Privacy Project: Simulating Data for Trip Reconstruction

Kirk Boyer kirk.boyer@gmail.com, Hao Chen haozi@haozi.name, Jingwei Chen chenjingwei1991@gmail.com, Jian Qiu qiujian3328103@gmail.com

ABSTRACT

Location privacy is one of the main challenges in vehicular networks. Modern technology allows for automatic recognition, or tracking, of a vehicle, often using license plate numbers, at fixed locations. Various parties have a vested interest in making use of the kind of data produced by this recognition, in particular to try to analyze the driving habits and routes taken by people who pass by. In this paper we present a procedure to meaningfully simulate traffic data and explore the implications of this type of travel-data collection. We show that even when restricted to information only about the structure of a road network, one can begin to set up an effective network of tracking devices to infer the travel patterns of individual drivers. We also develop a method to find good locations of trackers to reconstruct the real path of a vehicle in such a network, and describe a measure of the quality of an attempted reconstruction of a real path called the *k*-bubble quality.

Keywords: road networks, trip reconstruction, driving privacy

1. Introduction

A recurring theme in privacy is that the development of technologies that provide new useful forms of data often inadvertently provide access, even if indirectly, to information that is much more personal than originally intended or hoped. One such technology is automatic vehicle identification (AVI), which is used, for example, to streamline toll collection on toll roads by eliminating the need to stop at a collection kiosk [2]. What may not be immediately obvious is that information from disparate AVI devices can be used in conjunction with each other to discover much less locally constrained information.

We are interested in the problem of using AVI technology to reconstruct the path that people drive on a given trip [3]. Most AVI devices use a camera to read a vehicle's license plate [4], thereby providing a timestamped record that the vehicle was at the location of the device. Since there can be other technologies that accomplish this, such as closed-circuit television (CCTV) or road-rule enforcement cameras, we refer to any of these technologies more broadly as "trackers." By using such devices, people can use the Automatic Number Plate Recognition (ANPR) method to read vehicle registration plates . Usually, license plate information can be stored as a captured image or as text [6]. In this paper we consider a generalized setting where we can collect any of these kinds of data.

The first necessity in making use of AVI technology or the ANPR method is to decide where to place trackers. Ideally, this choice would be a function of real traffic patterns, but in the absence of detailed information about traffic patterns, we choose to simulate it.

Real traffic data might take the form of the frequencies with which roads are taken, or the relative frequencies with which a direction is chosen at a given intersection [5]. Also, when people drive, they start somewhere and they (almost always) have a particular destination in mind. So one way to simulate having traffic data is to create a collection of directed, but wandering, paths.

Once trackers are placed, they are used to guess paths taken by creating a timestamp-ordered sequence of trackers crossed by a given vehicle in a relatively small timeframe. Practically, this means taking a timestamp-ordered collection of trackers and deciding which path was likely to have been taken by the vehicle's driver.

After simulating traffic data, there is the question of how to choose the set of locations for a collection of trackers based on the data. In this context, there are the competing interests of placing trackers at frequently passed locations, and preventing clustering of trackers; the latter is a potential algorithmic pitfall when giving preference to high-traffic location because high-traffic locations are unlikely to be isolated, and placing trackers too close together can lead to unhelpful redundancy. (We do not consider factors such as resilience of the set of trackers to failure of one or more of the trackers.)

The questions we attempt to answer are: 1) How should traffic data be simulated, given only the structure of a road network? 2) How could traffic data be used to place trackers so they are most useful in guessing where people went?

In this project we used real data about the structure of the road network around Denver, CO, as the basis for our simulations. In Section 2 we will describe the real data we used, how we preprocessed it, and how we built simulated data on top of it. In section 3 we will describe what this data looks like and examine the quality of trackers it produced. In section 4, we will address the project's questions and discuss further avenues for exploring these same questions.

2. Methodology

2.1 Reducing the Road Network Data

The Denver road network data set originally came from OpenStreetMap.org in the form of XML formatted .osm files, which were then processed into the format we used by Dewri et. al. [1]. This format includes the edge list of a directed graph with 323928 nodes and 639395 edges, the lengths (in nautical miles) and classifications of those edges, and the latitudes and longitudes of the nodes.

The scale of this data makes graph computation very time-intensive. Because our goal is to develop reasonable simulated traffic data and to place trackers in a general setting, a smaller map should still capture a realistic enough setting. So, our first step is to restrict the map by latitude and longitude: Our traffic nodes have latitudes between 39.790931 and 39.654518, and longitudes between -105.053195 and -104.867402. The subgraph restricted to this rectangle has only 40253 nodes.

To further reduce the size of the graph, we made use of the observation that many of the nodes in the graph were not road intersections, but instead road interiors. In other words, these nodes simply mark subdivisions of roads, not places where a driver could change to a new road. Because a tracker placed at an intersection at either end of this road will capture the information that a driver passed through, there is no additional benefit to placing another tracker along the road. Moreover, trackers placed at intersections will be activated as part of other paths that do not go along that particular road.

The kind of non-intersection nodes we want to remove all have two neighboring nodes, but it is important to observe that removing all nodes with fewer than 3 neighbors does not accomplish what we want. For example, nodes with only one neighbor are dead-end nodes, and are not road interiors in the way described above. Because the graph is directed, there may be nodes with only two neighbors that we still would not want to remove: if a node has one incoming edge and two outgoing edges, this represents a road that is one-way on one side and two-way on the other side. Removing such a node would allow paths that could not have occurred in the original graph. The same applies for a node with two incoming edges but only one outgoing edge. A final type of node we would not want to remove is one with two neighbors, but that is of in-degree zero or out-degree zero.

Analysis of a graph without the aforementioned redundant non-intersection nodes would not lose robustness over the original graph. This redundant node elimination allowed a reduction of the graph to the final 13029-vertex, 39375-edge graph used in our experiments.

The actual process of reduction is a single sweep over the vertices; if a node has two neighbors and out-degree equal to its in-degree, we perform an elimination by connecting its neighbors directly to each other in the same direction as the original edges, and removing the center node and the edges connecting to it. Below are plots of the nodes by longitude and latitude.

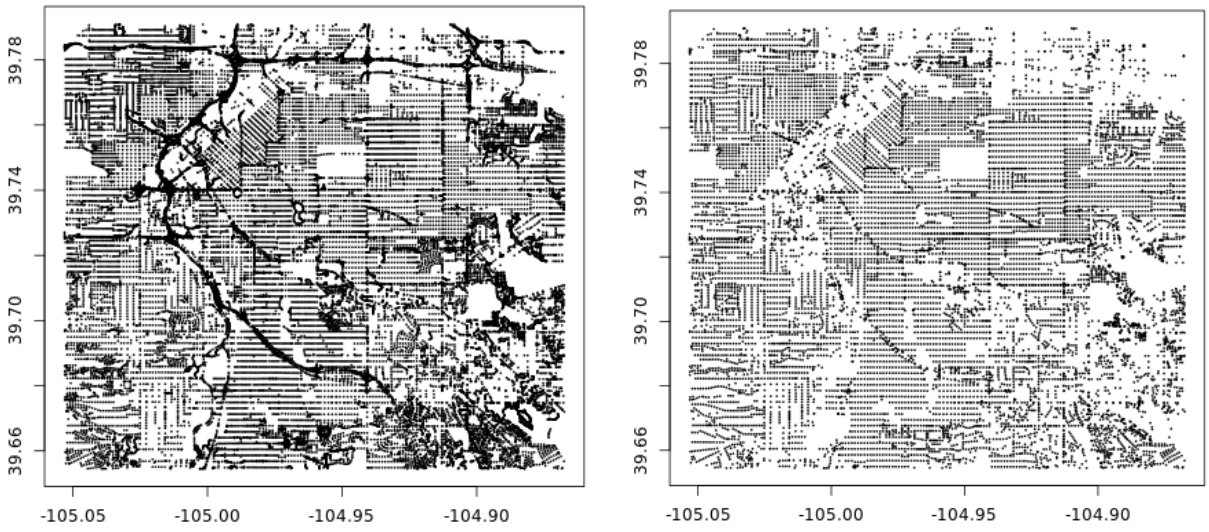


Figure 1: The nodes of the graph

(a) Original (b) Reduced

2.2 Edge Probabilities

Our first attempt to simulate traffic data was to create a very long random walk using uniform probabilities at each node, with the hope that this would give good node frequency data to work with. Creating such a random walk was straightforward using the R package “igraph”, which has a built-in function for this. But we abandoned this approach because given that it’s improbable that real traffic data would ever have a uniform distribution of probabilities, this kind of data is unlikely to be helpful.

Toward a solution to this issue, we decided to assign a probability to each outgoing edge to somehow simulate the real traffic data: when a driver reaches an intersection node, he must choose one neighboring node to go to next (unless he has reached his destination). With access to real data, it would be desirable to compute, for each intersection, the probability that a driver would choose each option. This real probability would represent how busy the corresponding road is with respect to the adjacent roads; at an intersection of a main road and a local road, for example, more traffic will go to the main road, which means the main road gets a higher probability. Without access to real data, we are left with either using heuristics to guess which roads are more likely to be chosen, or to choose randomly. In this experiment, we chose the latter.

Since we want to simulate the real probability, we set 2 rules for assigning random probability:

(1) For each node, no outgoing edge is more likely to be assigned a higher probability value.
(2) For each node, the spread in probabilities amongst its outgoing edges should not be too large. For example, for a node with four outgoing edges, we don’t want one edge with probability 97% and other three edges with probability 1%.

The following is our process for generating random probabilities so that they satisfy the above criteria.

Fix a node x, let n be the number of outgoing edges from x, and let p_i be the probability to be assigned to the i^{th} outgoing edge from x. Fix Δ; this last parameter bounds the ratio between different p_i, and should be at least 3 (in our experiments it is 3).

1. Choose $c_1, c_2, ...c_n$ uniformly from the interval $[-\frac{1}{\Delta n}, \frac{1}{\Delta n}]$.
2. Let $p_i = c_{i+1} - c_i + \frac{1}{n}$,for $i = 1, 2, ..., n-1$, and $p_n = c_1 - c_n + \frac{1}{n}$.

This guarantees that p_i is drawn from a uniform distribution in the range $[\frac{\Delta-2}{\Delta n}, \frac{\Delta+2}{\Delta n}]$, for $i = 1, 2, ...n$. Also, $\frac{max(p)}{min(p)}$ is bounded by $\frac{\Delta+2}{\Delta-2}$.

Using the map with probabilities assigned, we generated another long walk. It visits most of the nodes at least once, and as before gives some information about the likelihood that nodes would be visited by traffic. But there is still a fundamental problem with using a long random walk, in that no driver will ever really travel such a long distance consecutively within a small region of the map, and in particular, wouldn’t wander fully aimlessly.

To truly attain the information that these new edge probabilities can give us about how frequently drivers would pass through nodes, we needed to simulate the kind of paths real drivers would take, which is a path that starts somewhere, ends elsewhere, and most importantly, somehow goes in the general direction of the destination most of the time. A simulation of this is what we call a directed wandering path.

2.3 Generating Directed Wandering Paths

A directed wandering path is a path connecting a starting location with a particular destination, but which is chosen sequentially as a random walk with probabilities based on the underlying probabilities from frequency data. This is accomplished by scaling the original probabilities based on a general "direction sense;" directed edges with smaller angles from the vector pointing toward the destination are weighted higher than directed edges with larger angles.

Before choosing endpoints for the trips, we first divided the region into 140 artificial neighborhoods. In real data, these neighborhoods might come from municipality borders, population densities, general examinations of traffic patterns, or other information. Abstractly, different neighborhoods represent different sub-regions with which a regular traveler might be familiar.

In our experiments, starting locations and destinations were chosen by picking starting and ending neighborhoods uniformly at random and then choosing random nodes within the neighborhoods. During the random walk, at each node, the probabilities on the edges were scaled based on the new vector toward the destination, until the current node was in the destination's neighborhood. When in the neighborhood, we switched to following the weighted graph-shortest-path, because a neighborhood represents familiar enough territory to either a regular traveler or someone with a GPS (someone with a modern GPS might still travel as the random wanderer since the chosen route often depends on real-time traffic information, which varies as the probabilities themselves do).

The procedure to create a directed wandering path is as follows, using latitude and longitude as x- and y-coordinates, respectively. Fix d, the direction vector from the start location to the destination.

1. When at an intersection node v, let $P_v = (p_1, p_2, ...p_n)$ be the respective underlying probabilities of the n edges adjacent to v, and let $E_v = (e_1, e_2, ..., e_n)$ be these edges considered as vectors in the plane.
2. Let $\Lambda_v = (\alpha_1, \alpha_2, ...\alpha_n)$ be such that each α_i is the smallest angle between e_i and d.
3. Let $SF = (sf_1, ..., sf_n)$, where $sf_i = \frac{1}{180}(\varepsilon - 1)\ \alpha_i + 1$ for $i = 1, ..., n$ and $0 < \varepsilon < 1$. Note that as α_i increases from 0 to 180, sf_i decreases from 1 to ε.
4. The resulting probabilities that will be used to choose the next wandering step are $P'' = (p_1'', ..., p_n'')$, where $P' = (p_1', ..., p_n')$, $p_i' = p_i \cdot sf_i$ for $i = 1, ..., n$, and $P'' = \frac{1}{\sum_{i=1}^{n} p_i'} P'$. The conversion from P' to P'' is simply to normalize the values to probabilities that sum to 1.

The vectors in E_v can be pre-processed for all v in the graph, so that they are not re-computed when a node appears in a different directed wandering path. We can also perform the scaling in step 4 repeatedly on the result, if we want to more strongly emphasize the effect of pointing in the wrong direction.

In our implementation, we choose $\varepsilon = 0.1$ and apply step 4 twice. Also, while we fix d in these experiments, it is preferable to let d vary as the vector from the current node to the destination. We computed 7394 directed wandering paths in this fashion, and split the data into a training set of 6161 paths and a validation set of 1233 paths.

2.4 Reconstructing Paths

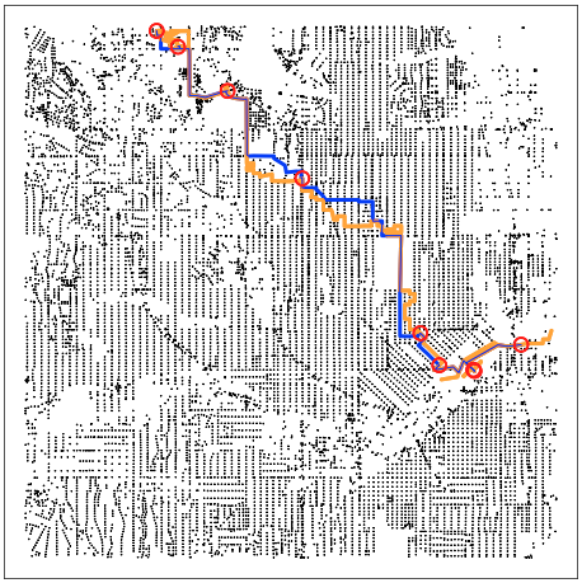


Figure 2-1: A path (orange) crossing tracers (red) and its reconstruction (blue)

Before choosing locations for trackers, we need to examine how they would be used. When a person takes a trip through an area scattered with trackers, some of the locations they pass through contain trackers and some don't. If the trackers record the time at which they see this traveler, their order can be determined. Making the assumption that, between trackers, people are most likely to have taken the shortest path, we connect consecutively activated trackers with these shortest paths and concatenate them to get a combined predicted path, which is our guess about what the actual trip was based on the number of trackers crossed.

One way of choosing a connecting "shortest path" is just by minimizing the distance traveled. This is a strong choice because people will often choose to drive the shortest path in the short term, because this is an easy optimization for someone at least moderately familiar with the surrounding area, and is often a good heuristic for minimizing travel time. To determine whether we have successfully reconstructed a path, one way to measure its quality is to count how many nodes in the reconstructed path are also in the actual path. This is the first method we used, referred to as the "overlap distance."

Another method, which is similar in spirit but allows more flexibility, is to count how many reconstructed nodes are *near* the original path. This is calculated by finding the union of distance-*k* neighborhoods around the nodes of the original path, and computing the fraction of the reconstructed path which lies in this union. This measure is closer in spirit to the aim of the project, which was to get a sense of where people have traveled. We call this measure the "*k*-bubble quality" of a path.

The image to the right shows the same example path (in orange) along with the union of distance-3 neighborhoods, or "*3*-bubbles", around the vertices it passes (in green). We evaluated paths using *k*-bubble quality and overlap quality, which is essentially *0*-bubble quality.

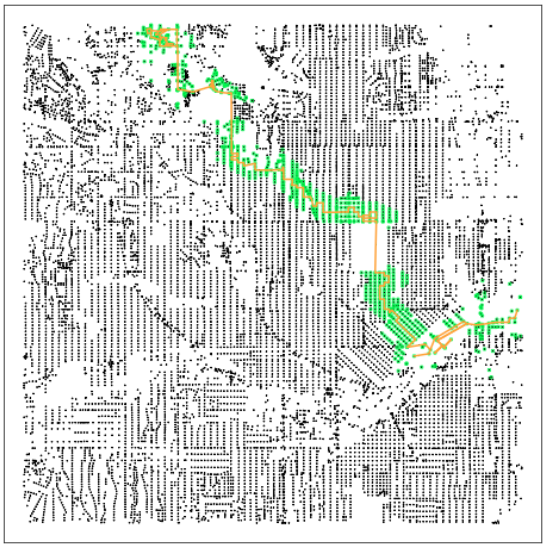
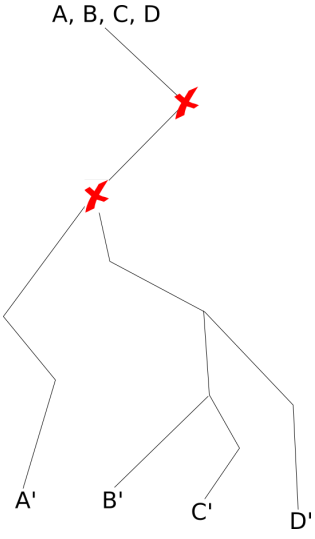


Figure 2-2: The *3*-bubble area (green) for a path (orange)

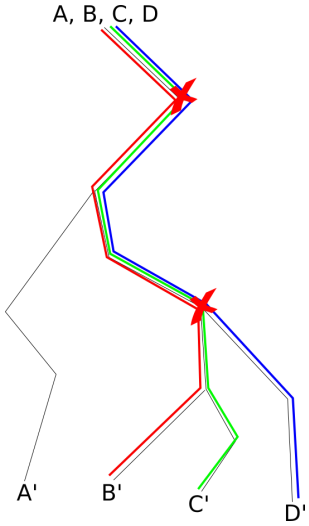
2.4 Placing Trackers

The goal in placing trackers is to maximize the ability -- of the collection of them as a whole -- to reconstruct paths as described above. Because the success of any such arrangement depends in large part on the set of trips it attempts to rebuild, we decided to use our simulated trips as the primary information affecting the choice of nodes at which to locate the trackers.

Intuitively, nodes that are crossed by many paths are good locations for trackers, if the goal is to maximize the number of paths successfully reconstructed by a collection of trackers. But if one naively chooses all of the most frequently crossed nodes, the set of paths that can successfully be reconstructed from the trackers as a collection might suffer.



In the examples to the left and right, there are four trips that briefly cross paths. On the left, the two red “X”s are trackers located at two nodes with the highest frequencies. All of the paths cross both trackers, but the reconstructed paths will each only contain two nodes in common with any of the original paths. Here, if our threshold for “successful reconstruction” is just 50% overlap distance, none of the reconstructions will be successful.



If the bottom tracker is instead placed as in the image to the right, further from the original tracker, it is placed on fewer paths, but the reconstructions it provides are all sufficient to pass a 50% overlap success criterion. So, even though the second tracker is not located at the second most frequently passed node, together this set of trackers provides a better reconstruction rate than had we just chosen the most frequently passed nodes.

To find a balance between these competing interests of high-frequency trackers and low redundancy, we chose trackers in greedy fashion by prioritizing frequency but restricting choices by distance to existing trackers. The selection of trackers was done in waves. For each collection of trackers there was a fixed separation distance S. At the beginning of a wave, the node with the highest frequency that was not already a tracker was chosen. Next, we chose the node with the highest frequency that was also not within S of any tracker from the current wave.

A wave ends when every node in the graph is no further than S away from some tracker in that wave. When this happens, we reset the constraint that new trackers must be far from existing ones, and choose the first node of that wave according to frequency, and repeat the process until we have the desired number of trackers.

We created a collection of trackers of sizes $N \in \{200, 250, ..., 950, 1000\}$ for each separation $S \in \{0, 1, ..., 13\}$. The number of trackers varies because resources for making trackers is not strictly a function of the road network, and for different numbers of trackers a different separation might be better,

so using varied datasets like this allowed us to explore how different combinations affect predictive effectiveness.

3. Results

As an initial remark, due to a miscommunication, the elimination of edges at non-intersection nodes ended up eliminating 11 more nodes than intended. These nodes are ones with degree 2, but with in-degree not equal to out-degree. This very slightly changed the topology of the network. However, it only increased connectivity rather than lowering it. Because of this, and the fact that the data are simulated anyway, these 11 nodes are not cause for concern.

DIRECTED WANDERING PATHS: The paths generated are split into a Training set of 6161 paths, which were used to calculate the node frequencies used to select tracker locations, and a Validation set of 1233 paths, which were used to check the efficacy of the tracker placement.

TRACKER GROUPS: Trackers, computed from node frequencies and graph connectivity, are grouped by the separation value $S \in \{0, ..., 13\}$ and the number of trackers $N \in \{200, 250, ..., 1000\}$ used to compute them. For each S, N pair a distinct collection of trackers was produced.

RECONSTRUCTED PATHS: For each collection of trackers and each validation path, we determined which trackers were crossed by the path and used these to rebuild paths. We computed the *3*-bubble quality and overlap quality of all paths predicted by each set of trackers and for each combination of S and N.

Consider a validation path w and the attempt p at reconstructing it from trackers. We calculate the overlap quality of p as a reconstruction as the fraction of p's nodes that are in w, denoted by $\omega(p)$. If $\omega(p)$ is greater than a given threshold, we will say p is a path *qualified by overlap*.

Similarly, we calculate the *3*-bubble quality of p as the fraction of p's nodes that are within a graph distance of 3 from any node in w, and denote this $\omega_3(p)$. If $\omega_3(p)$ is greater than the given threshold, we say p is *3-bubble qualified.*

Figure 3-1, 3-2, and 3-3 show the relationship between separation, number of trackers, and rate of overlap-qualified and *3*-bubble-qualified paths.

3.1 Overlap *k*-Bubble Quality Percentages

Below are plots of the percentage of paths that were qualified (defined above) for different thresholds. In Figure 3-1, the y-value of each point in the line plot represents the average percentage of qualified paths for all sizes of collections of trackers made with the x-value separation criterion. The lines of differing colors show the qualified/success rate for varying thresholds, and indicate that separations of two or three generally dominate other separations in quality.

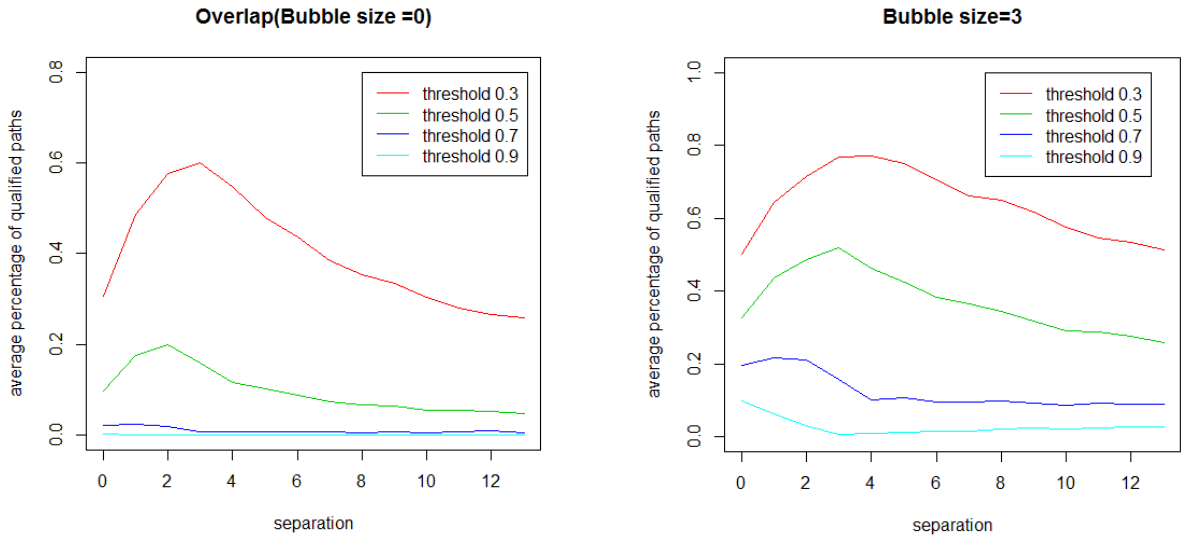


Figure 3-1. The percentage of qualified paths under each metric
(a) overlap quality (b) *3*-Bubble quality

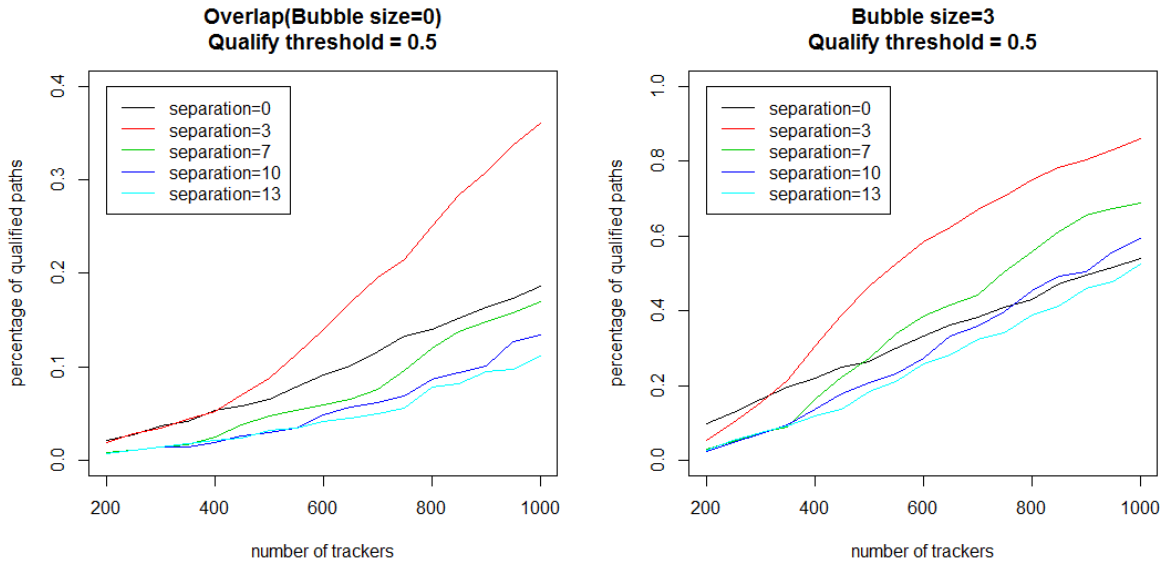


Figure 3-2. The percentage of qualified paths under different number of trackers
(a) overlap quality (b) *3*-Bubble quality

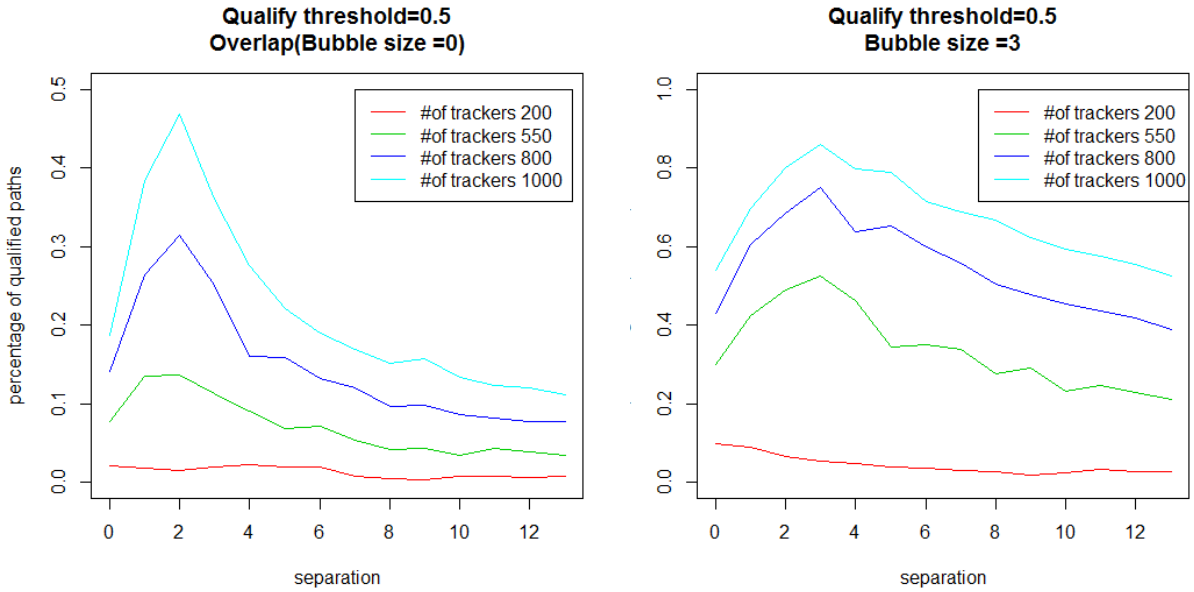


Figure 3-3. The percentage of qualified paths under different separations
(a) overlap quality (b) 3-Bubble quality

In Figure 3-2, each point represents the average percentage of qualified paths for all separation values made with collections of trackers of a fixed size (recorded in the x-value). The main thing to take away from the plots in this figure is that among sets of trackers chosen with nonzero separation, the order of relative success of different separation values is consistent (and even emphasized) as the number of trackers increases.

Figure 3-3 re-emphasizes the fact that performance of a set of trackers increases essentially linearly with number of trackers (once the number of trackers is large enough to fill empty regions), and is unimodal with respect to separation (the mode being around a separation of 3).

Something else that is apparent in Figure 3-2 is that sets of trackers with separation 0, i.e. not using separation, are outperformed by sets of trackers with nonzero separation except for when the number of trackers is relatively small. When the number of trackers is large enough, the relative performance of ignoring separation falls off.

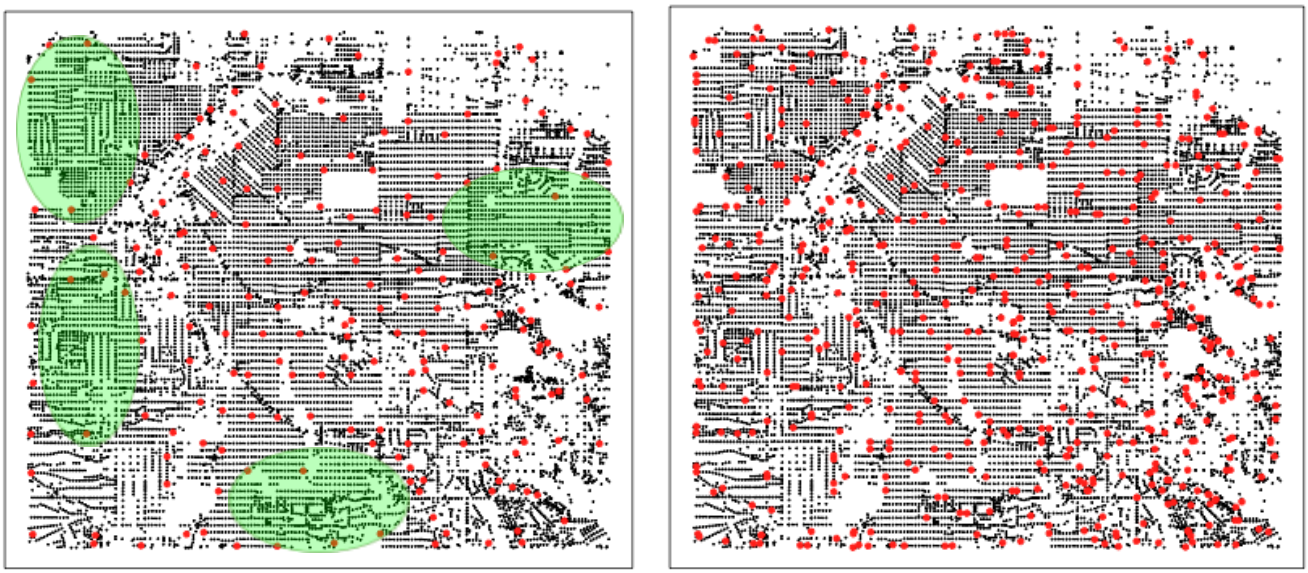


Figure 4. Two tracker sets made with separation 7, size 200 (left) and 600 (right), with some relatively empty regions marked in green

It should be noted that when the number of trackers is small (< 500), there are not enough trackers to adequately distribute them across all regions of the map. This can be seen in Figure 4, which shows separation-7 trackers for a group of 200 trackers and for a group of 600 trackers.

This suggests that using *k*-separation as a strategy has the strongest effect when there are enough trackers available to cover the road network with neighborhoods of size *k* that do not contain each others' centers.

4. Conclusions

The main goals of the project were to find a good way to simulate traffic data from road network structure, and also a good way to use such data to place trackers. The sequence of attempts that ultimately led us to create directed wandering paths were instructive for the first question. We demonstrate that one can create random, yet structured, variations in paths that connect two specific points in a road network.

Our analysis of the placement of our trackers suggests that using the frequency with which paths are crossed, in addition to avoiding placing trackers too close to each other unless it is necessary, is a strong initial strategy for placing trackers. The data suggests that a separation of 3 is optimal for this strategy, because both the overlap and *3*-bubble quality measures of success have maximal means and medians here, across different numbers of trackers (ignoring thresholds). We demonstrate that artificial data can be used to reveal some clear trends in the effectiveness of algorithms and data processing. In particular, this has implications for privacy, in that even less information is needed than one might guess to enable a breach of privacy such as deducing travel patterns.

5. Future Directions

Given more time, there are many related questions and directions that would be interesting to pursue. When reconstructing paths from a collection of consecutive trackers, it would be interesting to use different measures of the most likely connecting path. For example, one could use a weighted ratio of the sum of edge probabilities along a path to the sum of edge weights (distances) along that path. This would make routes that are very commonly used more likely to be guessed as reconstructions, even when they are technically longer paths. This is a very frequent occurrence in real life; there are often shorter routes that people do not take for reasons including a large number of turns (which slow down total travel time and are annoying to deal with), more stop signs, lower speed limits, or other factors.

We would also like to create directed wandering paths using the vector from the current node to the destination instead of the start node to the destination, because this more clearly relates to being guided by a sense of direction and knowing where the destination is.

It would also be interesting to use population density data and information about how locations are districted (that is, whether nodes are in residential areas or commercial areas, or how many residences are within a given distance of a node) to determine delineations of “neighborhoods”. This would make the simulated directed wandering paths more realistic in the sense that they would connect start points and endpoints that people are actually likely to travel between. The current implementation could potentially have starting points in the middle of roads where nobody ever actually starts or ends a trip.

In the same vein, it would be interesting to use edge labels (such as highway vs. residential vs. secondary, etc.) to extract information about nearby nodes, in similar fashion to what was suggested above. This is a particularly interesting avenue because the data is already available in the formatted data we worked with.

6. Contribution summary

All group members contributed in discussions about how to proceed and what goals needed to be satisfied next. All group members also contributed to preparing the presentation.

Writing code was the most divided task: Hao Chen reduced the map data in multiple ways and used the Google Maps API to visualize some of the simulated data; Jingwei Chen produced the initial edge probabilities and both the long random walk in the graph and the directed wandering paths. Jian Qiu made the divisions of the reduced map into neighborhoods and the code to reconstruct paths from lists of tracers hit. Kirk Boyer wrote the code for finding trackers based on the simulated data and for evaluating the quality of reconstructed paths using bubble *k*-quality and overlap quality.

As for data analysis as well as conclusion drawing, Kirk and Jingwei put in more effort.

7. References

[1] R. Dewri, P. Annadata, W. Eltarjaman, R. Thurimella, Inferring trip destinations from driving habits data, Proc. of the 12th ACM Workshop on Privacy in the Electron. Soc., November 4, 2013.

[2] M. Prado, “All electronic tolls make Monday hike go almost unnoticed at the Golden Gate Bridge”, Marin Independent Journal, San Rafael, CA, April 7, 2014.

[3] T. Huang, et al. "Automatic symbolic traffic scene analysis using belief networks." *AAAI*. Vol. 94. 1994.

[4] S. Ozbay and E. Ercelebi, 'Automatic Vehicle Identification by Plate Recognition', *World Academy of Science, Engineering and Technology*, vol. 41, no. 9, pp. 222-225, 2015.

[5] L. Guillaume, 'Road Traffic Data: Collection Methods and Applications', Working Papers on Energy, Transport and Climate Change, Seville, 2008.

[6] C. Patel, D. Shah and A. Patel, 'Automatic Number Plate Recognition System (ANPR): A Survey', *International Journal of Computer Applications*, vol. 69, no. 9, pp. 21-33, 2013.