Generation of Synthetic Urban Vehicle Trajectories

Chrysovalantis Anastasiou
Department of Computer Science
University of Southern California
Los Angeles, California
canastas@usc.edu

Seon Ho Kim
Integrated Media Systems Center
University of Southern California
Los Angeles, California
seonkim@usc.edu

Cyrus Shahabi
Department of Computer Science
University of Southern California
Los Angeles, California
shahabi@usc.edu

2022 IEEE International Conference on Big Data (Big Data) | 978-1-6654-8045-1/22/$31.00 ©2022 IEEE | DOI: 10.1109/BigData55660.2022.10020237

***Abstract*—The analysis of trajectory datasets has numerous applications ranging from urban planning to human mobility understanding, but to protect the privacy of individuals trajectory datasets are rarely released to researchers. And even when they are, they are limited in size and spatio-temporal coverage. To address these issues a number of methods for generating synthetic yet realistic trajectory datasets have been proposed. These existing methods either require a lot of complex parameters to be calibrated (simulators) or rely on existing trajectory datasets (generative models). In this paper, we propose *Data-Driven Trajectory Generator*, dubbed *DDTG*, a data-driven, model-free, and parameter-less algorithm for generating realistic synthetic vehicle trajectory datasets. Unlike existing approaches, *DDTG* relies on aggregate origin-destination and traffic data, both of which are publicly available and free of privacy concerns. Furthermore, we show that our method is orthogonal to the existing approaches with which *DDTG* can be combined to generate synthetic datasets of higher quality. Our experiments with real-world trajectory and traffic data show that the datasets generated by *DDTG* follow distributions that are very close to the distributions of real trajectory datasets.**

***Index Terms*—synthetic vehicle trajectory, synthetic datasets**

I. Introduction

Big Trajectory Data provides limitless opportunities for understanding human mobility, traffic congestion patterns [1], improving routing and planning [2], and more. At the same time, it also introduces challenges in protecting the privacy of individuals [3] [4] making them hard and costly to obtain. On one hand, even with their increasing availability, most publicly available trajectory datasets can only cover a portion of the population and are limited to crowded regions and active hours of the day, often leading to biased observations. On the other hand, proprietary trajectory datasets are more comprehensive but their data holders such as Google, Apple, and other private entities (e.g., private truck drivers) are not willing to share.

In order to address these challenges, synthetic vehicle trajectory generation approaches have been proposed. The task of synthetic trajectory generation is useful when either the input set is small and we need more trajectories for the downstream task (*scale-up*) or when the privacy of the input set must be preserved (*privacy preservation*). One can also use this methodology to generate trajectories for one geographical area (e.g., city, neighborhood) from the trajectories belonging to a different geographical area (*diversification*).

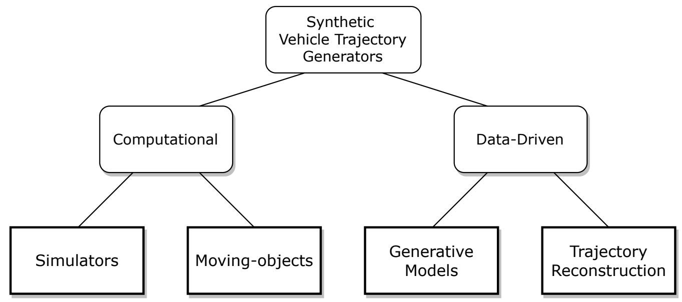

Fig. 1. Taxonomy of synthetic vehicle trajectory generators.

We divide the synthetic trajectory generators into two main categories: (i) *Computational*, and (ii) *Data-Driven*. The taxonomy of these generators is shown in Figure 1.

Computational generators typically process information about the environment to generate vehicle trajectories. We further classify these into *Simulators* and *Moving-object Generators*. Approaches in the latter class generate network-based moving objects using demand-supply models [5]. They used to be state-of-the-art back when they were originally proposed but are now outdated. Approaches that fall under the former class simulate the actions of agents (e.g., vehicles) in a closed environment [6]. Every action updates the state (i.e., the location) of the agent and their history represents a synthetic trajectory. In order to generate realistic vehicle trajectories, computational generators require a fully calibrated environment. However, acquiring and fine-tuning all their parameters is a challenging task [7].

Data-driven generators leverage real-world experiences (data) to synthesize trajectories. The current state-of-the-art in this category is the Generative Adversarial Networks (GAN). These deep learning models train on existing vehicle trajectory data and learn a generator model that synthesizes realistic vehicle trajectories given some noise as input [8]. Although GANs do not require the complex calibration of the computational generators, they do need large amounts of data to be trained and, even with their increasing availability, most publicly available trajectory datasets only cover a portion of the population and are limited to crowded regions and active hours of the day. As a result of this data skew, training leads to biased generative models. Another issue is that existing methods use a limited number of metrics that can only evaluate the generated dataset holistically, ignoring how realistic each individual trajectory is on its own.

In this study, we propose a data-driven, model-free, and parameter-less method of generating synthetic yet realistic trajectory datasets using only two inputs: (i) traffic data for the region of interest, and (ii) a target Origin-Destination matrix. Unlike the computational methods, our *D*ata-*D*riven *T*rajectory *G*enerator, dubbed *DDTG*, does not require any kind of calibration before it starts generating trajectories and unlike GANs it does not require an existing dataset to train on. In fact, the input to *DDTG* is aggregated data that is publicly available and free of privacy concerns [9] [10] (aggregate data can also be released under stronger privacy guarantees [11] [12]). Additionally, the datasets that are generated by *DDTG* can be used in lieu of real trajectory datasets for training generative models which makes *DDTG* orthogonal and complementary to GANs. In fact, as we show in our experiments, when real-world and *DDTG*-generated trajectories are combined as training data, the trained model generates higher quality synthetic trajectories.

DDTG disaggregates traffic and OD matrix data into individual trajectories, such that the collection of the generated trajectories results in the same aggregate data. The main challenge is that these aggregate data are time-dependent and hence the number and shape of individual trajectories differ depending on the time of the day. The main intuition behind *DDTG* is that individuals tend to select the fastest (or at least one of the fastest) path to their destination [5]. As we show in our experiments, *DDTG* is capable of generating synthetic datasets that retain the underlying distributions of a real vehicle trajectory dataset more accurately than other methods.

We summarize our contributions as follows:

- We propose *DDTG*, a model-free algorithm that generates synthetic yet realistic vehicle trajectory datasets.
- We propose a set of metrics to evaluate the quality of generated datasets at both holistic and individual levels.
- We conduct comprehensive experiments to evaluate the quality of trajectory datasets generated by *DDTG*.
- We perform a case study in which a GAN model is trained with different seed data (real and *DDTG*-output) and evaluate how the quality of generated datasets is affected.

II. *DDTG*: Data-driven Trajectory Generator

A. Overview

We define a vehicle trajectory $s = \{s^{(1)}, s^{(2)}, \ldots, s^{(M)}\}$ as a time-ordered sequence of spatio-temporal points. Each point $s^{(m)}$ consists of a pair of spatial coordinates (i.e., latitude, longitude) and a timestamp. A synthetic trajectory generator outputs a set $S = \{s_1, s_2, \ldots, s_n\}$ of n trajectories that preserve the mobility characteristics of the real world.

Because raw vehicle trajectories are very often noisy due to errors in localization sensors (e.g., GPS), before a trajectory dataset S is analyzed it is typically pre-processed so that every $s \in S$ is map matched to a road network G and converted into a more interpretable trajectory $\tilde{s}$. The transformed trajectory becomes a time-ordered sequence $\tilde{s} = \{\tilde{s}^{(1)}, \tilde{s}^{(2)}, \ldots, \tilde{s}^{(P)}\}$ where every $\tilde{s}^{(p)}$ consists of the graph edge (i.e., road segment) and the timestamp of arrival at the edge. In essence, the raw trajectory is converted to the corresponding path that the vehicle took.

B. Data Used

Unlike the computational methods that require calibration, and unlike GANs that require large amounts of existing trajectory data, *DDTG* only needs aggregated traffic data and an OD matrix for the region of interest along with the road network. Fortunately, this kind of aggregated data is free of any privacy concerns and very often publicly available as we provide examples below.

Road Network: Encoded as a graph $G = (V, E)$ with every node $v \in V$ representing an intersection and every edge $e \in E$ a road segment. Detailed public datasets are made available for most cities in the world by OpenStreetMap [13].

Traffic Data: Traffic sensors (e.g., loop detectors) are deployed on the road network and measure the average speed of passing vehicles at fixed time intervals. A traffic data point is a tuple $\rho = (l, t, v)$, where l are the spatial coordinates of the sensor, t the timestamp, and v the speed [14] [9].

Origin-Destination Matrix: An $M \in \mathbb{R}^{CxC}$ matrix that corresponds to a grid with $|C|$ cells overlayed over the region of interest. For each pair of cells (c_i, c_j) the OD matrix measures the probability that a trip originates within cell c_i and terminates within c_j. Such data are made available by some providers [15] but can also be estimated either from traffic [16] or trajectory data.

C. The DDTG Algorithm

DDTG operates in two phases. In the first phase, n trip definitions are sampled from a provided OD matrix. A trip definition is a tuple (o, d, t), where o and d are the origin and destination coordinates, respectively, and t is the departure time. Subsequently, in the second phase, the trip definitions are processed and corresponding synthetic trajectories are generated. Every generated trajectory is essentially a path constrained to a road network G and enriched with the travel time on each road segment by using historical traffic data C. We elaborate on each phase in turn.

1) Phase I: *DDTG* starts by sampling n trip definitions from a OD matrix. The pseudocode is shown in Algorithm 1. The distribution specified by M is used to sample a pair of cells (c_i, c_j) (line 4). Subsequently, the exact origin and destination coordinates are randomly sampled within the origin and destination cells (lines 5-6), respectively. Next, a departure time is sampled. This departure time must adhere to the real-world distribution, i.e., a noon departure time is more probable than a 2AM departure time. We capture the distribution as a histogram with bin size ϕ minutes and we sample departure times from it.

The first phase of *DDTG* plays a critical role in generating datasets of high holistic quality (Section III). Specifically, because the origins, destinations, and departure times are

Algorithm 1 Trip Sampler

Require: M origin-destination matrix, $\mathcal{T}$ time domain

1: **procedure** GENERATETRIPDEFINITIONS(n)
2: $S \leftarrow \{\}$
3: **for** $1 \dots n$ **do**
4: $(c_i, c_j) \leftarrow Sample(M)$
5: $o \leftarrow SampleLocation(c_i)$
6: $d \leftarrow SampleLocation(c_j)$
7: $t \leftarrow Sample(\mathcal{T})$
8: $S \leftarrow S \cup \{(o, d, t)\}$
9: **end for**
10: **return** S
11: **end procedure**

Algorithm 2 Trajectory Generator

Require: Road Network G, Traffic Data C, k

1: **procedure** GENERATETRAJECTORY(o, d, t)
2: $u \leftarrow Lookup(G, o)$ ▷ Source node
3: $v \leftarrow Lookup(G, d)$ ▷ Target node
4: $P \leftarrow FindFastestAlt(G, C, u, v, t, k)$
5: $p \leftarrow ChooseMostLikely(P)$
6: **return** p
7: **end procedure**

Algorithm 3 Trajectory Via Stops Generator

1: **procedure** GENERATEVIATRAJECTORY($\vec{L}, t, w$)
2: $\tilde{t} \leftarrow t$
3: **for** $i = 2 \dots |\vec{L}|$ **do**
4: $o \leftarrow \vec{L}_{i-1}$
5: $d \leftarrow \vec{L}_i$
6: $p \leftarrow GenerateTrajectory(o, d, \tilde{t})$
7: **yield** p
8: $\tilde{t} \leftarrow \tilde{t} + p.travel\_time + w_{i-1}$
9: **end for**
10: **end procedure**

sampled from real-world distributions, the more trajectories that *DDTG* generates, the closer it imitates these distributions. In Section IV we show how a bad choice of M (e.g., uniform) affects the holistic quality of the generated dataset.

2) Phase II: After all the trip definitions are generated, the algorithm proceeds to generate corresponding synthetic trajectories. The pseudocode of the second phase is depticted in Algorithm 2. Because *DDTG* directly operates on the road network, the first step is to match the origin and destination locations to their respective closest nodes u and v in G (lines 2-3). One challenge is what path to select as a trajectory. Intuitively, individuals exhibit a strong preference for taking a fast (typically the fastest but not always) path to their destination [5] instead of any other kind of path (e.g., shortest, random walk). However, always using the fastest as a trajectory leads to unrealistic datasets where all objects use the same path. To introduce the diversity of choices and personal preferences, a number of feasible paths from u to v is calculated (line 4). Specifically, the algorithm searches for the k time-dependent fastest paths [17] that have limited overlaps. This is achieved by using a variant of the k-alternative shortest paths algorithm [18] in which the cost function of the route becomes the travel time instead of the travel distance. Finally, one of the k paths is chosen as the actual trajectory (line 5).

The second phase of *DDTG* generates trajectories of high individual quality. Using real-world traffic data, the travel times of the generated trajectories are accurately estimated. This is also confirmed by our experiments (Section IV).

Trajectories in the real-world exhibit a variety of mobility behaviors. For example, delivery trucks very often begin and end their trips at the same location (e.g., post office, warehouse), and their path is a round trip with many quick stops. One challenge with Algorithm 1 is that it assumes that individuals always travel with a purpose, i.e., going directly from one location to another with no stops or detours. Although the focus of this algorithm is on generating this specific type of urban vehicle trajectories, a generalized variant of Algorithm 1 can be used to generate delivery truck-like trajectories. In this generalized variant, trajectories are viewed as a sequence of smaller sub-trajectories $s_1, s_2, \dots, s_w$ in which the destination of each s_i, where $1 < i < w$, is the origin of s_{i+1}. Trip definitions are also redefined to a tuple $(\vec{L}, t, w)$ such that $\vec{L}$ is a sequence of l locations with $\vec{L}_1$ being the origin and $\vec{L}_l$ the final destination, t the trip departure time, and w a sequence of $l-1$ stay durations at each intermittent location along the trip. Algorithm 3 presents the modified trajectory generator. The algorithm iterates over the trip locations (line 3) and generates one trajectory per each consecutive pair of locations (lines 4-6). After every generated trajectory, the departure time for the next trajectory is computed by adding the travel time and stay duration to the current trajectory's departure time (line 8).

Using this variant, it becomes straightforward to model delivery truck trajectories; $\vec{L}_1$ and $\vec{L}_l$ will typically be the warehouse and $\vec{L}_2, \dots, \vec{L}_{l-1}$ will be the delivery locations planned for the truck.

III. SYNTHETIC DATASET EVALUATION

Evaluating the quality of synthetic datasets and how realistic they are is a critical task. On one hand, the generated dataset may look realistic at a high level (e.g., preserving the OD matrix) while the individual trajectories are not reasonable (e.g., unrealistic travel time). On the other hand, the individual trajectories may be realistic while the holistic dataset is not. Therefore, they must be evaluated at two levels; the *holistic* level and the *individual* level. In this section, we describe a set of metrics that can be used for each evaluation level.

A. Holistic Quality

At the dataset level, a synthetic trajectory dataset must retain the collective mobility characteristics of the metropolitan city. Such characteristics include, but are not limited to, the distribution of origins and destinations, the distribution of origin-destination pairs, the distribution of departure times, and the congestion density. Calculating these in a continuous space is computationally intractable, hence, the distributions are estimated by discretizing the space (and time) into an RxC grid with cell size m. The finer the grid, the closer the estimation is to the real distribution but the more computationally expensive. We summarize these metrics below.

- **Distribution of Origins ($p(o)$):** The spatial distribution of the locations at which the trajectories begin. This is typically the first location of a trajectory.
- **Distribution of Destinations ($p(d)$):** The spatial distribution of the locations at which the trajectories end. This is typically the last location of a trajectory and implicitly represents the popularity of locations.
- **Origin-Destination (OD) Matrix ($p(o, d)$):** The distribution of (o, d) pairs, where o is the origin and d is the destination. It captures the probability of visiting one region (i.e., grid cell) from another.
- **Transition Matrix ($p(c_j|c_i)$):** The probability of transitioning from cell c_i to cell c_j. Trajectories can be translated from a sequence of locations to a sequence of grid cells $\tilde{C} = \{\tilde{c}_1, \ldots, \tilde{c}_p\}$. Then, the pairs of consecutive cells in a trajectory can be used to estimate the transition matrix.
- **Distribution of Departures ($p(\tau)$):** The temporal distribution of the timestamps at which the trajectories begin. This is typically the timestamp of the first location of a trajectory.

These distributions are estimated for both the real and generated datasets and compared using Jensen-Shannon (JS) distance, a well-known distance metric for probability distributions (Equation 1).

$$d_{JS}(P, Q) = \sqrt{JS(P||Q)} = \sqrt{\frac{KL(P||M) + KL(M||Q)}{2}} \tag{1}$$

where $JS(P||Q)$ is the Jensen-Shannon divergence, $KL(P||Q)$ is the Kullback-Leibler divergence (Equation 2), and $M = \frac{P+Q}{2}$.

$$KL(P||Q) = \sum_i p_i \log \frac{p_i}{q_i} \tag{2}$$

The value of d_{JS} ranges from 0 to 1, where $d_{JS} = 0$ when the two distributions are identical and $d_{JS} = 1$ when they are completely different.

Previous work mostly focuses on $p(d)$ to evaluate the holistic quality of generated datasets [19] [20]. The authors in [8] consider $p(o)$ and $p(o, d)$ in addition to $p(d)$. In this work, we also consider the transition matrix $p(c_j|c_i)$ and distribution of departures $p(\tau)$ to evaluate more aspects of how realistic the generated dataset is. The number of individuals traveling at night is significantly lower than the number of individuals traveling in the day. This important characteristic is captured and evaluated using the $p(\tau)$ metric.

B. Individual Quality

At the individual level, the trajectories themselves must be examined for their quality. For example, the travel time of a synthetic trajectory must be within an acceptable range of the actual travel time. Additionally, the path from the origin to the destination must be reasonable, i.e., not straying far away from the destination or driving in loops. We summarize these metrics below.

- **Travel time:** The generated trajectory must be consistent with the traffic dynamics. Not only the overall travel time but also the travel time of each segment of the trajectory must be reasonable. This can be verified using traffic data.
- **Deviation:** The generated trajectory should be mostly en route to the destination, i.e., it should not be taking detours or heading in a direction that is opposite to the destination. One way to evaluate this is by using Markov Models to check whether the transitions from one location to the next make sense. Another way when a reference dataset exists is to retrieve real trajectories that are similar (i.e., identical origin and destination, similar departure time) and calculate the dissimilarity. A low value indicates that the generated trajectory does not differ much from a typical trajectory. A large value indicates that the generated trajectory follows a path that is not common (or popular).

IV. Experimental Evaluation

In this section, we quantitatively and qualitatively evaluate synthetic trajectories generated by *DDTG* with experiments on real-world data.

A. Environment Setup

All experiments were performed on an Ubuntu 18.04 server equipped with an Intel(R) Core(TM) i9-9980XE CPU at 3.00GHz and 128GB of RAM. The processor has 18 cores (36 threads) with private L1 (32KB for data and 32KB for instructions) and L2 (1MB) caches and a shared L3 (24.75MB) cache. The server is also equipped with a GeForce RTX 2080 Ti GPU card.

Datasets:

We limit our study to the metropolitan region of the city of Los Angeles.

(1) We use a real-world trajectory dataset from Veraset as the ground truth for evaluation. The dataset contains tens of millions of trajectories out of which approximately 4 million trajectories are within our region of interest. We pre-process these trajectories and only extract the segments during which the user was traveling (details omitted due to space constraints). The resulting set contains 1.5 million trajectories spanning the first two weeks of December 2019.

(2) We use a real-world historical traffic dataset exported from ADMS [14]. The dataset contains approximately 16,000 sensors covering more than 4,300 miles in the region of interest. The sensors have a refresh rate of up to 60 seconds but we aggregate the traffic speed data into 15-minute intervals.

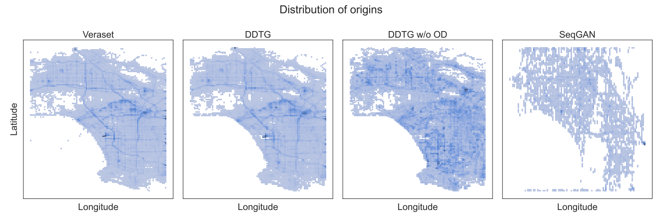

Fig. 2. Distribution of origins in the real dataset (left) and the synthetic datasets generated by *DDTG* (center-left), *DDTG* without OD matrix information (center-right), and SeqGAN (right). *DDTG* imitates the real distribution with a higher accuracy.

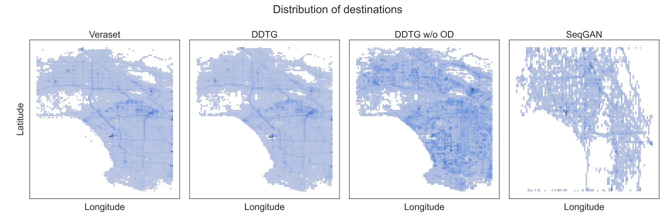

Fig. 3. Distribution of destinations in the real dataset (left) and the synthetic datasets generated by *DDTG* (center-left), *DDTG* without OD matrix information (center-right), and SeqGAN (right). *DDTG* imitates the real distribution with a higher accuracy.

Furthermore, for every road segment, i.e., edge in the road network graph, we calculate its time-dependent travel time by querying the traffic data for the speed. If traffic speed is not sensed at a particular road segment, we obtain an estimation of the traffic at that segment using interpolation methods [21].

(3) The OD matrix for the region of interest is calculated from the trajectory dataset. We use a grid with a cell size of 500 meters. The cell that contains the first (last) GPS reading of a trajectory is considered the origin (destination).

Compared Approaches: We compare the real-world trajectory dataset from Veraset with 3 synthetic datasets.

- **SeqGAN:** A synthetic dataset of 1.5 million trajectories generated by a sequential GAN model [22]. The real trajectory dataset is used as training data for the model.
- ***DDTG* w/o OD:** A synthetic dataset of 1.5 million trajectories generated by *DDTG* without OD distribution information, i.e., all OD pairs have equal probability.
- ***DDTG*:** A synthetic dataset of 1.5 million trajectories generated by *DDTG* using an OD matrix as input that was extracted from the Veraset dataset.

B. Quality Analysis

We evaluate the quality of synthetic datasets at two levels using the metrics described in Section III.

Holistic Quality

First, we compare the origin distributions. The JS distances between Veraset's origins and synthetic origins are 0.1629, 0.3466, and 0.9299 for *DDTG*, *DDTG* w/o OD, and SeqGAN, respectively. The distance between the real and *DDTG* distributions is very close to 0 which means that the two distributions are very similar. In contrast, the distance of the *DDTG* w/o OD is two times larger meaning that it is significantly different from the ground truth.

Next, we compare the destination distributions of the datasets. Figure 3 shows that Veraset and *DDTG* destination distributions only have slight differences. In Veraset, we can see a few hot spots close to the region's eastern border, whereas, in *DDTG*, not as much. This is mostly attributed to the cleaning method that was employed for the real dataset, i.e., trajectories that had a destination outside the region of interest were cut short and their destination was assumed to be at the boundary. The JS distances are 0.1326, 0.3292, and 0.9086 for *DDTG*, *DDTG* w/o OD, and SeqGAN, respectively. Even though there are minor differences in the destination hot spots of Veraset and *DDTG*, the majority of the remaining grid cells appear to incur similar probabilities of being destinations.

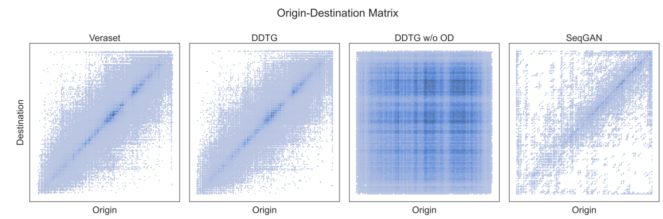

Fig. 4. Origin-destination matrix calculated from the real dataset (left) and the synthetic datasets generated by *DDTG* (center-left), *DDTG* without OD matrix information (center-right), and SeqGAN (right). *DDTG* imitates the real distribution with a higher accuracy.

An interesting observation is that in the real (Veraset) and SeqGAN datasets, some origins and destinations appear to exist in the middle of the ocean (bottom left sub-region), whereas, in DDTG-generated synthetic datasets no such cases exist. In the real dataset, this is due to the noise that GPS sensors naturally incur. In the case of SeqGAN, this happens because the model has no information about the road network.

Additionally, we note that SeqGAN generates very sparse origin and destination distributions. In essence, the model learns the skewed distribution of the input training dataset, i.e., trips concentrated in certain regions. As a result, the model is biased toward generating trips in those areas instead of the entire region.

We also compare the OD matrices of the datasets. Figure 4 visualizes the distributions. On the x-axis is the origin and on the y-axis is the destination. Again, as expected, the *DDTG* w/o OD information produces almost uniform results. SeqGAN achieves a visually similar distribution but with a great amount of noise. *DDTG*, on the other hand, imitates the true distribution almost perfectly. Some OD pairs that appear more prominently in the ground truth, are slightly smoothed out by *DDTG* due to the generator's randomness. However, the general distribution is preserved. The JS distance between Veraset's OD Matrix and *DDTG*'s is 0.3223 which, considering the vast amount of OD pairs, is close to zero.

As shown in Table I, *DDTG* generates trajectories that retain the transition matrix more accurately than the others. Specifi-

TABLE I
JENSEN-SHANNON DISTANCE OF THE SYNTHETIC DATASET DISTRIBUTIONS AND THE REAL DATASET DISTRIBUTIONS.

Distribution	Jensen-Shannon distance d_{JS}		
	DDTG	*DDTG* w/o OD	SeqGAN
$p(o)$	**0.1629**	0.3466	0.9299
$p(d)$	**0.1326**	0.3292	0.9086
$p(o, d)$	**0.3223**	0.8881	0.9356
$p(c_j \| c_i)$	**0.2348**	0.3084	0.5928
$p(\tau)$	**0.0158**	0.2609	-

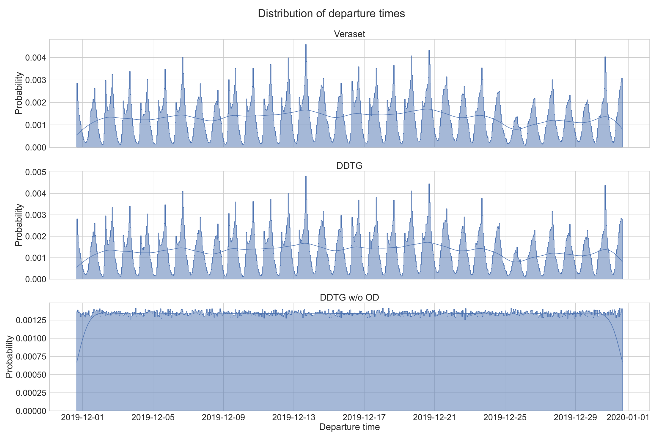

Fig. 5. Departure time distributions calculated from the real dataset (top), the synthetic dataset generated by *DDTG* (middle), and the synthetic dataset generated by *DDTG* without OD matrix information (bottom).

cally, the JS distance of *DDTG*'s and the real transition matrix is 0.2348, whereas SeqGAN's is almost three times greater at 0.5928. This happens because *DDTG* is able to leverage the road network and generate high-resolution trajectories that retain the properties and constraints of the road network. On the other hand, trajectories generated by SeqGAN are more likely to contain noise, i.e., jumps from one cell to another while skipping cells in between.

Lastly, we compare the distributions of departure times. Figure 5 plots the probability (y-axis) of each departure time (x-axis). Not surprisingly, the distribution for *DDTG* w/o OD is uniform. On the other hand, *DDTG*'s distribution is almost identical to that of the real dataset. These results are confirmed by the JS distance values (Table I). We omit the departure times for SeqGAN because the model is highly discretized in the temporal domain.

Holistically, the dataset generated by DDTG retains the mobility characteristics of the real dataset.

Individual Quality

We verify how realistic each individual trajectory is by comparing how much it resembles common paths from origins to destinations. Specifically, we randomly sample 50,000 synthetic trajectories generated by *DDTG* and for each trajectory, we retrieve all trajectories with identical origin and destination cells and similar departure times. Next, the average distance between a synthetic trajectory and its corresponding real trajectories is calculated. For our experiments we use two distance metrics; the Frechèt distance and the Hausdorff distance. Figure 6 presents the histograms of these distances. The lower the distance, the closer the synthetic trajectory is to the common real trajectories. For both distance metrics, the histogram is skewed toward 0. This means that the generated trajectories are reasonably similar to the paths that individuals tend to follow. More critically, the generated trajectories are not exactly identical to the real paths. This shows that, while the individual trajectories make sense, they still protect the privacy of individuals.

Figure 7 presents six examples of real trajectories (green lines) and their corresponding synthetic trajectories (red lines). Origins are marked with blue pins and destinations with red pins. In the first three examples, the generated trajectory is very similar to the real trajectory. This happens because the real path taken by the individual was one of the fastest (if not the fastest) and *DDTG* itself uses that information to generate trajectories. In the last three examples, the generated paths are quite different. This can happen either because *DDTG* chose a different fastest path to the destination or because the real path itself was a detour and not one of the fastest paths.

C. Case Study: Improving the quality of GAN-based Synthetic Trajectories

As we summarize in Table I, the SeqGAN model trained from real-world data (REAL-based) generates lower quality trajectories than *DDTG*. The main reason behind this is the lack of diversity, or in other words the skewness of the real dataset. In an attempt to alleviate this, we use *DDTG*'s output as the training set of the SeqGAN model. The result is a slightly better model with an OD matrix JS distance of 0.8242 as compared to 0.9356 of the REAL-based model. Although the quality improves, the *DDTG*-based model now lacks real-world experiences, i.e., trajectories that are outside the mobility behaviors simulated by *DDTG*. To this end, we use a combination of real-world and *DDTG* generated trajectories to train a SeqGAN model. As a result, the JS distance of trajectories generated by this model drops to 0.3995. This shows how*DDTG* can be an orthogonal component for GAN-based trajectory generation models. *DDTG* can essentially be used as an intermediate method for augmenting an existing real-world trajectory dataset before it is used for training generative models.

Figure 8 presents the spatial distribution of destinations $p(d)$ for the datasets generated by the three models. On the left, we observe the REAL-based model generating somewhat sparse trajectories with a few popular destinations (hotspots). The *DDTG*-based output is denser and evenly distributed, however, the popular destinations are still not showing up as prominently as they should (see Figure 3). Lastly, similar to the *DDTG*-based, the combined output is dense and evenly distributed while also showing signs of popular destinations being correctly sampled. One common observation for all three models is that in certain cases, a destination in the middle

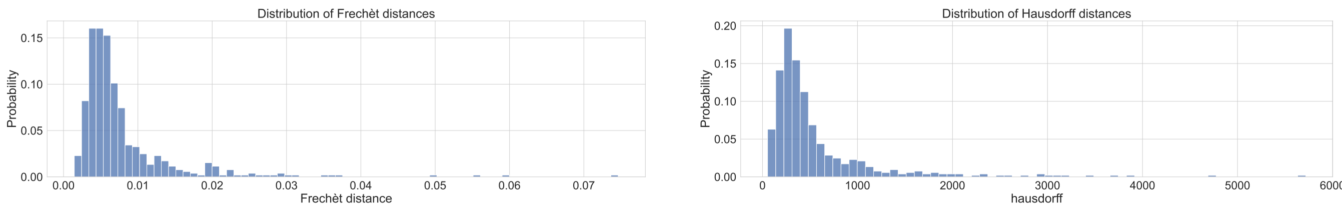

Fig. 6. Distribution of Frechèt (left) and Hausdorff (right) distances between generated and corresponding real trajectories.

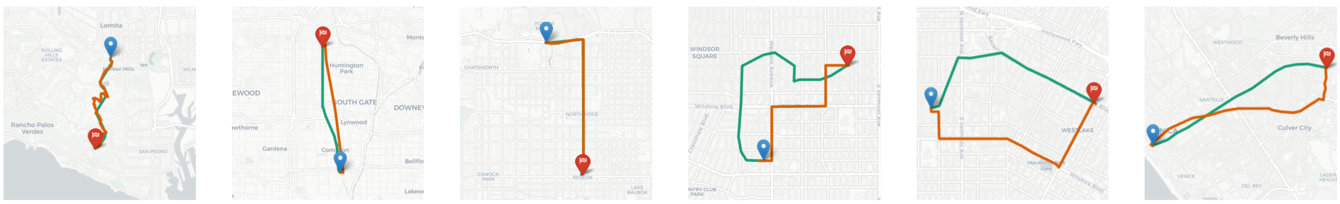

Fig. 7. Examples of real trajectories (green lines) and corresponding synthetic trajectories (red lines) generated by *DDTG*.

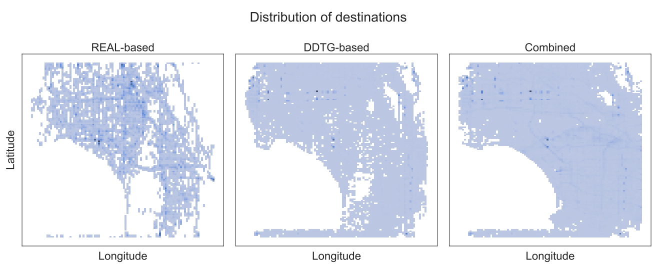

Fig. 8. Distribution of destinations ($p(d)$) of trajectories generated by the REAL-based (left), *DDTG*-based (center), and Combined (right) SeqGAN models.

of the ocean is sampled by the model (bottom left). This is due to the inherent noise and randomness of the model but is exacerbated in the DDTG-based and combined models where the output becomes denser.

V. Related Work

In this section, we summarize related work and their shortcomings and discuss how our method addresses them.

A. Computational Generators

One of the first studies in spatio-temporal moving object generators (MOG) is [23] with the goal to generate artificial trajectories with realistic spatiotemporal properties. These early MOGs, however, generate trajectories in free space but vehicle trajectories are by definition constrained to a road network. To overcome this, network-constrained generators have been proposed. The most widely cited attempt of generating moving objects constrained to a network in a realistic way is proposed in [5]. The framework selects origin and destination nodes and routes an agent along the network with limited interactions with other agents on the network. Perhaps the most prominent simulator is the Simulation of Urban MObility (SUMO) framework [24]. SUMO is a microscopic traffic simulator that generates and exports trajectories by simulating the environment from three inputs: road network, traffic infrastructure, and traffic demand. It also allows the user to specify a variety of parameters such as traffic infrastructure (e.g., traffic lights), number of lanes, and vehicle types.

Most of the aforementioned methods are highly configurable. However, to generate realistic trajectories, their parameters must be calibrated in order to match the target real-world environment (e.g., probability of simulated agent changing lanes, traffic lights operation); a task that is often time-consuming and requires domain knowledge. Additionally, as the region of interest becomes larger, e.g., at the scale of a metropolitan city, simulators require large amounts of computational power, a fact that led to the proposal of distributed architectures [25] [26]. *DDTG* does not require any calibration or time-consuming configuration before it can generate realistic vehicle trajectories and can easily scale to metropolitan-sized road networks. *DDTG*'s algorithm can also be parallelized to generate millions of trajectories in minutes (details omitted due to space constraints).

B. Data-Driven Generators

Computational generators, as mentioned above, require the calibration of complex parameters. To address this, non-parametric machine learning methods have been proposed to synthesize trajectories. In the earliest attempt [27], an LSTM architecture is proposed for predicting movements from GPS traces. Because very often the training dataset size is limited due to privacy and other constraints, the authors in [28] use a variational autoencoder to map the input to a hidden space where the characteristics can be preserved.

More recently, Generative Adversarial Networks (GAN) have been proposed as the state-of-the-art in generating synthetic trajectories. In [29], the authors discretize the input in both the spatial and temporal domains and train a non-parametric generative model. The discretized output is then converted back into location trajectories. [30] proposes a self-attention-based sequential modeling network as the generator to encode the complex temporal transitions of human mobility. The authors also make use of a mobility regularity-aware loss in the discriminator to help it distinguish between real and synthetic trajectories. In [8] the authors formulate the task as an imitation learning problem in a partially observable Markov decision process. In [19] the GAN is trained with a new loss function, named TrajLoss, that accounts for the dissimilarity of generated trajectories to the input.

Generative models require large amounts of data in order to learn how to generate realistic trajectories. However, even in big trajectory datasets, the distribution tends to be skewed, i.e., limited to only certain regions of the metropolitan city and hours of the day, and hence the trained models are biased. Another shortcoming of these methods is that the optimization objective is limited to the holistic level quality of the generated dataset and the quality of individually generated trajectories is mostly ignored. *DDTG* only requires the traffic data and a target OD matrix for the region of interest, both of which are very often publicly available and free of privacy concerns. The synthetic datasets generated by *DDTG* follow distributions that are very close to those of real trajectory datasets.

VI. Conclusion

In this paper, we proposed *DDTG*, a data-driven, model-free, and parameter-less algorithm for generating realistic vehicle trajectory datasets. Unlike existing approaches, *DDTG* only requires two main inputs: the traffic data, and the OD matrix for the region of interest. These aggregated inputs are very often publicly available because they either do not reveal any private information or if needed can be sanitized further with differential privacy. As we showed in our experiments, datasets generated by *DDTG* imitate the distributions of a real trajectory dataset. Additionally, we show empirically that *DDTG* is orthogonal to existing state-of-the-art approaches and in tandem with real data it can complement them in generating synthetic datasets of higher quality.

Acknowledgements

This research has been funded in part by NSF grant CNS-2125530, USDOT contract PSR-201603605-02, LA Metro contract PS75498000, the USC Integrated Media Systems Center, the USC METRANS Transportation Center, and an unrestricted gift from Google and Microsoft.

References

[1] J. Kim and H. S. Mahmassani, "Spatial and temporal characterization of travel patterns in a traffic network using vehicle trajectories," *Transportation Research Procedia*, vol. 9, 2015.
[2] Q. Huang and Y. Xiao, "Geographic situational awareness: mining tweets for disaster preparedness, emergency response, impact, and recovery," *International Journal of Geo-Information*, vol. 4, no. 3, 2015.
[3] C. Keßler and G. McKenzie, "A geoprivacy manifesto," *Transactions in GIS*, vol. 22, no. 1, 2018.
[4] C.-Y. Chow and M. F. Mokbel, "Trajectory privacy in location-based services and data publication," *ACM Sigkdd Explorations Newsletter*, vol. 13, no. 1, 2011.
[5] T. Brinkhoff, "A framework for generating network-based moving objects," *GeoInformatica*, vol. 6, no. 2, 2002.
[6] J. Barceló, E. Codina, J. Casas, J. L. Ferrer, and D. García, "Microscopic traffic simulation: A tool for the design, analysis and evaluation of intelligent transport systems," *Journal of intelligent and robotic systems*, vol. 41, no. 2, 2005.
[7] Y. Hollander and R. Liu, "The principles of calibrating traffic microsimulation models," *Transportation*, vol. 35, no. 3, 2008.
[8] S. Choi, J. Kim, and H. Yeo, "Trajgail: Generating urban vehicle trajectories using generative adversarial imitation learning," *Transportation Research Part C: Emerging Technologies*, vol. 128, 2021.
[9] "Pems," https://pems.eecs.berkeley.edu/, accessed: 2022-08-13.
[10] P. Krishnakumari, H. van Lint, T. Djukic, and O. Cats, "A data driven method for od matrix estimation," *Transportation Research Procedia*, vol. 38, 2019.
[11] S. Zeighami, R. Ahuja, G. Ghinita, and C. Shahabi, "A neural database for differentially private spatial range queries," *Proceedings of the VLDB Endowment*, vol. 15, no. 5, 2022.
[12] S. Shaham, G. Ghinita, and C. Shahabi, "Differentially-private publication of origin-destination matrices with intermediate stops," *Proceedings of the 25th International Conference on Extending Database Technology (EDBT)*, 2022. [Online]. Available: https://par.nsf.gov/biblio/10333510
[13] "Openstreetmap," https://openstreetmap.org, accessed: 2022-08-13.
[14] C. Anastasiou, J. Lin, C. He, Y.-Y. Chiang, and C. Shahabi, "Admsv2: A modern architecture for transportation data management and analysis," in *Proceedings of the 2nd ACM SIGSPATIAL International Workshop on Advances on Resilient and Intelligent Cities*, 2019.
[15] "Veraset," https://www.veraset.com, accessed: 2022-08-13.
[16] T. Abrahamsson, "Estimation of origin-destination matrices using traffic counts: A literature survey," *IIASA Interim Report*, 1998.
[17] U. Demiryurek, F. Banaei-Kashani, C. Shahabi, and A. Ranganathan, "Online computation of fastest path in time-dependent spatial networks," in *International Symposium on Spatial and Temporal Databases*. Springer, 2011.
[18] T. Chondrogiannis, P. Bouros, J. Gamper, U. Leser, and D. B. Blumenthal, "Finding k-shortest paths with limited overlap," *The VLDB Journal*, vol. 29, no. 5, 2020.
[19] J. Rao, S. Gao, Y. Kang, and Q. Huang, "Lstm-trajgan: A deep learning approach to trajectory privacy protection," *arXiv preprint arXiv:2006.10521*, 2020.
[20] N. Xu, L. Trinh, S. Rambhatla, Z. Zeng, J. Chen, S. Assefa, and Y. Liu, "Simulating continuous-time human mobility trajectories."
[21] C. Anastasiou, J. Zhao, S. H. Kim, and C. Shahabi, "Data-driven traffic index from sparse and incomplete data," *IEEE Transactions on Intelligent Transportation Systems*, 2022.
[22] L. Yu, W. Zhang, J. Wang, and Y. Yu, "Seqgan: Sequence generative adversarial nets with policy gradient," in *Proceedings of the AAAI conference on artificial intelligence*, vol. 31, no. 1, 2017.
[23] Y. Theodoridis, J. R. Silva, and M. A. Nascimento, "On the generation of spatiotemporal datasets," in *International Symposium on Spatial Databases*. Springer, 1999.
[24] P. A. e. a. Lopez, "Microscopic traffic simulation using sumo," in *21st International conference on intelligent transportation systems (ITSC)*. IEEE, 2018.
[25] H. Xie, E. Tanin, K. Ramamohanarao, S. Karunasekera, L. Kulik, R. Zhang, and J. Qi, "Generating traffic data for any city using smarts simulator," *SIGSPATIAL Special*, vol. 11, no. 1, 2019.
[26] J. Yu, Z. Fu, and M. Sarwat, "Dissecting geosparksim: a scalable microscopic road network traffic simulator in apache spark," *Distributed and Parallel Databases*, vol. 38, no. 4, 2020.
[27] X. Song, H. Kanasugi, and R. Shibasaki, "Deeptransport: Prediction and simulation of human mobility and transportation mode at a citywide level," in *Proceedings of the Twenty-Fifth International Joint Conference on Artificial Intelligence*, 2016.
[28] D. Huang, X. Song, Z. Fan, R. Jiang, R. Shibasaki, Y. Zhang, H. Wang, and Y. Kato, "A variational autoencoder based generative model of urban human mobility," in *2019 IEEE Conference on Multimedia Information Processing and Retrieval (MIPR)*. IEEE, 2019.
[29] K. e. a. Ouyang, "A non-parametric generative model for human trajectories." in *IJCAI*, vol. 18, 2018.
[30] J. Feng, Z. Yang, F. Xu, H. Yu, M. Wang, and Y. Li, "Learning to simulate human mobility," in *Proceedings of the 26th ACM SIGKDD international conference on knowledge discovery & data mining*, 2020.