

Route Prediction from Trip Observations

Jon Froehlich

University of Washington

John Krumm

Microsoft Research

Copyright © 2008 SAE International

ABSTRACT

This paper develops and tests algorithms for predicting the end-to-end route of a vehicle based on GPS observations of the vehicle's past trips. We show that a large portion of a typical driver's trips are repeated. Our algorithms exploit this fact for prediction by matching the first part of a driver's current trip with one of the set of previously observed trips. Rather than predicting upcoming road segments, our focus is on making long term predictions of the route. We evaluate our algorithms using a large corpus of real world GPS driving data acquired from observing over 250 drivers for an average of 15.1 days per subject. Our results show how often and how accurately we can predict a driver's route as a function of the distance already driven.

INTRODUCTION

Route prediction is the missing piece in several proposed ideas for intelligent vehicles. In this paper, we present algorithms that predict a vehicle's entire route as it is driven. Such predictions are useful for giving the driver warnings about upcoming traffic hazards or information about upcoming points of interest, including advertising. One of the most innovative applications of end-to-end route prediction is for improving the efficiency of hybrid vehicles. Given knowledge of future changes in elevation and speed, a hybrid control system can optimize the vehicle's charge/discharge schedule. For example, if a hybrid vehicle knows about an upcoming opportunity to recharge its batteries from regenerative braking (e.g., stop-and-go traffic, sharp curves, or a steep hill), it can use up part of its battery power prior to the opportunity to make room for the expected incoming charge. Researchers from Nissan showed that it is possible to improve hybrid fuel economy by up to 7.8% if the route is known in advance [1]. Tate and Boyd also explore the optimal control scheme for a hybrid assuming the route is already known [2].

While the driver could be asked for his or her route before every drive, we suspect that most drivers would tire of this quickly. This is especially true for a driver's

regular routes, which is where we concentrate our efforts. We found that, for drivers observed for at least 40 days, nearly 60% of their trips were duplicated in our observations. Our prediction algorithms look at a GPS trace of a driver's current trip and attempt to find the best match to a previously driven trip. We find that, in some cases, we can predict a driver's route with 100% accuracy within the first two miles of the trip. Our accuracy is lower in other cases, and our results give details on how often our algorithm achieves various levels of prediction accuracy.

We trained and tested our algorithms on GPS data from 252 drivers. The next few sections describe how we cleaned our typically noisy GPS data, extracted distinct trips, and found drivers' regular routes. We then go on to describe two algorithms for route prediction and give details on how well they perform. First, we highlight some related work.

Route prediction for smart vehicles was addressed by Karbassi and Barth [3] for a car-sharing application. Their task was to predict which route a driver would take between given starting and ending drop-off stations. In our work, we do not rely on the driver to enter his/her destination. Torkkola *et al.* [4] learn destinations and routes from GPS data. As in our work, these learned routes are the basis for prediction, although their prediction algorithm is not given. Using a hidden Markov model learned from 46 sampled trips, Simmons *et al.* [5] predict destinations and routes based on knowledge of the road network. They quote an accuracy of predicting the next road segment as high as 99%, although in 95% of the cases the next road segment is the only one connected to the current one. Their rich model allows the incorporation of time-of-day, day-of-week, and speed sensitivity into their predictions. Their results show that only speed is a significant help in boosting their prediction accuracy. Patterson *et al.* [6] applied machine learning and a particle filter to people's GPS traces to predict their destination, route, and even mode of transportation from an inferred list of previous destinations. Our approach also differs from the short-term route prediction in [7]. In their work, the goal was to

predict ahead just a few blocks based on the most recent road segments. In this paper, we make long-term predictions about the entire route.

The research we present in this paper is primarily differentiated from previous work by our novel process for extracting routes from GPS data, our analysis of route regularity, our prediction algorithms, and our large corpus of test data from 252 separate drivers. We also note that our work is different from GPS-based learning for finding people's frequent destinations (e.g., [8]) or for predicting a future destination (e.g., [9]). The goal of our work is to accurately predict a driver's entire route very early in a trip.

FROM GPS DATA TO REGULAR ROUTES

Our route predictions are based on observations of individual drivers with GPS. This section explains how we gathered raw GPS traces, extracted discrete trips, and found regular routes.

THE GPS DATA (MULTIPERSON LOCATION SURVEY)

The GPS traces used in our analysis were collected as part of the Microsoft Multiperson Location Survey (MSMLS). The MSMLS is an ongoing study at Microsoft Research aimed at gathering driving data for location-based research. Since June 2005, over 2.2 million GPS location points have been collected from 252 subjects, who volunteer to drive with a GPS recorder in their vehicle for two weeks or more. Subjects are recruited from the Seattle, WA area and nearly all are employees of Microsoft or their family members.

The *Garmin Geko 201* portable GPS receiver is used for data collection. It is capable of recording up to 10,000 time stamped GPS points in internal memory (which includes latitude, longitude and altitude). We used the device's adaptive recording mode, which dynamically modifies the logging frequency based on the device's movement. In this mode, the GPS unit tends to record more points when the vehicle is changing speed or direction and fewer points when it is stopped or traveling in a straight line at constant speed. The median time interval between logged data points in our dataset is six seconds and the median distance is 208 feet.

All participants in the MSMLS are instructed to place their loaned GPS receivers on their vehicle's dashboard or some other area within the vehicle with a clear view of the sky and to drive normally during the survey period. The receivers are powered via a car adapter cable plugged into the vehicle's cigarette lighter socket. We modified the GPS units slightly such that they would automatically turn on whenever power was supplied. This was advantageous because it meant that the subjects did not have to remember to turn their GPS receivers *on* when they started their vehicles and *off* when they stopped. Thus, once the GPS receivers were plugged in, the subjects could forget about them for the

rest of the survey period. Some vehicles supply power to the cigarette lighter socket even when the vehicle is off. The adaptive recording mode, however, decreases the logging frequency when no motion is detected, thereby reducing the accumulation of data when the vehicle is parked (even if the device is still receiving power).

Prior to the data collection period, subjects are asked to fill out a short demographic survey. At the end of the collection period, they return their GPS receivers to the research team. As compensation for their participation, volunteers are entered into a drawing to win a \$250 portable media player.

Currently, the MSMLS database contains 2,222,758 data points collected from 252 subjects, for an average of 8,821 recorded points per person (Figure 1a). On average, we have 15.1 days worth of driving data per subject (Figure 1b). Although the maximum storage capacity of the Geko 201 GPS device is 10,000 data points, recently our subjects have been given the option of continuing their data collection after their device memory initially fills up. In these cases, we have the subjects switch to a new GPS receiver after every two weeks. At present, 41 subjects have more than 10,000 data points (average among these subjects: 19,071). From this raw data, we extracted discrete trips, a process we describe in the next section.

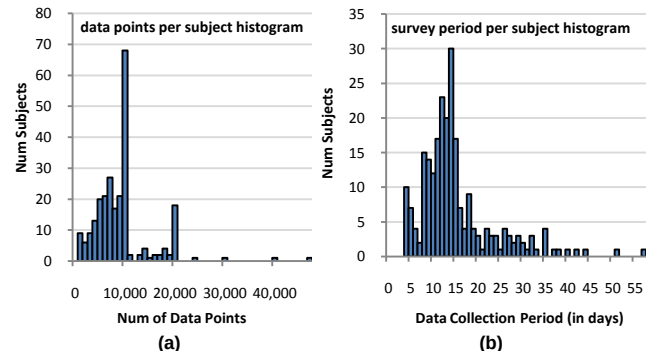


Figure 1. A high level view of the MSMLS dataset: (a) shows a histogram of the number of data points per subject and (b) a histogram of the number of days worth of driving data per subject.

FROM GPS DATA TO TRIPS

Our route prediction algorithm depends on knowledge of a driver's discrete trips, but the GPS data gives no explicit indication of when a trip begins or ends. In addition, GPS points themselves are often noisy and some contain invalid sensor data. This section explains how we go from raw GPS measurements to a plausible set of discrete trips for each driver.

For the purposes of this project, we define a trip simply as a set of temporally ordered, time stamped GPS data points collected by a given subject. Trips are composed of N data points and $N-1$ trip segments (the edges between these points). We created a three stage process to transform the raw GPS data into trips: first,

we segment the trips into multipoint trip objects; second, we clean the trips by removing invalid data points; and third, we filter the trips to eliminate false trip objects. These are described below.

Trip Segmentation

The trip segmentation algorithm sorts each subject's raw GPS data chronologically and looks for gaps between two consecutive recorded points (P_1 , P_2) of three minutes or more. If a gap is found, P_1 becomes the end point of the last trip and P_2 the beginning point of the current trip. The threshold value of three minutes was chosen empirically based on the logging interval distribution (Figure 2a) and experimentation with map visualizations of the data. Although primitive, this algorithm takes advantage of the adaptive recording mode used by our GPS receivers, which limits the number of recorded points when no motion is detected. In addition, such a gap also exists when a vehicle is parked and turned off for more than three minutes thereby signifying the end of a trip (assuming the vehicle *does not* supply power to the GPS device when off).

Selecting a reasonable threshold value here is challenging because it is difficult to distinguish between brief stopovers that are legitimate ends to a trip (e.g., stopping at a convenience store) and normal traffic stoppage (e.g., stopping at a long traffic light). With our dataset, it is not possible to tell the difference between a three minute gap that resulted from a car being shut off and a three minute gap that resulted from a long traffic light. Setting the threshold to three minutes resulted in 27,610 trips.

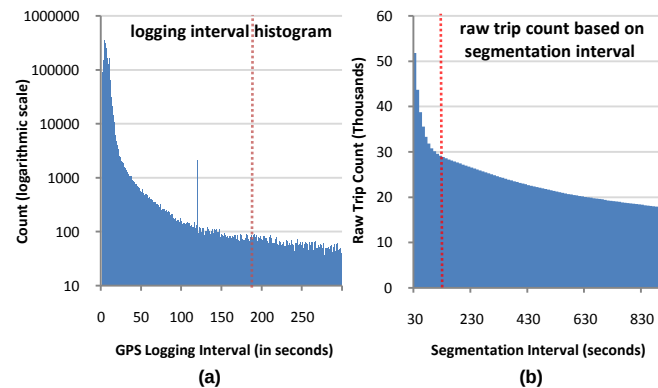


Figure 2. (a) A three minute gap (red line) between logged data points was used to segment the raw data into trips. (b) The number of raw trips changes depending on the segmentation interval. Here, the red line indicates the segmentation interval used in our dataset.

Trip Cleansing

Once the trips were segmented, the data points within those trips needed to be cleaned, because our GPS data contained outliers. Although the Geko 201 GPS receiver utilizes the Wide Area Augmentation System (WAAS) to assist in location accuracy, outlier points from obscured line of sight, device cold starts and other satellite

disruption phenomena can result in outlier GPS readings (see Figure 3). Outliers typically occur in isolation and appear as a wildly mistaken point along an otherwise reasonable sequence of GPS points. Since the outlier is normally far away from the temporally adjacent points in the trip, they stand out when we compute local speed and acceleration from adjacent points.

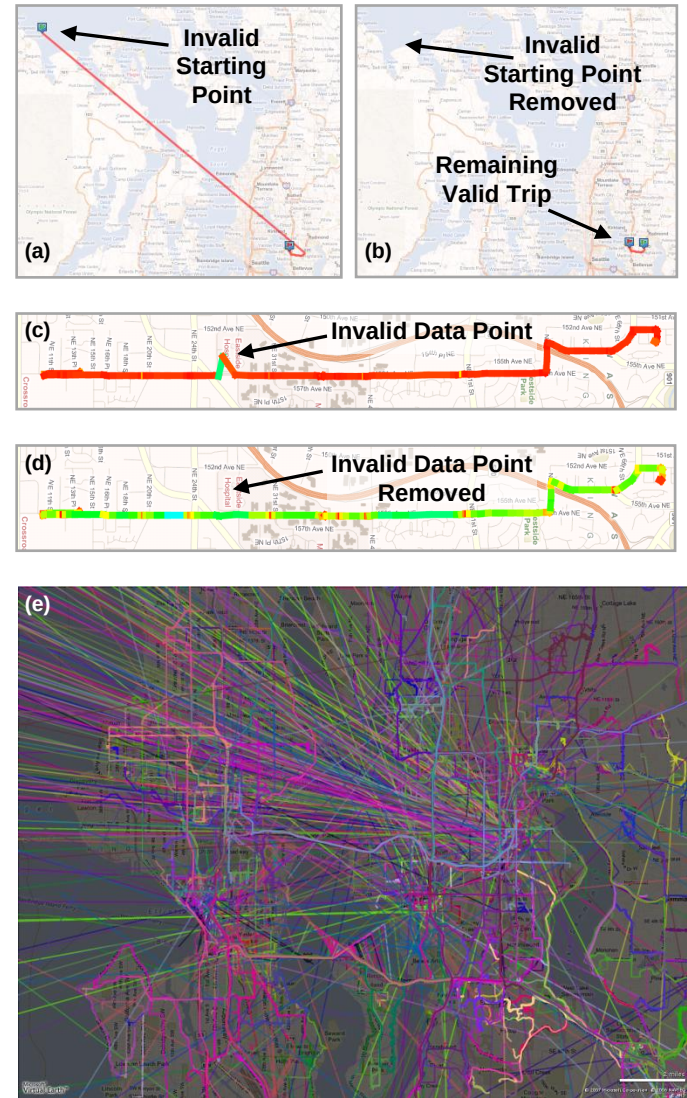


Figure 3. The above figures demonstrate example outliers in our GPS data. (a) The invalid starting point of this trip created a trip segment of a speed exceeding 97,500 mph. (b) This same trip once the invalid data point was removed. (c) An invalid data point in the middle of a trip resulted in an anomalous 397.8 mph trip segment. (d) The same trip after cleansing. (e) A map of the Seattle metropolitan area with a subset of the (uncleansed) trips in our dataset. The trips are colored by subject. Note the spurious straight lines in this figure; these are caused by outliers. This is the same map view as that found in Figure 7b but with raw, uncleaned trips.

We created two trip cleansing algorithms based on speed and acceleration to remove the offending points. Data points that contributed to a speed of over 100 mph or an acceleration of over 80,000 mph^2 were eliminated. We iterated over every trip segment in each trip looking

for outliers. If a trip segment exceeded our speed or acceleration thresholds both data points were tested for validity. If by removing one of the data points, the irregularity subsided then we had found our invalid point and it was removed (e.g., Figure 3c and 3d). If not, then both data points were removed. The trip was then reformed into new trip segments and the cleansing algorithm started over until all invalid points were eliminated. This reduced the number of data points in our trips from 2,222,758 to 2,196,828 (a reduction of 1.2%).

Trip Filtering

After cleansing, the next processing stage was trip filtering. Vehicles that continued to power their GPS receivers even when turned off would often produce intermittent streams of GPS readings. Satellite location sensing drift would fool the adaptive recording mode into thinking that the GPS device had begun moving even though the vehicle was parked (Figure 4a). These faulty readings could continue for as long as 10-15 minutes at a time, though they tended to come in shorter spurts. Our trip segmentation algorithm did not discriminate between these *fictitious* trips and the *real* trips in our dataset. Thus, we created a set of trip filtering algorithms that work in succession to identify and eliminate these false trips. Trip filtering, however rudimentary, is an essential part of our work. Before running experiments with our route prediction algorithms we had to ensure that a vast majority of our trip data was valid.

MinimumPointCountTripFilter: The minimum point count trip filter removed trips with less than a specified point count, which we set to ten data points.

MinimumTripTimeFilter: The minimum trip time filter removed trips that took less time than a supplied threshold value, which we set to 30 seconds.



Figure 4. (a) 34 garbage trips were created by GPS location drift over a three day period while this driver's vehicle was parked outside his house. (b) These garbage trips were transformed to intermediate centroid based trips (shown in orange), which are less than 0.1 miles in length and therefore are filtered out of our dataset.

CentroidBasedDistanceTripFilter: The centroid-based distance trip filter was built to remove dense, highly circular trips—a salient characteristic of fictitious trip data. It works by iterating over a given trip using a sliding window of 10 points long. At each step, a centroid point

is generated from the data currently in the window, essentially making this into a moving average filter on 2D (latitude, longitude) data. These centroid points are then combined into a new trip whose travel distance is computed. If this distance is below 0.1 miles, we eliminated the original trip. An example of this processing is shown in Figure 4.

DirectionChangeFrequencyTripFilter: The direction change frequency trip filter removed trips with a high degree of directionality change. The algorithm works by counting the number of four-way¹ and eight-way² direction changes between each trip segment in a trip. If the ratio of direction changes to trip segments is greater than 0.5 for four-way directions or 0.6 for eight-way directions, the trip is removed. The invalid trip shown in Figure 5a shares many characteristics with a legitimate trip; it has a reasonable length (1.7 miles) and average speed (19.8 mph)—however, one can clearly tell visually that the trip data is erroneous. The direction change frequency trip filter properly removes this trip as its direction change ratios of 0.64 (four-way) and 0.66 (eight-way) exceed our set thresholds.



Figure 5. (a) The direction change frequency trip filter correctly filters this erroneous trip (in red). (b) The trip would not have been eliminated via the centroid based trip filter (shown in orange). (c) Even a very “turny” legitimate trip like that in (c) is not erroneously filtered, which has a direction change ratio of 0.19 and 0.33.

Finally, we used the following two filters to ensure that the remaining trips took place within Washington State and were *complete* trips.

WithinBoundsTripFilter: The within bounds trip filter removed trips that took place outside of a specified geographical bounding box, which was set to the approximate boundary of Washington State. Some of our subjects took their GPS units with them while traveling and placed them in their rental cars. This filter removed occurrences of those trips.

MostRecentTripFilter: The most recent trip filter removed the subject's last trip if their GPS receiver's storage memory was full when it was returned to the research team. Removing this trip was necessary because we could not guarantee that the last trip in memory was a complete trip (i.e., that it did not get cut off when the internal memory ran out).

¹ Cardinal directions: North, South, East West

² Cardinal and inter-cardinal directions: North, Northeast, East, Southeast, South, Southwest, West, Northwest

After filtering, our corpus of trips appeared much more reasonable, although some invalid trips remained, and some valid trips were removed. Acknowledging that our filtering process was not perfect, we tried to err on the side of removing too many valid trips with the goal of minimizing invalid trips that would distort our prediction accuracy experiments.

THE TRIP DATA

Trip filtering reduced the number of trips from 27,479 to 14,523 (a 47.2% reduction). For our analysis, we also required that subjects have more than ten total trips; this excluded twelve subjects. We felt that ten or fewer trips were not enough to properly establish a driver's regular patterns. Thus, our final dataset included 14,468 trips from 240 subjects. On average, our subjects took 4 trips per day. The average trip length was 7.7 miles and trip time 16.3 minutes. For comparison, according to the 2001 National Household Travel Survey, an average driver in a metropolitan area the size of Seattle makes 4.07 trips per day with an average length of 9.87 miles [10]. Figure 6 reveals some high level statistics and distributions of our trip data and Figure 7 displays the trip data overlaid on two maps, one of Washington State and the other of Seattle.

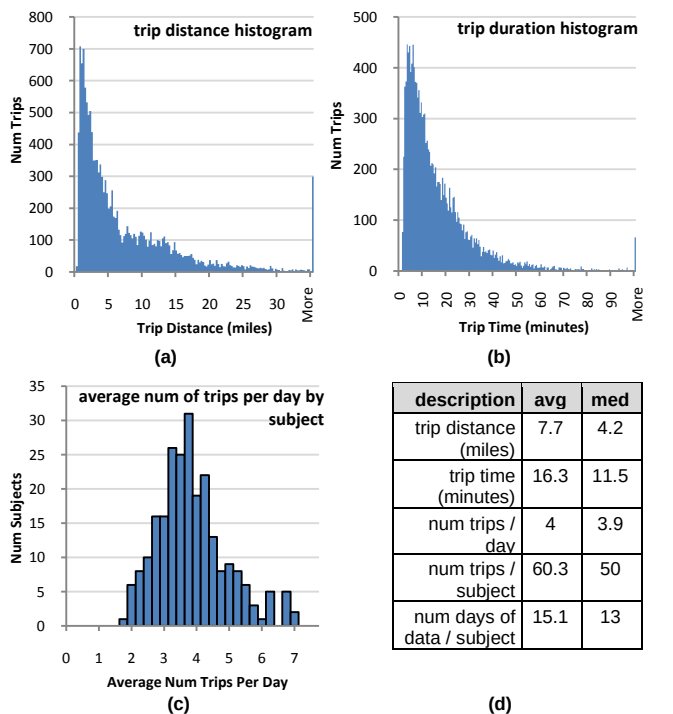


Figure 6. A histogram of trip distances is shown in (a) and trip durations in (b). Most subjects drove between 3-5 trips per day on average as can be observed in (c). Finally, (d) shows a small table of trip statistics.

FROM TRIPS TO REGULAR ROUTES

When making a route prediction, we try to pick which previously seen route a driver is on. We make a distinction between trips, which we extract from the GPS data, and routes, which we build from repeated trips. A trip describes a driver's path through time and space

using time stamped GPS data. A route is simply an abstraction of a trip (or trips) without the temporal component. That is, a route is a collection of latitude, longitude pairs that define a directed path. A *regular* route is a path that a driver drives often. This section describes how we extract routes from trip data.

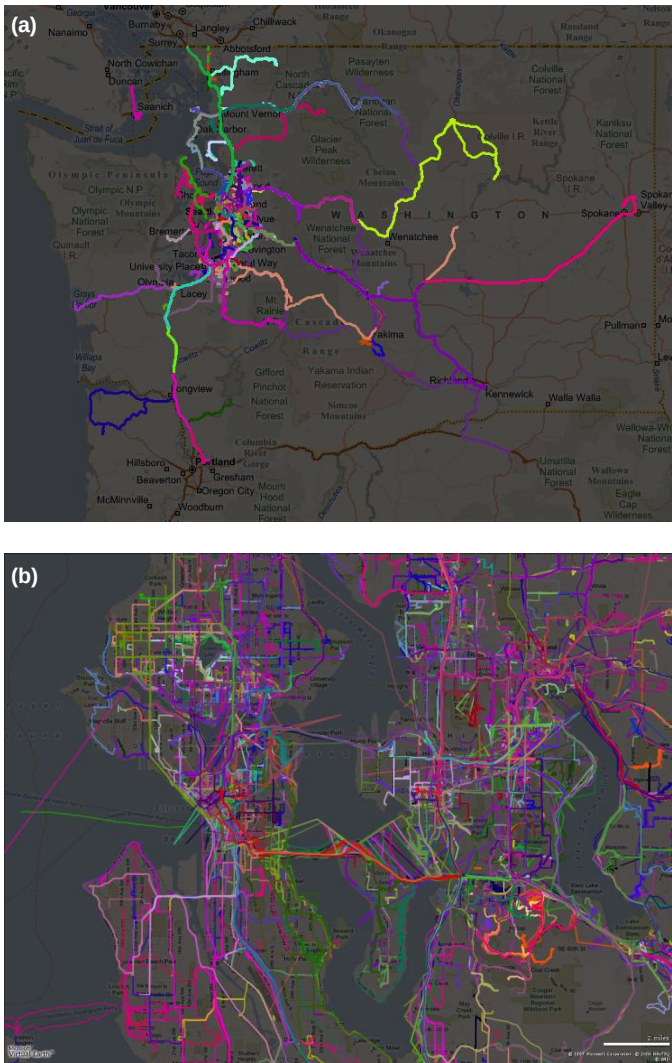


Figure 7. A map of Washington (a) and the Seattle metropolitan area (b) with overlaid trip data from the MSMLS dataset. The trips are colored by subject. Figure (b) is the same map view as Figure 3 but with cleaned and filtered trips. Note that the maps have been darkened to emphasize the trip data.

Trip Similarity

Intuitively, trips that a driver takes along the same path should be considered the same route. To do this programmatically, we must formally define some notion of trip similarity. If two or more trips are similar, then we combine them into a route. This section shows how we assess the similarity of two trips. The algorithm relies solely on the (latitude, longitude) data from our filtered trips. It does not require an extra step of inferring which roads were traversed.

At a high level, the algorithm works by computing the average minimum point-segment distance between two trips (e.g., from Trip A to Trip B and from Trip B to Trip A). These averages are added together and divided by two to calculate a symmetric score representing the similarity between the two trips.

In particular, our similarity algorithm works as follows:

1. For every point P_{Ai} in Trip A, find the closest trip segment TS_{Bj} in Trip B (Figure 8a). A trip segment is simply a straight line between two temporally adjacent GPS points. Calculate the minimum point-segment distance between P_{Ai} and TS_{Bj} (Figure 8b).
2. Add together these point-segment distances to compute the total distance between Trip A and Trip B: $TotalDistance_{AB}$.
3. Calculate the similarity score ($Score_{AB}$) by dividing $TotalDistance_{AB}$ by the number of data points in Trip A. This score is asymmetric; it represents the similarity from Trip A to Trip B but not from Trip B to Trip A (Figure 8c).
4. Repeat the above steps but this time compare Trip B to Trip A. This produces a 2nd asymmetric score: $Score_{BA}$ (Figure 8d).
5. Add both scores together and divide by two to calculate the final similarity score. Set $Score_{AB}$ and $Score_{BA}$ equal to this final score thereby enforcing symmetry. A lower score indicates more similarity.
6. Finally, store the results in a similarity matrix.

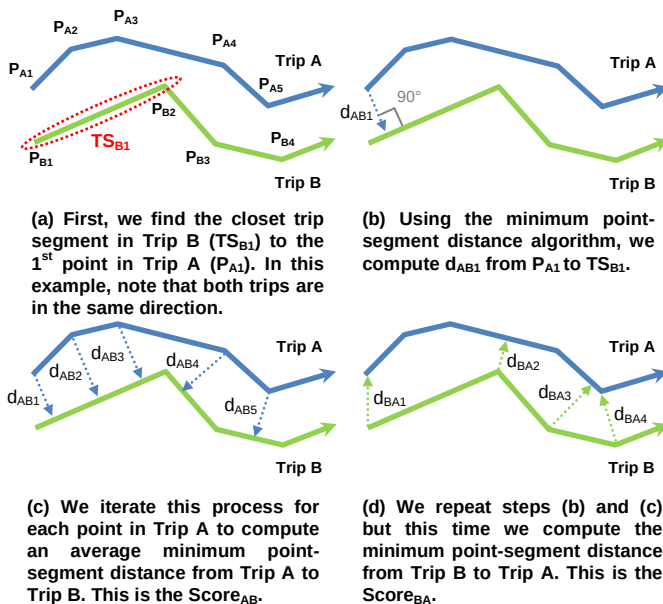


Figure 8. The four figures above walk through some high level aspects of our trip similarity algorithm.

There is one specific nuance in the algorithm which allows us to differentiate the directionality of two trips along the same path: the comparison between points and segments is an *ordered* comparison. As we iterate through a trip's data points looking for its comparison trip's closest segment, the closest segment found must always have an index equal to or greater than the last closest segment found (as illustrated in Figure 9). Two trips along the same path but in opposite directions will then correctly be identified as having unique routes.

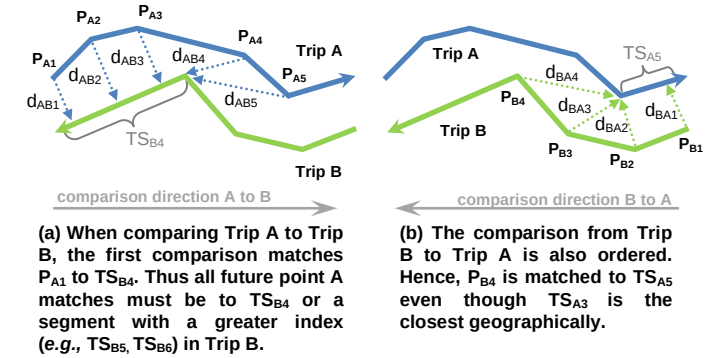


Figure 9. Even though Trip A and Trip B follow roughly the same path, they are in opposite directions. The ordered comparison ensures that their similarity score is comparatively large.

Our similarity measure is a version of the Hausdorff distance algorithm, sometimes used in computer vision to compare object outlines [11]. Torkkola *et al.* [4] use a purely point-based algorithm for comparing two trips. Their similarity measure looks at the set of closest matches between the points in each of the two trips. If the maximum of the minimum distances exceeds a threshold, the trips are declared dissimilar. Our technique compares points in one trip to line segments in the other trip, which helps account for variations in GPS sampling rate. Also, our similarity measure is sensitive to the direction of travel.

Detecting Routes through Trip Similarity Clustering

In order to create routes from trip data, every trip in a subject's dataset must be compared (this takes n^2 comparisons where n is the number of trips). We use the results of these comparisons to construct a trip similarity matrix, which holds all of the similarity scores between trips for a given subject. The scores in the trip similarity matrix are then used to cluster similar trips together. Finally, these clusters are converted to routes. The precise definition of this cluster-to-route transformation is covered in the next section. Here, we focus on route detection through trip similarity clustering.

To detect routes, we repeatedly combine trips with the lowest scores (highest similarity) in our similarity matrix into trip clusters. We apply *dendrogram clustering*, which is a hierarchical clustering technique that recursively clusters data points (Figure 10f), to repeatedly combine trips until the lowest score in the similarity matrix exceeds a set threshold. For our analysis, this threshold

was set to 0.05 miles (264 feet)—meaning that, on average, two trips must be within 0.05 miles apart (using our distance measure) to be clustered together. Each trip cluster is transformed into a route. The size of the cluster represents how frequently that route was traveled. Those trips that have not been clustered are assumed to be unique routes—routes that have only been driven once.

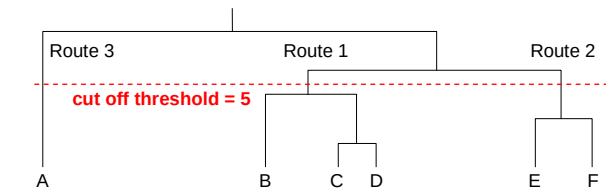
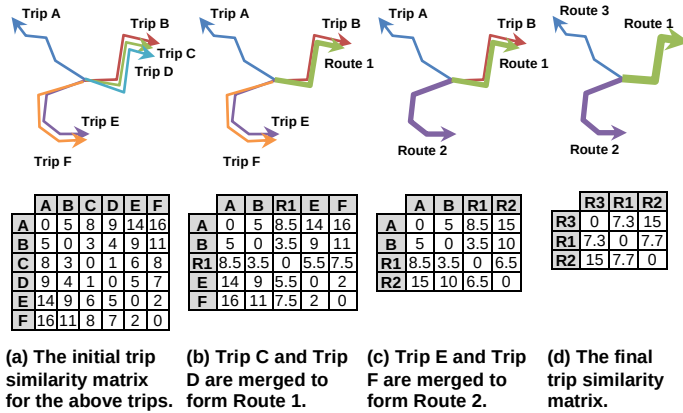


Figure 10. In this example, dendrogram clustering uncovers three routes (d) from six example trips (a). Note how Dendrogram Clustering uses a threshold to constrain the composition of clusters (f).

Figure 10 illustrates how trip clustering is used to discover routes. In this example, a subject has driven six trips over three distinct routes (Figure 10a). For illustrative purposes, our clustering threshold is set to 5. The algorithm begins by constructing an initial 6 x 6 trip similarity matrix. Each row and column in the matrix represents a trip. Note that the matrix is symmetrical; this is an effect of the score symmetry property enforced by our trip similarity algorithm. After the initial matrix has been constructed, the next step is to scan the matrix for the smallest similarity score. In our example, Trip C and Trip D have the lowest score and thus, are the first to be clustered. This produces Route 1. Trip C and Trip D are subsequently removed from the matrix and replaced by Route 1 (Figure 10b). The matrix, which now has a dimension of 5 x 5, is recomputed and scanned. In this case, Trip E and Trip F are found to have the smallest score and are clustered into a second route, Route 2 (Figure 10c). Once again the dimensionality of the matrix is reduced. The next scan reveals that Route 1 and Trip B have the smallest similarity score. To merge a route and a trip, the *original trips* that make up the route are *reclustered* with the new trip. Thus, in our example, trips B and C are reclustered with D to reform Route 1.

Finally, we are left with a 3 x 3 matrix composed of two routes and 1 trip; however, the smallest remaining score in the matrix is greater than the dendrogram clustering threshold so the clustering ceases (Figure 10f).

From Trip Clusters to Routes

Left unsettled in the explanation above is how we combine multiple trips into one representative route. The key objective of this transformation is to maintain the overall path structure that the trip cluster follows. We experimented with three separate trip merging algorithms before selecting one for our analysis:

EvenIntervalMerge: The even interval merge algorithm resamples trips in the trip cluster at even distance intervals and calculates the average (latitude, longitude) point at every interval. These averaged points comprise the route. The sampling rate is determined by the average number of data points per trip in the trip cluster.

ExhaustiveIntervalMerge: The exhaustive interval merge algorithm is similar to the above but the trip data is not resampled at a fixed rate. Instead, every data point in each trip becomes a resampling point. This provides a higher fidelity reproduction of the overall path structure compared to the even interval merge algorithm but at a cost of complexity. This algorithm produces a route that grows exponentially with the number of trips and the number of data points within those trips.

BestRepresentativeMerge: Unlike the previous two merging algorithms, this algorithm does not attempt to explicitly merge trips into an aggregate representation. Instead, it selects an existing trip as the canonical route. The selected trip has the smallest average similarity score to all other trips in the trip cluster. Its path structure (set of latitude, longitude points) is copied directly into the route.

For our route prediction experiments we chose the even interval merge algorithm. Although the exhaustive merge algorithm offers a slightly better reproduction of path structure, its increase in complexity made our clustering computations prohibitively expensive. Note that when merging two routes or a route and a trip, the original trip data that comprises the route is used for merging and not the route's abstract form.

ROUTE PREDICTION

Our expectation is that much of an individual's driving is spent traversing regular routes. If this is true, then we can try to recognize when a driver is on a regular route and thus predict that he or she will stay on that route. In this section we use our trip and route data to show how regular drivers are, and then we develop two route prediction algorithms based on this regularity. We should note that in our dataset we did not track which vehicles had more than one primary driver. We expect that a shared vehicle will take longer to produce regular routes

simply because there will likely be a greater variance in routes driven.

DRIVERS ARE REGULAR

We define a *repeat trip* as any trip that occurs more than once along a route. On average, 39.3% of the trips our drivers take are repeat trips (Figure 11). For 67 of our subjects, this rate was greater than 50%—that is, one out of every two trips for these drivers is along an established route. Five of our subjects, however, never drove a route more than once. These subjects are somewhat anomalous in that we only have an average of 4.6 days worth of driving data from them. For subjects whom we have at least 20 days worth of data ($n=45$), the average increases to 49.8%.

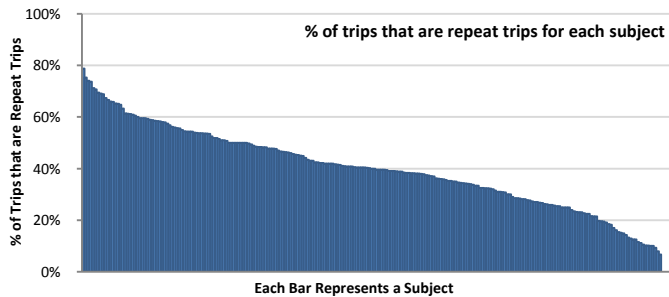


Figure 11. This bar graph shows the percentage of trips that are repeat trips for each subject (average=39.3%, median=39.6%). Each bar represents a subject.

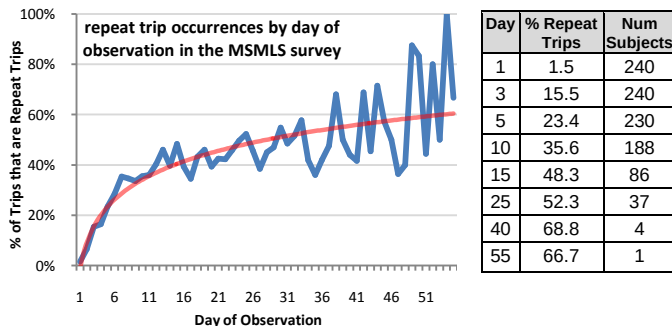


Figure 12. The percentage of repeat trips that a driver takes grows with the number of days of observation. By the 25th day of observation, a majority of trips driven are over routes that have been seen before. The trend line is graphed in red.

In general, we find more repeat trips with more days of observation. In fact, the number of repeat trips approaches 60% for subjects observed 40 days or more, as show in Figure 12. This repeat trip growth rate is relevant because our proposed system predicts a driver's route based on the driver's trip history. A slow growth rate for repeat trips would mean that the system would have to monitor the driver for a longer period of time before making accurate predictions. On the first day of observation (the day when our subjects received their GPS recorders), 1.5% of the trips were determined to be repeat trips. By day three, this percentage jumps to 15%. By the first full week (7 day period), over 35% of a driver's trips are repeat trips.

The percentage of unseen routes appears to decay exponentially with the number of days of observation before leveling off around the 30th day at 50-60%. Clearly, the number of new routes that a driver may take is unbounded. However, the core set of routes a driver takes appears to be discoverable within a month. Note that the number of subjects contributing data drops as the days of observation increases (see table next to Figure 12). As a result, the y-axis value becomes more variable.

Finally, one additional way to look at route regularity is by the distribution of a driver's trips across routes. A highly regular driver will have a large portion of his/her trips distributed over only a few routes. That is, a small amount of routes will account for a large percentage of this driver's trips. In our data, the route traveled most frequently accounts for an average of 12% of a driver's total trips. This is typically the morning commute route. The top ten most frequently traveled routes account for nearly 50% of a driver's trips. Figure 13 shows an averaged cumulative distribution of trips to routes for each subject. To build this graph, we ordered each subject's set of routes by the frequency of travel—that is, the first route for each subject is the route comprising the most trips, the 2nd route is the route comprising the 2nd most trips, and so on. We then averaged the number of trips in each of these routes across subjects to determine the average distribution of trips in routes in our dataset. Thus, Figure 13 shows that, on average, the most popular route makes up 12% of a driver's trips and the top 10 most popular routes make up 50% of a driver's trips. Note that this graph includes routes that were only driven once—hence the graph correctly progresses towards 100% (where 100% of the trips make up 100% of the routes).

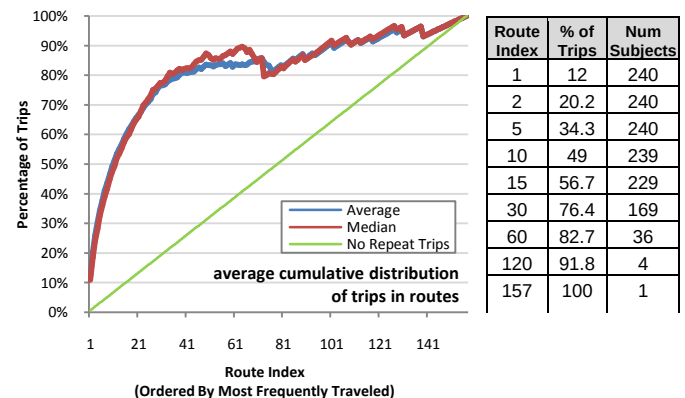


Figure 13. This graph reveals that a small amount of routes comprise a majority of trips driven. It takes roughly 10 routes to account for 50% of a driver's trips. For comparison, the thin green diagonal line depicts a trip to route ratio of 1:1. In other words, it graphs the hypothetical case of our dataset having no repeat trips.

ROUTE PREDICTION ALGORITHMS

As a trip progresses, we try to find which previously driven route, if any, the driver is on. Using the trip similarity algorithm described earlier, we calculate the

distance between the current trip and existing routes and store the results in a similarity matrix. This matrix is continuously updated as the trip progresses at pre-specified distance intervals, for example, every quarter mile. The values d_1 and d_2 store the distances between the current (partial) trip and the 1st and 2nd closest routes. These distances obviously change as the trip progresses. In addition, different route objects will likely comprise the 1st and 2nd closest routes over the course of a trip (Figure 14).

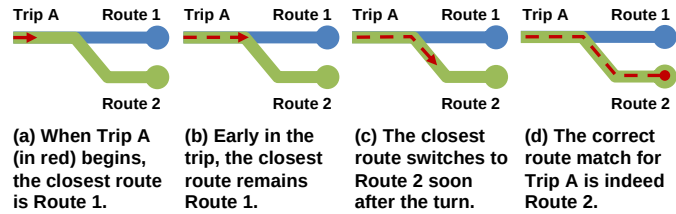


Figure 14. Both of our route prediction algorithms are based on calculating a continuously updated similarity matrix, which tracks the distance from the current trip to the existing routes. Note that before the trip completes, the closest route is *not* always the correct route.

We created two simple algorithms to predict a driver's route based on the above calculations. (1) The *closest match* algorithm always returns an ordered list of the routes most similar to the current trip, and we take the closest route as the predicted route. (2) The *threshold match* algorithm returns a route and a confidence measure given the current trip's travel distance so far and the distances to the 1st and 2nd closest routes (d_1 and d_2). Details of these two algorithms are given in the next section.

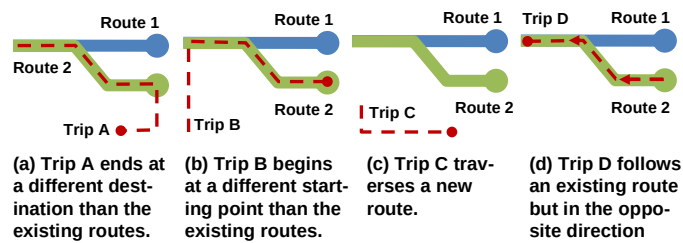


Figure 15. A set of example trips that forge new routes and therefore cannot be correctly matched to existing routes using our algorithms.

Of course, not all routes can be predicted. As our data shows (Figure 12), even after a month of observation 40% of a driver's trips are still over new routes. In these cases, the closest route found is *never* a correct match (as the correct match does not exist). Figure 15 highlights a set of example trips whose routes cannot be predicted using our algorithms. In Figure 15c, for example, the closest route to Trip C is Route 2. However, Trip C clearly does not traverse the same path as Route 2. Figure 15b highlights a different case, which covers an area of future work for us involving partial route matching. Although Trip B begins on a new route, it eventually converges to two pre-existing routes (Routes 1 and 2). At this point, it may be possible to

predict Trip B's remaining path. This type of prediction, however, is beyond the scope of this paper. Instead, we focus on matching routes in their entirety.

ROUTE PREDICTION RESULTS

We tested our two route prediction algorithms on 14,468 trips from 240 subjects. To test our algorithms, we applied a leave-one-out approach, where one test trip was left out of a driver's dataset and the remaining trips were clustered into routes. The test trip was then "virtually driven" in 5% increments and our route prediction algorithms were applied. This process was repeated for every trip for each subject. In the real world, the test trip is analogous to the driver's current trip that s/he is currently driving.

For both algorithms, we present our prediction accuracies for the full set of trips and for just the *repeat* trips. The former is much more challenging because it includes routes that have only been driven once. With the leave-one-out approach, these routes are eliminated while their trips are tested—thus there is no correct route left in the trip clusters to match. Neither of our two algorithms attempt to predict routes that have never been driven in the past and thus perform poorly in these cases as a result.

CLOSEST MATCH PREDICTION RESULTS

The closest match prediction algorithm greedily selects the closest route to the currently driven trip as the *predicted* route. To measure the distance to a candidate route, we use the Hausdorff-like, trip similarity measure described earlier. The closest match algorithm also returns an ordered list of best route match candidates. Figure 16 shows the average location of the correct route in this ordered list as the trip advances. As expected, when the trip first begins and little information is known about its path, the correct route is not necessarily the best match. After 35% of the trip has been driven, the correct route moves to within the top 5 locations in the ordered list; after 50%, the correct route is, on average, within the top 2 matches (Figure 16a).

Note, however, that speaking in terms of "percent of trip completed" is useful in capturing how well the algorithm performed but it is not practical. A vehicle navigation system can never be certain of how far along a trip it is *in terms of percentage complete* without knowing the exact route of the trip from start-to-end—this, of course, is what we are trying to predict. Instead, a much more practical input parameter is the trip's current distance traveled—that is, how far the vehicle has traveled since the trip began. Every vehicle is capable of tracking this value. Thus, Figure 16b more accurately portrays the performance of the closest match prediction algorithm in a real usage scenario. It shows, on average, how highly the correct route is ranked as a function of the trip's current travel distance. For example, by the end of a trip's first mile, the correct route is, on average, within

the top 8 matches and within the top 5 after 5 miles of driving. Note that because this graph is an aggregate view of performance, each point includes trips that are in a range of completion. For instance, at the mile 3 slice, the graph includes data on short trips that are just completing (e.g., trips of 3-4 miles in length) as well as long trips that are really just beginning (e.g., > 20 miles).

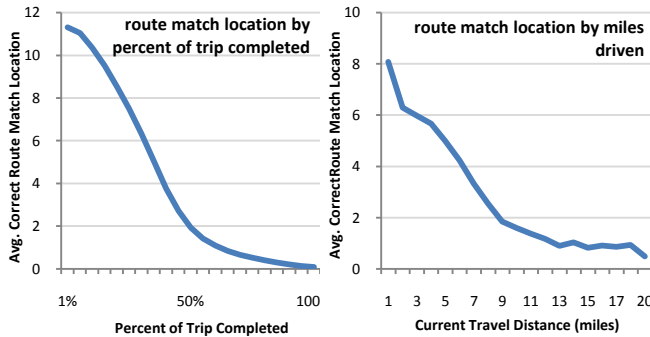


Figure 16. The closest match prediction algorithm returns an ordered list of the best matching route candidates. The average correct route index is graphed above by both percent of trip completed (a) and current travel distance (b). At the halfway point of a trip's completion, the correct route is within the top 2 matches.

Obviously the goal here is to distinguish the correct route as early in a trip's lifetime as possible. We believe the primary reason why it takes so long to select the correct route is because there is a good deal of overlap between a driver's most frequently traveled routes. Clearly there is a small set of paths that extend from a driver's home or workplace—these paths only become distinct after a certain amount of divergence. Thus, early in a trip's lifetime there are many reasonable candidates for route matching. As we propose toward the end of this paper, looking at the time of day and day of week may help make the distinction.

Another performance metric to explore is the correct prediction rate based on trip progress. Figure 17 shows a quadrant of graphs; the upper quadrant is by percent of trip complete and the lower quadrant is by current travel distance. By the end of the first mile, 30% of repeat trips are correctly predicted (13% overall). However, by this point, the correct prediction for repeat trips is already within the top 10 closest routes 70% of the time—by the third mile this percentage increases to 80%. Depending on the application, even an n-best list of matching routes could be useful. If there is an important warning, for example an upcoming accident site, then alerting the driver even with some uncertainty about his/her route may be worthwhile. In the case of a hybrid vehicle, the control system could plan for the worst case route (or the average case) until a clear winner emerges. Our 2nd algorithm, which we describe next, returns confidence estimates along with its predictions—these could be used to better reason about instituting dynamic behaviors based on the predictions.

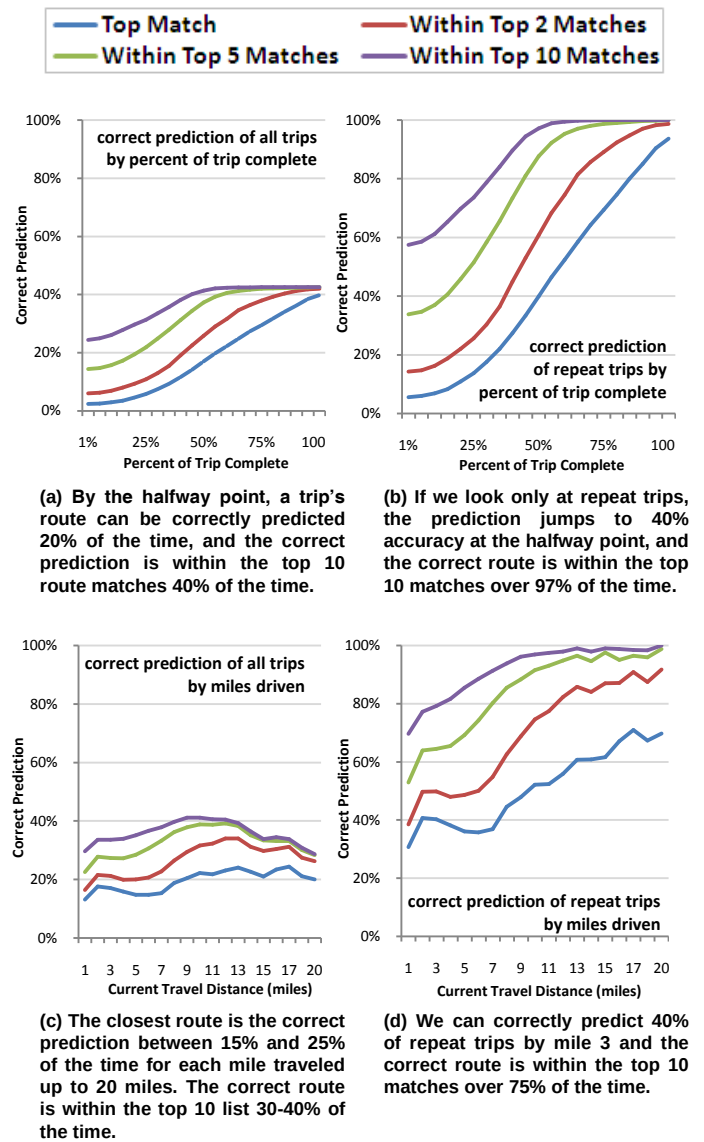
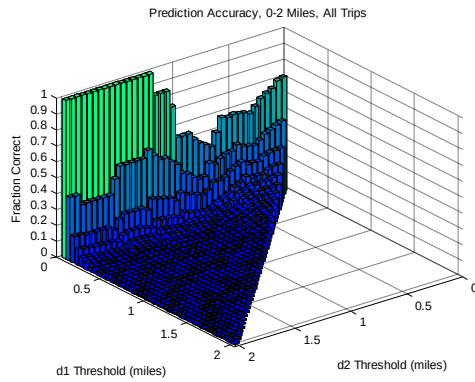


Figure 17. These graphs show the performance of the closest match prediction algorithm. The figures on the top row are by percent of trip complete while the figures on the bottom row are by distance traveled (in miles). The left column includes route prediction for all trips while the right column only includes repeat trips.

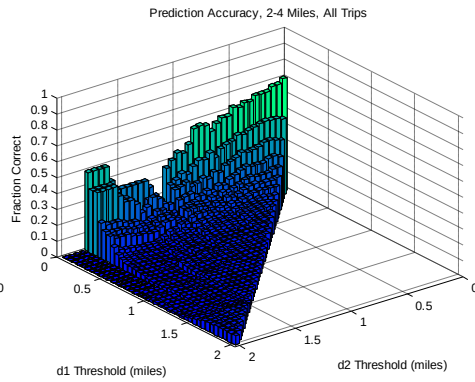
THRESHOLD MATCH PREDICTION RESULTS

The threshold match prediction algorithm receives as input the current trip's travel distance and the trip's distance to the 1st and 2nd closest routes (d_1 and d_2). The parameters d_1 and d_2 are not point-to-point distances but rather the Hausdorff-like, trip similarity measure described earlier. The algorithm returns the fraction of closest routes that were correctly predicted in the past with those combinations of values. Our intuition was that the percentage of correctly predicted routes would rise proportionally as d_1 grew small and d_2 grew large. In other words, our predictions would become more accurate as the current trip comes closer to the closest route and farther away from the 2nd closest route.

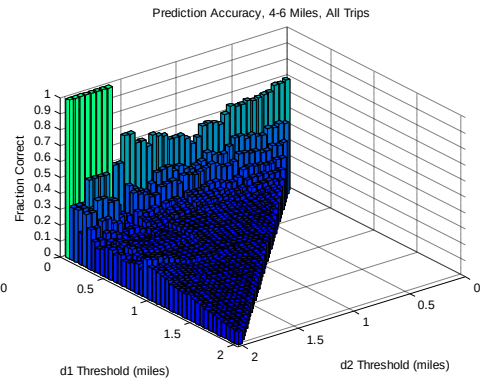
All Trips



(a) Within the first 0 – 2 miles of a trip, the prediction accuracy largely follows our intuition. When d_1 is small and d_2 is comparatively large, the accuracy is high. For example, when $d_1 \leq 0.05$ miles and $d_2 > 1$ mile, our accuracy is greater than 85%.

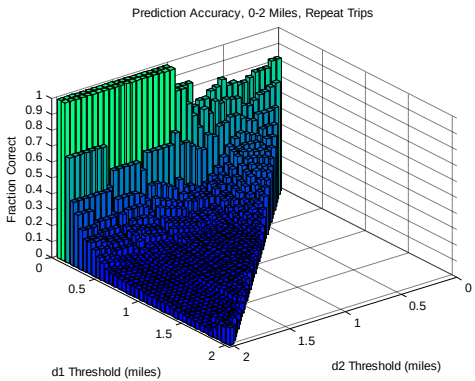


(b) Here, when both d_1 and d_2 are small ($d_1 \leq 0.05$ and $d_2 > 0.05$) we achieve 71% accuracy. As d_1 grows large, the accuracy drops to less than 10%. Interestingly, we not see the rate of predictions increase as d_2 grows large.

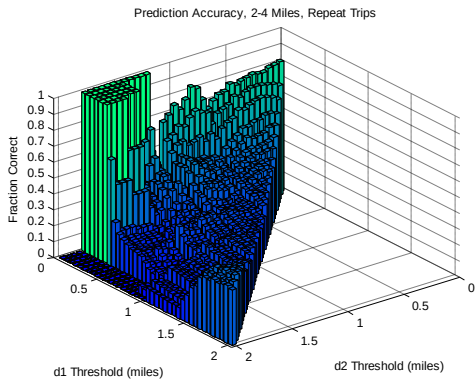


(c) At the 4 – 6 mile point in a trip, more is known about the trip's path structure and thus the overall predictions are slightly better than the previous intervals.

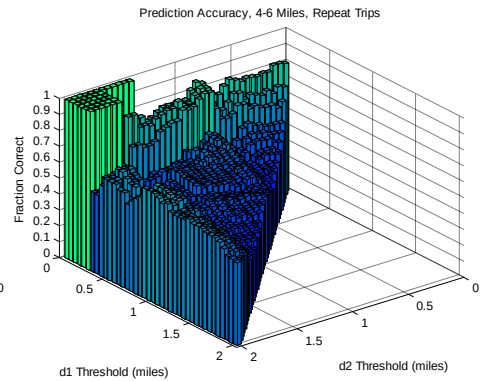
Only Repeat Trips



(d) Similar to (a), when a trip just begins and only one route is close, it is highly likely that this route is the correct match. For all values of d_2 , when d_1 is very small we correctly predict the route more than 70% of the time.

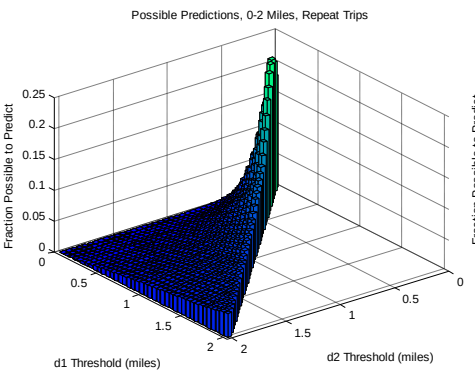


(e) As expected, the overall prediction rates are greater for repeat trips than for all trips. Compare this graph to that found in (b) above.

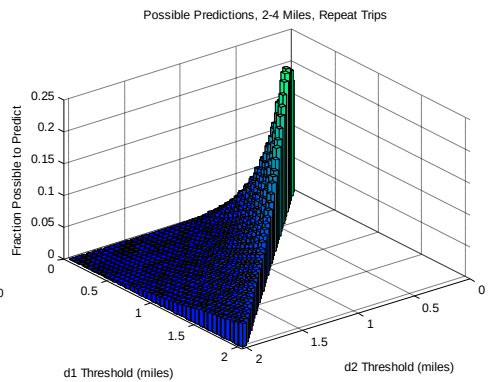


(f) By the time a trip reaches 4-6 miles in travel distance, there is often enough path structure to correctly predict the route even with relatively high values of d_1 . The entire row of predictions when $d_1 < 0.05$ miles has an average accuracy of 85%.

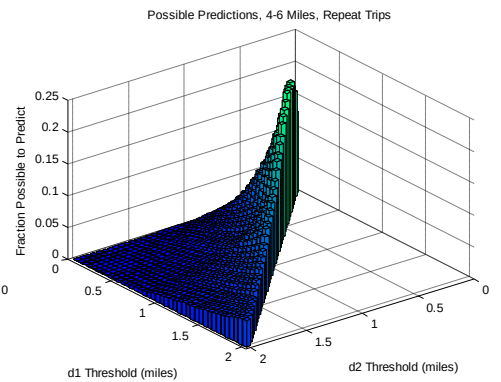
Frequency of d_1 , d_2 Values



(g) The frequency of occurrence of d_1 , d_2 combinations of values at 0 – 2 miles.



(h) The frequency of occurrence of d_1 , d_2 combinations of values at 2 – 4 miles.



(i) The frequency of occurrence of d_1 , d_2 combinations of values at 4 – 6 miles.

Figure 18. These graphs were constructed by iterating over d_1 and d_2 threshold values at different trip distance intervals and calculating the correct route prediction accuracies when $d_1 \leq$ the d_1 threshold and $d_2 >$ the d_2 threshold. The top two rows of graphs show the prediction accuracies for all trip and repeat trips respectively. The bottom row shows the frequency of these d_1 , d_2 combinations in our data.

Our results were computed by iterating over a set of test d_1 and d_2 threshold values for discrete moments in a trip and calculating the percentage whose routes were correctly predicted. In particular, we checked how often the closest route was the correct route when d_1 was *less than or equal* to the test d_1 threshold and d_2 was *greater than* the test d_2 threshold (i.e., $d_1 \leq \text{threshold } d_1$ & $d_2 > \text{threshold } d_2$). Although we ran a range of experimental values, the graphs included in this paper were constructed by setting the initial thresholds to $d_1 = d_2 = 0.05$ miles. We then incremented the d_1 threshold value in an outer loop and the d_2 threshold value in an inner loop to build a 2D-array of prediction performance for those thresholds. We expected much higher accuracies when d_1 was small and d_2 was large. The 0.05 mile increment and starting value were chosen based on the density of data when d_1 and d_2 are small (a majority of our trip data is relatively close to an existing route). We ran this experiment in travel distance bins of 2 miles long from 0 – 2 miles to 20-22 miles. Given that the average trip length in our corpus was 7.7 miles, we present graphs from 0 – 2 miles, 2 – 4 miles, and 4 – 6 miles as they cover a large portion of trips before they have completed.

Figure 18 displays a 3 x 3 graph matrix of our results. Each row of graphs represents a different aspect of the threshold match algorithm's performance. Each column is organized by travel distance: 0 – 2 miles, 2 – 4 miles, and 4 – 6 miles. The top row shows prediction accuracies for *all* trips, the second row shows prediction accuracies for *repeat* trips, and the bottom row shows the occurrence frequency of those d_1 , d_2 combinations at those travel distances.

The data in the plots in Figure 18 could be used by an application engineer to assess the prediction algorithm's confidence. Given the distance into the trip and the values of d_1 and d_2 , the first two rows of plots give the fraction of time that the nearest route was the correct prediction, based on our experiments. Sometimes the algorithm was 100% correct in its predictions, and other times it was lower. The location-aware application can then decide what action to take. For example, a hybrid vehicle may change its battery rate of charge based on the confidence of the route prediction. The bottom row of plots shows how often the corresponding d_1 and d_2 occurred in our test data. Some trips never satisfy certain settings of the thresholds, so they would not trigger a route prediction for that particular setting. This gives an idea of how often we can actually make a prediction of a given confidence.

Overall, the graphs tend to follow our expectations—as the closest route (d_1) and 2nd closest route (d_2) diverge, our prediction accuracies increase. We also note that predictions for repeated trips (2nd row of plots) are generally more accurate than those for all trips, which include both single and repeated trips (1st row of plots). As we showed previously, longer observation times result in the discovery of more driver repetition, meaning

that we expect our prediction accuracy would rise as the driver is observed for more days.

One aspect that is immediately identifiable from the last row of graphs is that there is a very high density of trips where both d_1 and d_2 are small. This makes the problem of correct end-to-end route prediction even more challenging—it means that a trip often has at least two close routes as it progresses. Distinguishing the two may not be possible by focusing completely on their geographic similarity. In the next section we propose additional attributes that could be used to aid prediction in these cases.

CONCLUSION

We begin this section by enumerating potential areas of future work before summarizing this paper's primary contributions and concluding.

FUTURE WORK

There is much future work to be done. First, we only incorporated one feature, that of geographic distance, into our route prediction algorithm. In the future, we would like to explore how incorporating additional features such as frequency and recency of route travel would allow us to make more accurate predictions earlier in a trip's lifetime. Other factors such as identifying the vehicle's driver or whether or not there are passengers in the car may also aid prediction but are more difficult to detect.

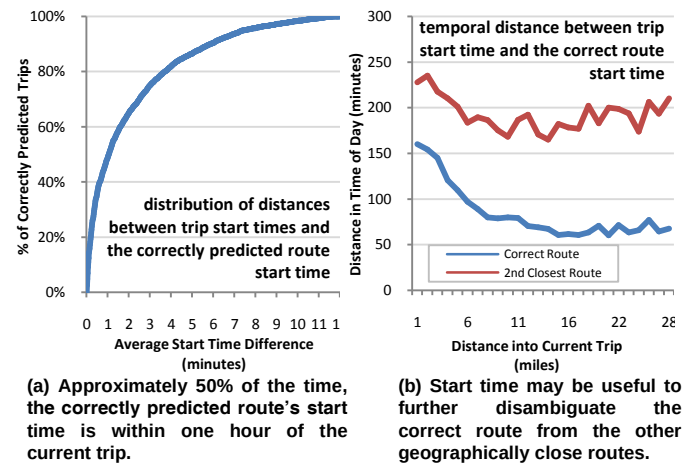


Figure 19. In both of these figures the correct route was predicted using *only* the geographic-based trip similarity algorithms described in the paper. Note, however, that there is a strong temporal pattern between a trip and its matching route as well.

In addition, we have just begun exploring route periodicity—that is, the temporal patterns that exist in route driving behavior (e.g., by time of day, day of week). Although some related work [5, 7] suggests that the temporal aspects of a route do not increase prediction accuracy, these papers focused on near-term predictions and not on routes as a whole. We have early

evidence to suggest that time may be leveraged to improve general route prediction (Figure 19).

Finally, correctly predicting a route that has never been traveled is an open problem. In many cases, it is simply not possible to predict a driver's route once they deviate from their trip history. We suspect, however, that a few elements may be leveraged to increase accuracy under certain conditions:

1. Rather than focusing all of the prediction efforts on full route matching, for those trips that follow a new route, it may be possible to predict at least part of the route (e.g., if the latter part of a new trip overlaps with an existing route like in Figure 15b). We did not explore partial route matching in this paper.
2. Although we have yet to investigate this in our data, our intuition is that for many new routes, the *return route* will follow the same path but in the opposite direction. At best, this could decrease the amount of new routes by 50% if we assume that all new routes are driven in reverse on the return trip. If nothing else, the return route could be weighted higher by the prediction algorithm.
3. Finally, driver familiarity, general route popularity, common destinations among area population, optimal path behavior [9], etc. could all be used as heuristics to guide new route predictions.

SUMMARY

In this paper, we showed how the regularity of a driver's traveling behavior could be exploited to predict the end-to-end route for their current trip. We made three primary contributions. First, we provided a methodology for automatically extracting routes from raw GPS data without knowledge of the underlying road structure. Second, we presented a detailed discussion and analysis of repeat trip behavior from a real world dataset of 14,468 trips from 252 drivers. Finally, we developed and evaluated two algorithms that used a driver's trip history to make route predictions of their current trip. Although performance varies, we believe some application areas such as improving hybrid vehicle efficiency and dynamic traffic alert systems could still benefit from long-term route predictions even with a degree of uncertainty.

CONTACT

Jon Froehlich
PhD Student
University of Washington
Computer Science and Engineering
185 Stevens Way, PAC 101
Seattle, WA 98195

jfroehli@cs.washington.edu

REFERENCES

1. Deguchi, Y., et al., *Hev Charge/Discharge Control System Based on Navigation Information*, in *SAE Convergence International Congress & Exposition On Transportation Electronics*. 2004: Detroit, Michigan USA.
2. Tate, E.D. and S.P. Boyd, *Finding Ultimate Limits of Performance for Hybrid Electric Vehicles*. 2001 SAE Transactions - Journal of Passenger Car - Mechanical Systems, 2001.
3. Karbassi, A. and M. Barth, *Vehicle Route Prediction and Time of Arrival Estimation Techniques for Improved Transportation System Management*, in *Intelligent Vehicles Symposium*. 2003. p. 511-516.
4. Torkkola, K., et al., *Traffic Advisories Based on Route Prediction*, in *Workshop on Mobile Interaction with the Real World (MIRW 2007)*. 2007: Singapore.
5. Simmons, R., et al., *Learning to Predict Driver Route and Destination Intent*, in *2006 IEEE Intelligent Transportation Systems Conference*. 2006: Toronto, Canada. p. 127-132.
6. Patterson, D., et al., *Inferring High-Level Behavior from Low-Level Sensors*, in *UbiComp 2003: Ubiquitous Computing*. 2003, Springer: Seattle, Washington USA. p. 73-89.
7. Krumm, J., *A Markov Model for Driver Turn Prediction*. Society of Automotive Engineers (SAE) 2008 World Congress, 2008. Paper 2008-01-0201.
8. Marmasse, N. and C. Schmandt. *Location-Aware Information Delivery with comMotion*. in *HUC 2K: 2nd International Symposium on Handheld and Ubiquitous Computing*. 2000. Bristol, UK: Springer.
9. Krumm, J., *Real Time Destination Prediction Based on Efficient Routes*. SAE 2006 Transactions Journal of Passenger Cars - Electronic and Electrical Systems, 2006.
10. Hu, P.S. and T.R. Reuscher, *Summary of Travel Trends, 2001 National Household Travel Survey*, U.S.F.H.A. U.S. Department of Transportation, Editor. 2004.
11. Huttenlocher, D.P. and W.J. Rucklidge, *A Multi-Resolution Technique for Comparing Images using the Hausdorff Distance*, in *Computer Vision and Pattern Recognition*. 1993. p. 705-706.