlands
{*first.last*}@cwi.nl

Abstract. Database benchmarking is most valuable if real-life data and workloads are available. However, real-life data (and workloads) are often not publicly available due to IPR constraints or privacy concerns. And even if available, they are often limited regarding scalability and variability of data characteristics. On the other hand, while easily scalable, synthetically generated data often fail to adequately reflect real-life data characteristics. While there are well established synthetic benchmarks and data generators for, e.g., business data (TPC-C, TPC-H), there is no such up-to-date data generator, let alone benchmark, for spatiotemporal and/or moving objects data.
In this work, we present a data generator for spatiotemporal data. More specifically, our data generator produces synthetic GPS traces, mimicking the GPS traces that GPS navigation devices generate. To this end, our generator is fed with real-life statistical profiles derived from the user base and uses real-world road network information. Spatial scalability is achieved by choosing statistics from different regions. The data volume can be scaled by tuning the number and length of the generated trajectories. We compare the generated data to real-life data to demonstrate how well the synthetically generated data reflects real-life data characteristics.

1 Introduction

Performance is one of the major selling points of Database Management Systems (DBMSs). However, objectively capturing DBMS performance is hard. The data management community has defined many benchmarks to capture DBMS performance in a single, or at least few, numbers. However, designing a representative benchmark, i.e., one that makes it easy for potential users to extrapolate a DBMS's performance for their application from the benchmark results, is hard. A representative benchmark must contain at least a) a representative query set and b) a representative database that the queries are performed on. A reasonably

* This publication was supported by the Dutch national program COMMIT

small, yet representative set of queries is hard to find and beyond the focus of this paper. A representative dataset for applications of a given domain, however, may be achievable. We focus entirely on the generation of such a representative dataset for the domain of (historical) moving objects data management.

Traditionally, there have been two ways to achieve such a representative dataset. The first is to start with a real-life dataset and stripping it down to the minimal database that is still representative for the application. The second is to synthetically generate a representative dataset from scratch. Both of these approaches have their merits and problems that we discuss in the following.

Real-Life Data is a good basis for a representative dataset because it has the desired characteristics. The distribution and correlation of the values, e.g., can have a large impact on the performance of applications on top of the data. These effects will be discovered when evaluating a system using a real-life dataset. However, a real-life dataset lacks the configurability of a synthetic dataset, most importantly, regarding its size. Since a real-life dataset is always a snapshot of the application data at a given time, it can not be used for what-if-analysis. This makes it hard to detect, e.g., future scalability problems. In addition, real-life data is often sensitive with respect to user privacy. Especially when tracking user positions, a dataset could be used to generate presence/absence profiles of people. It could even be matched to an address database to identify individuals. Both of these problems can be addressed by synthetically generating data.

Synthetic Data is naturally anonymized since it never contains data about real users. In addition to that, synthetic data can, usually, be generated at any *scale-factor* (i.e., dataset size). However, care must be taken to generate data that resembles the characteristics of real-life data of the targeted applications. This is usually achieved by encoding domain specific rules into the data generator. However, these rules are not only tedious to create. The manual creation of rules is also error prone and might not consider all relevant characteristics of the data.

To resolve the conflict between realistic and scalable representative datasets, we propose to use a hybrid approach that is illustrated in Figure 1. Datasets are based on a "sample" dataset. Statistics are extracted from this sample dataset and combined with a model of the domain to produce a scalable dataset that closely resembles real-life data. In our case we distinguish a user model and a physical model. The user model represents the intention of a user (e.g., the route a user takes between two points). The physical model captures constraints that are given by the real world (speed limits, traffic lights, ...).

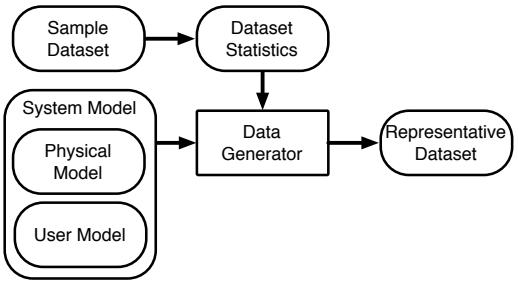


Fig. 1: Overview Hybrid Dataset Generation

To present our approach, we structured the rest of this paper as follows: In Section 2 we present the requirements for realistic Global Positioning System (GPS) data and other approaches to generate data that fulfills these requirements. In Section 3 we describe our approach to generate GPS data from sample data and a physical model. We evaluate the quality of the generated data as well as the generator performance in Section 4 and conclude in Section 5.

2 Background

2.1 Use cases and requirements

Traffic data is the basis for many applications. To illustrate the benefit of our data generator, we want to briefly discuss the range of applications that can be evaluated using the generated data. We target at least two types of applications that can be classified into the well known domains of Online Transaction Processing (OLTP) and Online Analytical Processing (OLAP): Location Based Services and Traffic Monitoring and Analysis.

Location Based Services aim at providing end-users with answers to geospatial queries that involve their current position. Typical queries are: "*What are the three closest restaurants to me?*", "*What is the best route to avoid this traffic jam?*" or "*Where can I meet up with my husband within half an hour?*". Low query latency is critical to achieve a good end user experience. However, few queries require historical data which limits the amount of considered data.

Traffic Monitoring and Analysis applications have very different characteristics. The goal is to provide an insight into the traffic at a macro scale, often focusing on trends that develop over time. Queries such as "*Which were the busiest routes in Europe this year?*" or "*What is the impact of a new road on regional traffic?*" naturally involve data acquired over a period of time. On the one hand, the large data volume of traffic monitoring applications poses a challenge. On the other hand, low query latency is less critical.

While targeting all of these cases, we limit our simulation to road-bound vehicles that are tracked using GPS. There are no restrictions on the streets (city or highway), time, or region. However, the generator should produce data that resembles real GPS data, including factors like precision and noise.

In practice, GPS devices produce *fixes* (i.e., samples) that are defined by four attributes: `trace_id`, `longitude`, `latitude` and `time`. The first field is a code that identifies the GPS device for a certain period of time, e.g., one day. The next two fields are the GPS coordinates in degrees at a precision of 5 decimals. The time is the unix timestamp, i.e., time at a resolution of one second.

2.2 Spatio-temporal data generation

Moving objects For the last decade, the interest for Moving Objects databases has led to the creation of several dedicated generators.

The *GSTD* algorithm (Generate_Spatio_Temporal_Data) [11] is one of the first contributions. It generates a set of points or rectangular regions according

to a predefined distribution (e.g., Gaussian). These objects are then translated and resized by random functions with user-defined parameters. This algorithm is extended in [7] to create more realistic data. New parameters affect the direction shifts, and some objects move in clusters. More importantly, an infrastructure is introduced. The system generates a set of rectangles in which the objects cannot enter. This is the first attempt to impose *constraints* on the movements. The Oporto generator [10] is based on an other approach. It simulates swarms of fish and ships using *attraction* and *repulsion* between the moving objects.

These projects offer different levels of control over the trajectories, and several refinements were introduced to avoid a fully chaotic behavior. Nevertheless, none of them can simulate environment constraints such as road networks.

Traffic simulators Many traffic simulators have been proposed, with various objectives. The transportation engineering community makes heavy use of micro-scale simulators, that aim at generating short term traffic conditions with physical models. For instance, DRACULA [8] creates urban mobility patterns. The work presented in [9] generates highway traffic with a parallel architecture. Our objective is different. First, we aim at creating large amounts of data in a short time. These micro-simulators target high precision rather than volume. Second, we require realistic data about collective behaviors over large periods of time (e.g., the Netherlands during a month). These solutions target short term vehicle movements at a local scale. The intentions of the travelers (where they come from, where they are going, when the next trip will be) may not be accurate [3].

The data management community proposed several large scale generators. The closest project to ours is presented by Brinkhoff [1]. It is based on network information and routing functions defined by the user. Each edge of the network has a user-specified maximum capacity and speed. Moving objects are created at each timestamp, travel, then "disappear" when they reached their destination. The speed of object creation and their routing is defined by user functions. Our approach differs on two points. First, the data generation does not depend on arbitrary user defined functions or parameters: we use historical data. Second, we maintain consistency between the trips of a same vehicle. The benchmark Berlin-MOD [3] contains a realistic data generation algorithm. Nevertheless, it relies on several rules fine-tuned for the benchmark use-case (the traffic of Berlin).

3 Generating Trajectory Data

The simulated GPS data is generated in three steps. First we gather statistics about real world historical GPS data collected from in-car navigation devices and use these to randomly generate Origin-Destination (OD) pairs with similar characteristics. These OD pairs describe the geographical start and end of a sequence of GPS points (i.e., a *trip*). In the next step, we calculate a trajectory between the two points of each OD pair using a route finding algorithm. We use a digital map of the road network in the considered area to compute the fastest

route. In the last step, we apply time dependent speed limitations for each edge of the map. In the rest of this section we describe each of these steps in detail.

3.1 Generating Origin-Destination Pairs

Gathering Statistics The collection of statistics is the first step of the synthetic trace generation. We used the TomTom GPS archive that contains more than 4 billion hours of GPS data from in-car navigation devices. First, the real-life traces are divided into trips. This is done by simply checking the temporal gaps between each pair of succeeding GPS fixes against a threshold of 15 minutes. If a gap is larger than this threshold the trace is split between the respective fixes. In addition, we use meta-data, namely device events of type "suspend", which are assigned to a certain GPS fix within the belonging trace. Of each trip, we use the first and the last point (origin and destination) for subsequent processing. We build a set of histograms on these points:

- A two dimensional equi-width histogram on the origin coordinates.
- A two dimensional equi-width histogram on the destination coordinates.
  Note: *Naturally, a finer resolution of the histogram yields better results but comes at a performance penalty. During our experiments, we found that a histogram covering the target area at a resolution of 400 by 500 cells (a.k.a. bins or buckets) yields good results at acceptable performance.*
- An equi-width ($1\,km$) histogram on the euclidean distance between origin and destination.
- A set of histograms of the discrete values for the time and date components (year, month, day of week, minute of day) of the origins of the traces.
- A histogram of the discrete values for the GPS sampling rate.
- A histogram of the discrete values for the number of trips per device per day (trips per trace).
- A histogram of the discrete values for the pause between two trips of a device in minutes.

The Digital Road Network In addition to the histograms of the GPS data samples, we use a *Digital Road Network* to simulate vehicles on real roads. In its essence, the network is a directed graph with labels on nodes and edges. Edges represent road segments and are, thus, attributed with labels that describe, e.g., speed limits, road classes or average speeds. Nodes in the digital road network are largely defined by a geographical position. However, the exact definition of a node is hard. Intersections of roads are, naturally, represented as nodes in the digital network. However, a node could also be a change of direction of the road: If two nodes are connected, we assume the connection to represent a straight road. A bent road is, therefore, represented by multiple nodes and edges. Whilst TomTom's digital road network is not disclosed, projects like OpenStreetMap [4] provide publicly available digital road networks that follow the same idea.

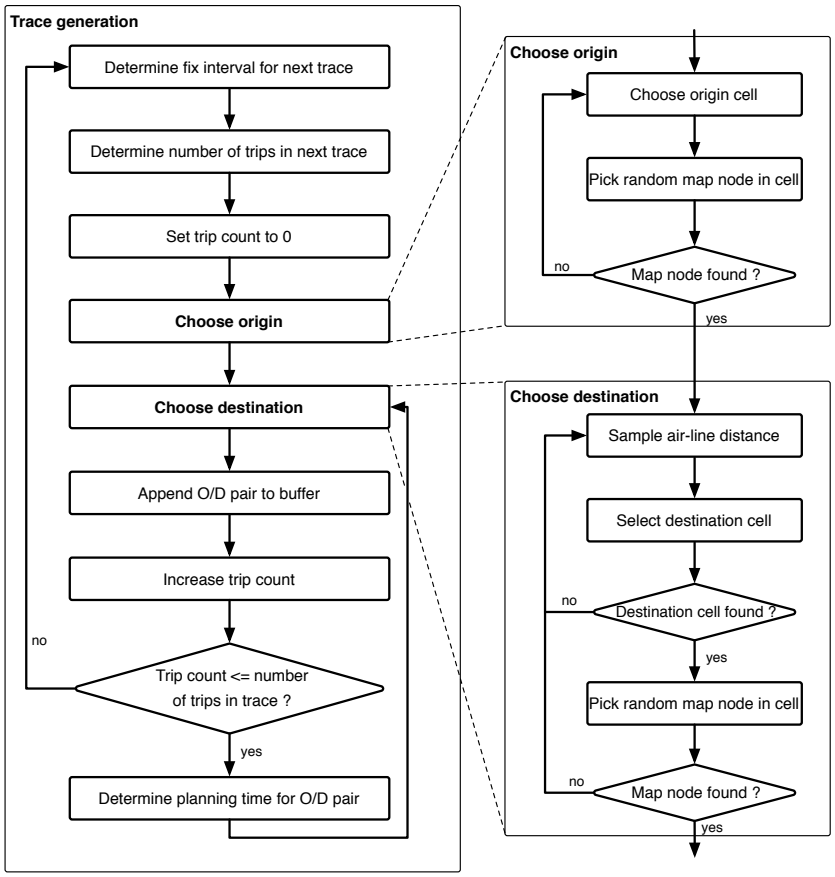


Fig. 2: Trace generation algorithm

Generating OD Pairs As basis for the trip generation, we generate OD pairs (see Figure 2 for an overview). We will describe this process in the following.

The first step of the trace generation is setting the trace related parameters[1]. These values are sampled according to the respective histograms and form the basis for the trip simulation. Since, in reality, trips that are acquired using the same GPS device are often correlated, we have to take care to reproduce such correlations in the generated data. To illustrate this, consider the following example: a commuter generates a trip in the morning on the way to work and one trip in the evening on the way home. We call such a set of correlated trips a *trace*. Thus, we generate trips such that all trips of a given device on a given day form a trace[2]. The origin of the first trip of a trace is selected as follows: a

[1] GPS sampling interval, number of trips, date and time of first trip

[2] In practice, our synthetically generated traces can span more than one day, which explains the bias towards an earlier time of day in Figure 5b.

random cell (aka. bucket) is selected according to its relative frequency in the origin histogram. From all the nodes of the digital road network that fall into the selected cell we randomly (uniform) select one to be the start point of the trip. If the cell contains no nodes, we resample a cell. The time of the first (origin) sample of the trip is chosen according to the respective distribution.

The destination of a trip is generated as follows: we randomly select an approximate value for the air-line distance between origin and destination according to the respective distribution. As the origin is already fixed we generate a set of candidate destination cells. This set contains all cells that intersect a circle around the origin with a radius of the set distance. From these, we randomly select one according to the relative frequency in the destination histogram. Within the selected destination cell we randomly (uniform) select one map node as destination. If we fail in one of the described steps, i.e., if there is no map node in the cell selected or the frequencies of the destination cell candidates are all zero, resample a new air-line distance and restart the process.

To determine the starting time of subsequent trips, we add a random idle time to the end time of the previous trip. The idle time is set according to the respective histogram.

3.2 Routing

Given the origin and destination of a trip our algorithm will generate GPS points along the fastest path on the digital road network. We utilize TomTom's routing kernel that is also used in their navigation devices. It uses several heuristics for accelerating the search including $A^\star$ [5] and arc-flags [6]. Accelerating shortest path computations in road network has received some attention from the scientific community in recent years [2].

3.3 Physical Modeling

Simulating a journey As the last step we simulate a journey on the calculated route. The idea is to virtually drive along the route and sample the position at uniform time intervals.

To generate GPS data that closely matches the individual speed characteristics of vehicles on each street of the network, we use the speed profiles from the digital road network combined with a physical model based on a set of parameters that we manually selected (see Table 1). As a basis for the speed, we use the average speed on a road segment at the given time of day and day of week. To add a realistic variance to the speed we multiply the traveling speed on each road segment with a stretch factor ssf. With a probability p_{sc}, ssf is set to a

Symbol	Value
ssf_l	.8
ssf_h	1.2
p_{sc}	.1
p_{stop}	.02
$p_{stop@end}$	.9
ts_l	$5s$
ts_h	$40s$
σ_{shift}	3 meters
p_{shift}	.05
p_{drift}	.03
σ_{drift}	10 meters

Table 1: Model Parameters

random value between ssf_l and ssf_h for a road segment. With probability $1-p_{sc}$ it is set to the same value as the previous road segment in the trip.

In addition to the variance of the speed, vehicles occasionally have to stop, e.g., for traffic lights. Hence we occasionally stop the virtual journey at likely spots. We simulate a stop on a road segment with a probability p_s. The position of the stop on the road segment is determined randomly but biased towards the end of the segment. With a probability $p_{stop@end}$, the stop will occur in the last 20% of the segment (and with $1 - p_{stop@end}$ in the first 80%). The duration of the stop is set randomly between ts_l and ts_h seconds.

Simulating GPS noise The last step of the simulation covers GPS signal noise. To achieve a realistic noise, a semi-random perturbation is applied to each of the sampled GPS points. Two components make up the simulated noise:

- A random shift for each individual GPS point.
- A random drift over a sub-sequence of GPS points.

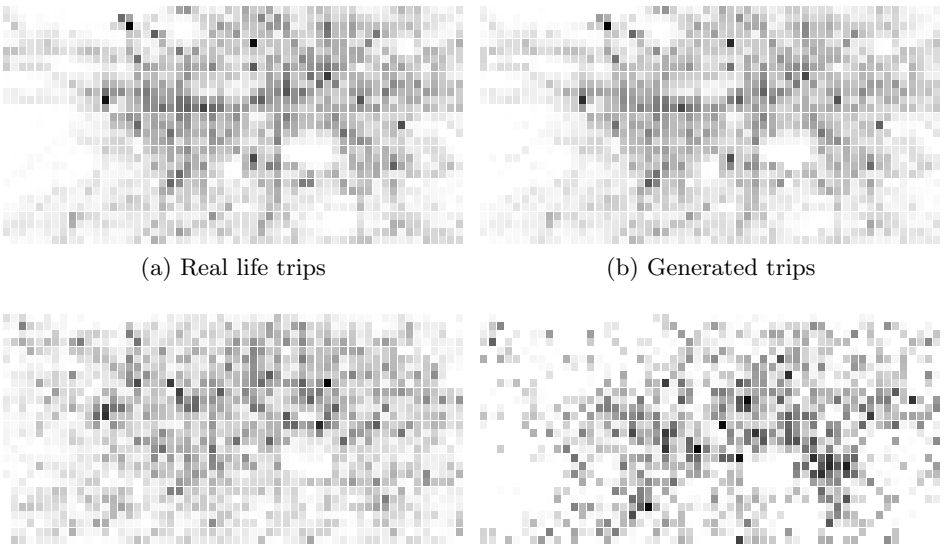
The former is simulated by adding a Gaussian noise to each of the two dimensions (longitude and latitude) of each GPS point. We use a distribution with mean value of 0 and with a standard deviation of σ_{shift}. However, this may lead to successive GPS points having different deviation directions. They would appear to "jump" from one side of a road to the other which is untypical for real GPS samples. To limit this effect we apply such shifts only with a probability of p_{shift} *per second*[3]. With probability $1 - p_{shift}$ *per second*[3], we add the deviation of the previous point to the calculated deviation of the current point.

The second noise component, the random drift, is initiated with a probability of p_{drift} *per second*[3]. "Drift" means a shift along the orthogonal of the current driving direction. More precisely, a shift along the orthogonal of the line defined by the predecessor and the successor of the considered point. Whenever a drift is initiated, a new maximal drift distance is determined according to a Gaussian distribution with mean 0 and standard deviation of σ_{drift} meters. The determined maximal drift distance will then be reached from the current drift distance (0 if there is no incomplete preceding drift) within exactly 30 seconds (in case no new drift is initiated along the way). For example, this corresponds to 3 GPS points in case of GPS point interval of 10 seconds. After the maximal drift distance has been reached, the drift distance decreases to zero within 30 seconds again.

4 Evaluation

In order to evaluate how well our data generation approach mimics real-life data, we compare the synthetically generated data with real-life data. We do so from

[3] The probability is normalized by time between two GPS fixes in order to avoid simulating less/biased noise and drift for traces with a higher GPS sampling rate.



(a) Real life trips (b) Generated trips

(c) Trips generated with Brinkhoff (d) Trips generated with BerlinMOD

Fig. 3: Spatial distribution of departures

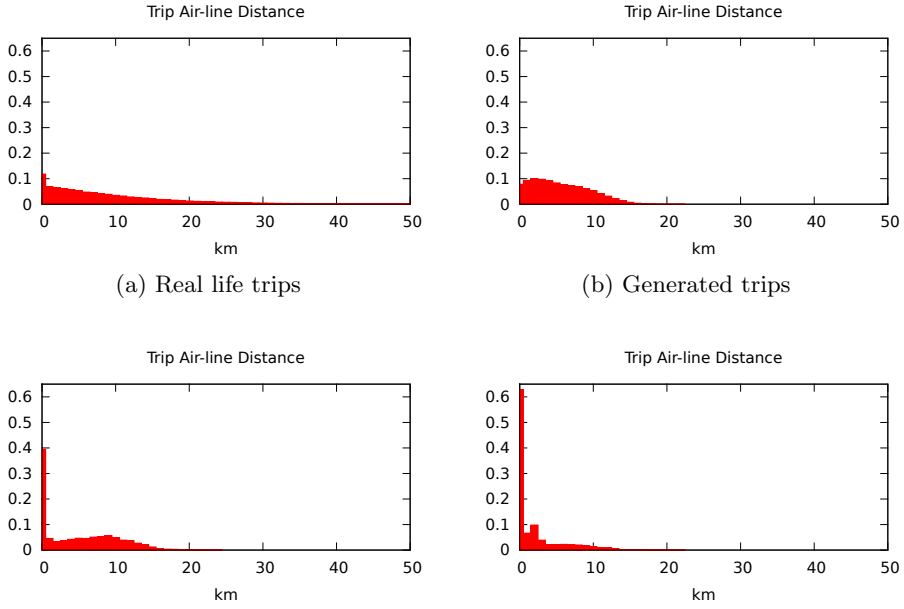
two different angles. First, we compare various statistical properties of both synthetically generated and real-life data. This is mainly a sanity check that the statistics extracted from real-life data are correctly used during the data generation process and thus correctly reflected in the generated data. Second, we compare the spatial distribution of real-life and synthetically generated data. Mostly visual inspection of density plots suggests that our synthetically generated data "looks very similar to" real-life data for various geographical regions at different resolutions. Finally, we assess the performance of our generator to ensure that we can generate large volumes of data in adequate time.

4.1 Statistical properties of the trips

To assess the quality of the generated data, we compare four sets of trips:

1. An archive of user traces provided by TomTom from February 2011, region of Berlin. There are 217,165 traces. We cropped the dataset to a smaller region to make comparisons possible [4].
2. A set of 135,000 traces containing 289,716 trips generated with our system. The input statistics are extracted from the first set (Berlin area, February 2011) and the generator was setup to cover the baseline region.

[4] 52.42 °N - 52.56 °N, 13.22 °E - 13.50 °E



(a) Real life trips

(b) Generated trips

(c) Trips generated with Brinkhoff

(d) Trips generated with BerlinMOD

Fig. 4: Distribution of air-line distances

3. 28 days of data provided with the BerlinMOD generator. The set is publicly available [5]. We removed all the traces that did not belong to our region. The dataset is made of 111,114 trips, generated by 1589 vehicles.
4. A set of 302,400 trips generated by the Brinkhoff generator over the same region. The transportation network was generated from the roads layer of OpenStreetMaps [6]. We used the class DefaultDataGenerator. The parameters were set in a best effort way.

Geographical validation Figure 3 describes the distributions of departures in the Berlin area for each set of trips. Our data is very close to the TomTom archive. The two other datasets exhibit fairly similar distributions.

We represent the distributions of trip air-line distances in Figure 4. The data created by our generator seems fairly realistic. One major difference is the distribution of very small values (less than 1 km). There is a small but distinct peak in the real data, which is not present in our dataset. This can be explained by the grid that we apply on the network for data generation. A side effect of this method is that very small values are often over-approximated.

[5] http://dna.fernuni-hagen.de/secondo/BerlinMOD/BerlinMOD.html
[6] http://www.openstreetmap.org/

The authors of BerlinMOD assume that two kind of trips may be defined. *Work trips* are very short and frequent. They are described by the first peak. Oppositely, *additional trips* are longer, less common and there is more variance in the distribution of their lengths. The same type of behavior appears in the Brinkhoff dataset. This might add realism in more urban regions.

Scheduling of the trips An abstract timestamp represents time in the default Brinkhoff generator. There is no notion of hour of the day or calendar, the system has a uniform behavior at each unit of time. Also, it does not group trips in traces. Therefore, we do not consider the Brinkhoff generator in this section.

Figure 5 illustrates the time and day of the trip departures. The shape of the real-life distributions matches an urban traffic scenario: we can identify the early morning rush hours, the weekdays are busier than weekends.

Regarding the distribution of traces among the days of the week, our data is close the the real distribution. Similarly, there are less traces during the weekend in the dataset generated with BerlinMOD. However, the distribution of traces between Saturday and Sunday is different.

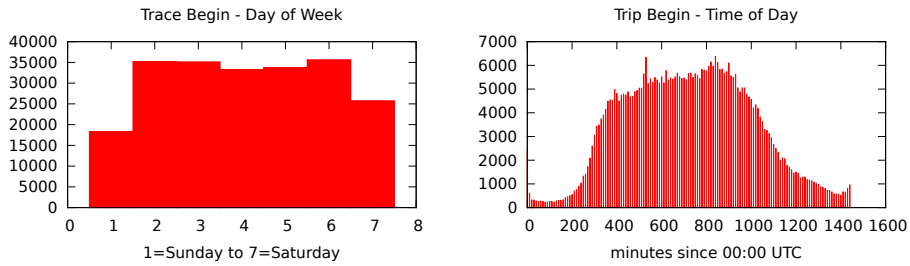
The overall shape of the distribution of trips during the day in our dataset is similar to the original. However, it shows a slight bias to the earlier hours as explained in Footnote 2 in Section 3.1. The distributions of the BerlinMOD trips is a direct consequence of how the data is generated. The authors specify several specific times in the day, then distribute departure hours around those with Gaussian distributions. This is represented by the three peaks in the graph.

Figure 6 describes the distribution of trip durations. They are directly related to the distributions of air-line distances (Figure 4). The trips are slightly shorter in our dataset. The fact that simulated vehicles always take the shortest way explains the difference. The BerlinMOD dataset also contains shorter trips. This is a consequence of the shorter air-line distances.

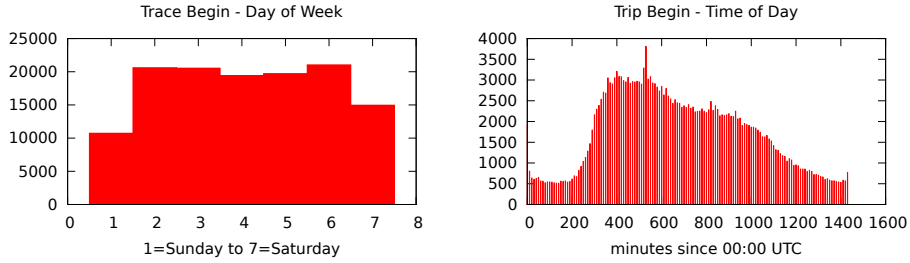
4.2 Spatial comparison

In order to support a spatial comparison of the generated traces with the real world traces we generate density plots at different resolutions. These are pictures where the color of each pixel encodes the number of GPS-points falling into the area represented by the pixel. In case each pixel represents a small area (less than 3×3 square meters) we give each GPS point a circular shape with radius 3 meter. Remember that the path taken between the origin and the destination of a trip depends exclusively on the digital road map and the weight function of its edges (speeds). Hence, the correctness and precision of the map used has a vital influence on the produced GPS points. As the density of regions differs largely we use a logarithmic scale for the colors. The density plots have been generated on a synthetic trace archive covering whole Europe with 1 million traces.

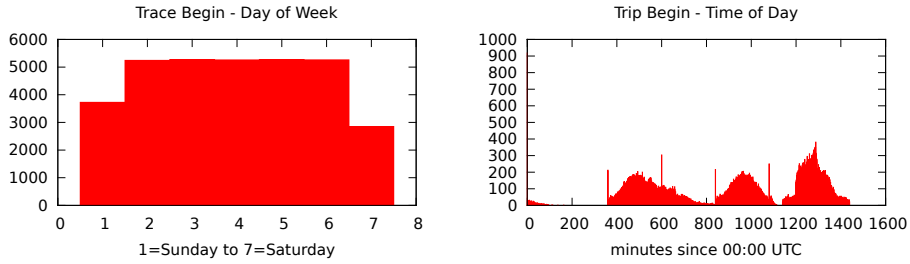
The first example in Figure 7 shows two pictures with density of GPS points for whole Germany. The left picture is generated from the real world data archive whereas the right picture is made from the synthetic trace archive. The different



(a) Real life trips



(b) Synthetic trips

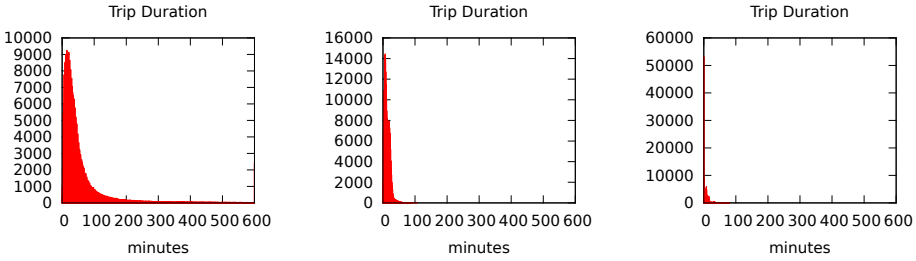


(c) Generated with BerlinMOD

Fig. 5: Trip departure schedules

densities of the real world data are well reflected in the picture of the synthetic traces. In fact, we think it is hard to tell them apart.

The same is true also for the pictures in Figure 8 which show the densities for the region of Amsterdam. In order to use approximately the same number of traces we used real world GPS-data for the left picture that has been collected during one day only. The small differences in the pictures mainly come from standing or parking cars that generate a high density as they do not move (uniform sampling in time).



(a) Real life trips (b) Synthetic trips (c) BerlinMOD trips

Fig. 6: Trip durations

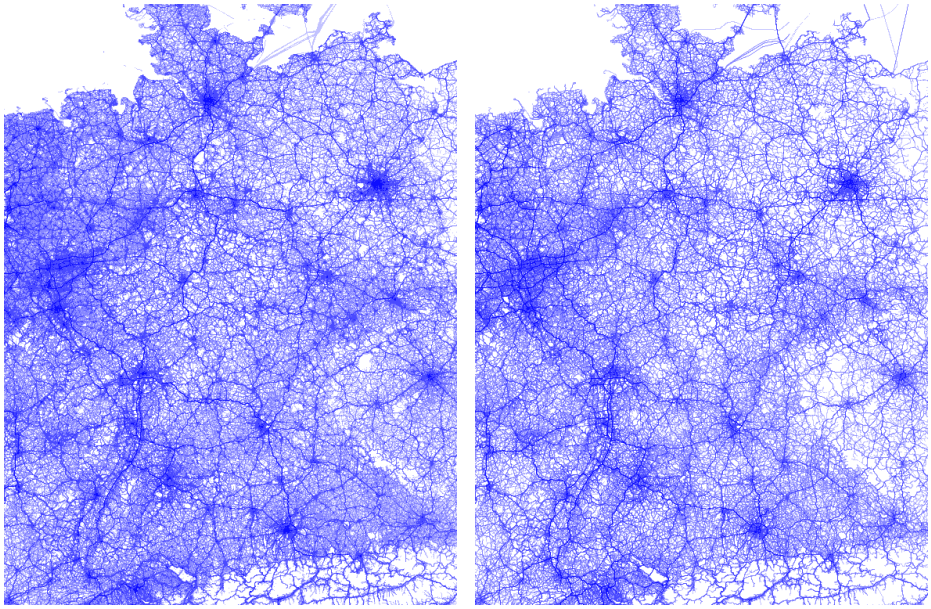


Fig. 7: Density of GPS points for Germany. Left: real world data. Right: synthetic data. Color coding with logarithmic scale.

A more detailed view for an urban area in Berlin is shown in the pictures of Figure 9. The high densities on the left picture (real world data) are due to a large percentage of vehicles that have to wait in front of the traffic light. The street with largest density on the right side (synthetic traces) has no density in the left picture as the road is closed in reality.

Figure 10 shows a big motorway junction south of Berlin. We observe that the real world GPS points exhibit a much smaller deviation from the center of the

Fig. 8: Density of GPS points for Amsterdam. Left: real world data. Right: synthetic data. Color coding with logarithmic scale.

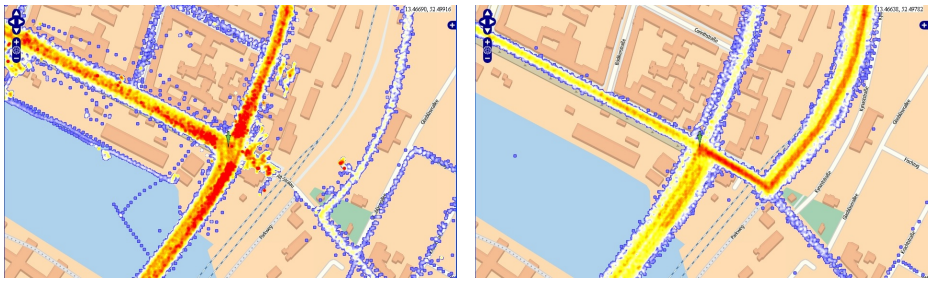


Fig. 9: Density of GPS points. Left (real world data): Standing cars in front of traffic lights produce higher density. Right (synthetic data): Many routes use street which is closed in reality.

street compared to the synthetic traces. This is due to the local conditions (no building or trees, wide open sky) that allow much better GPS reception leading to lower noise levels. An opposite situation is shown in Figure 11 covering an urban canyon in Frankfurt/Main with large buildings impairing the GPS quality.

4.3 Performance evaluation

Finally, we measure how much data our generator can generate per time.

Setup The experiments were run on a machine equipped with a 3.4 GHz quad-core Intel(R) Core(TM) i7-2600 CPU (8 hardware threads, 8 MB L3 cache), 8 GB of main memory, and a 7200 RPM 1 TB Seagate Barracuda ES.2 SATA hard disk connected via USB 2.0. To evaluate the performance of our current implementation, we generated 100,000 traces with simulated GPS noise within the bounding box of Germany[7]. For this purpose, we used the following input:

[7] 47.26784 °N - 55.13216 °N, 5.7344 °E - 15.07328 °E

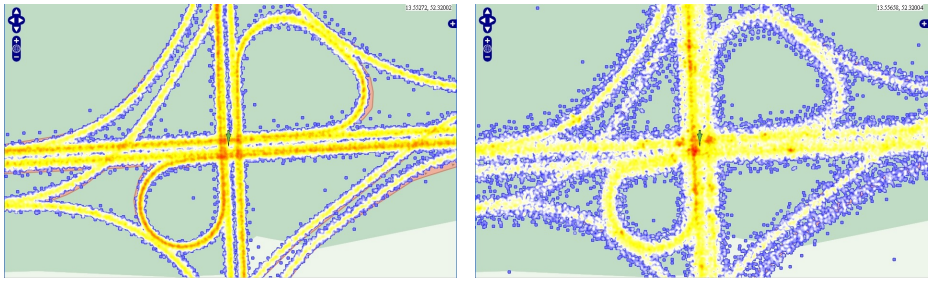


Fig. 10: Density of GPS points for motorway junction south of Berlin. Left: real world data, Right: synthetic data. Due to good GPS reception is the noise level of real world data smaller than for the synthetic traces.

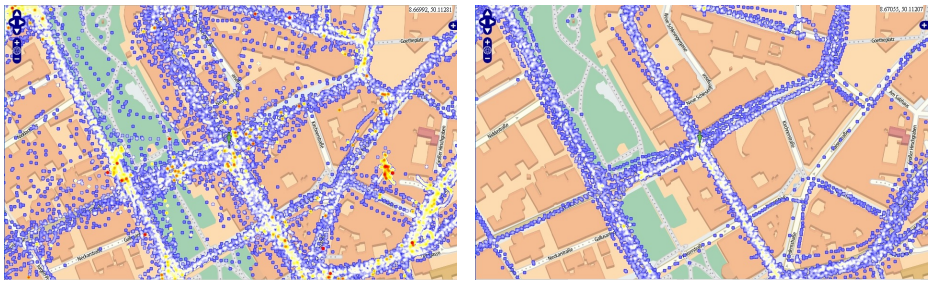


Fig. 11: Density of GPS points in urban canyon (Frankfurt/Main). Left: real world data. Right: synthetic data. Due to bad GPS reception, the real world data exhibits more noise than the synthetic traces.

- Trace/trip statistics (O/D grid cell size: 0.02 °x0.02 °) gathered from all traces within the bounding box of Germany contained in TomTom's real-life GPS data archive for 2010.
- A digital map of the entire road network of Europe.

Results The generated data (100,000 traces) consists of 216,875 trips, with 404 GPS fixes per trip on average, i.e., a total of 87.6 million GPS fixes, resulting in a ∼4 GB CSV file. With an overall execution time of just under 105 minutes (6279.644 seconds), our generator created on average 15.92 traces per second, respectively 34.54 trips per second (13,954 GPS fixes per second). In other words, it took on average 62.80 ms per trace, respectively 28.96 ms per trip (72 μs per GPS fix). The total execution time breaks down as follows. Generating the 216,875 OD pairs (one pair per trip) took 6 minutes and 49 seconds (6.5 %). Generating the location queries took 7.5 minutes (7.2 %). Planning the routes between all 216,875 OD pairs took 57.5 minutes, i.e., about 55 % of the total execution time. Generating the 100,000 traces took 6 minutes and 9 seconds (5.8 %). Writing the results to disk took 26.5 minutes (∼25 %).

While we believe that these are rather reasonable performance results, we point out that our initial implementation of our generator is purely sequential, i.e., it uses only a single CPU core.

Given that the whole process is for more than 75 % of the time "IO-free" and thus CPU-bound, parallelization is straight-forward, e.g., using spatial partitioning. The given region can be split into sub-regions (e.g., one per available CPU core / hardware thread), and independent generator processes can be run concurrently, one per sub-region.

5 Conclusion

We set out to resolve the conflict of real-life and synthetic representative data in the domain of traffic monitoring. To resolve this conflict, we introduced a hybrid data generation technique: Statistics that are gathered from a real-life application set are combined with a system model to generate a scalable dataset that preserves real-life data characteristics. We evaluated the generated dataset against our real life input data using visual inspection as well as statistic analysis. We found that the generated data closely resembles the real-life data and is, thus, a good basis for the evaluation of data management solutions.

References

1. T. Brinkhoff. A framework for generating network-based moving objects. *GeoInformatica*, 6(2):153–180, 2002.
2. D. Delling, P. Sanders, D. Schultes, and D. Wagner. Engineering route planning algorithms. In *Algorithmics of Large and Complex Networks*, pages 117–139, 2009.
3. C. Düntgen, T. Behr, and R. Güting. Berlinmod: a benchmark for moving object databases. *The VLDB Journal*, 18:1335–1368, 2009. 10.1007/s00778-009-0142-5.
4. M. Haklay and P. Weber. Openstreetmap: User-generated street maps. *Pervasive Computing, IEEE*, 7(4):12–18, 2008.
5. P. E. Hart, N. J. Nilsson, B. Raphael, and B. Raphael. A formal basis for the heuristic determination of minimum cost paths. pages 100–107, 1968.
6. M. Hilger, E. Köhler, R. Möhring, and H. Schilling. Fast point-to-point shortest path computations with arc-flags. *The Shortest Path Problem: Ninth DIMACS Implementation Challenge*, 74:41–72, 2009.
7. D. Pfoser and Y. Theodoridis. Generating semantics-based trajectories of moving objects. *Computers, Environment and Urban Systems*, 27(3):243–263, 2003.
8. D. V. V. R. Liu and D.P.Watling. Dracula: Dynamic route assignment combining user learning and microsimulation. *PTRC*, E, 1994.
9. M. Rickert, P. Wagner, and C. Gawron. Real-time simulation of the german autobahn network, 1997.
10. J.-M. Saglio and J. Moreira. Oporto: A realistic scenario generator for moving objects. In *DEXA Workshop*, pages 426–432, 1999.
11. Y. Theodoridis, J. R. O. Silva, and M. A. Nascimento. On the generation of spatiotemporal datasets. In R. H. Güting, D. Papadias, and F. H. Lochovsky, editors, *SSD*, volume 1651 of *Lecture Notes in Computer Science*, pages 147–164. Springer, 1999.