Monitoring and mining GPS traces in transit space*

Leon Stenneth[1, 2], Philip S. Yu[2]
[1]Nokia L&C, Chicago, Illinois, USA
[2]Department of Computer Science
University of Illinois, Chicago, USA
lstennet@cs.uic.edu, psyu@cs.uic.edu

Abstract
Users of mass transit systems such as those of buses and trains normally rely on accurate route maps, stop locations, and service schedules when traveling. If the route map, service schedule, or stop location has errors it can reduce the transit agency's ridership. In this paper, the problem of deriving transit systems by mining raw GPS data is studied. Specifically, we propose and evaluate novel classification features with spatial and temporal clustering techniques that derive bus stop locations, route geometries, and service schedules from GPS data. Subsequently, manual and expensive field visits to record and annotate the initial or updated route geometries, transit stop locations, or service schedules is no longer required by transit agencies. This facilitates a massive reduction in cost for transit agencies. The effectiveness of the proposed algorithms is validated on the third largest public transit system in the United States.

General terms
Algorithms, Design, Experimentation, Performance

Keywords
Classification, clustering, GPS tracking, maps, mobile, probe

1. Introduction

One goal of transit agencies is to increase the ridership of its transits. A strategy for transit agencies to increase their ridership is to improve the user experience of the commuters who utilize these transit systems. A way of improving the user experience has taken the form of intelligent transportation systems (ITS), where spatial and temporal information such as route geometries, stop locations, real time bus locations, and service schedules are correct and accessible via the Internet. For example, see [5].

There is a high cost associated with these intelligent transportation systems. Commercial ITS providers such as NextBus [1] and Clever Device [3] are the predominant venders. However, usage of these services incurs extensive initial and recurring fees. One transit agency budgeted over $20M USD for implementation of such intelligent transportation systems [2]. Transit agencies in emerging markets or developing countries may not have such substantial transportation budgets. Hence, they cannot provide these basic services to the riders of their transits.

Often, transit systems have predetermined knowledge of their transit artifacts such as bus stop locations, schedules, and route geometries (i.e. actual current transit path on maps). The problem for these agencies is maintaining updated artifacts, since artifacts can change due to re-routes, weather, construction, event etc. In this paper, novel technology is presented that automatically derives and updates transit systems from the raw GPS traces of the transits. In other words, using these newly proposed algorithms, transit agencies can derive or update their own transportation systems. This automated strategy to derive a transit system from raw GPS traces reduces cost to the transit agency.

*This work is supported in part by Nokia L&C, NSF through grants IIS-0905215, CNS-1115234, IIS-0914934, DBI-0960443, OISE-1129076, and Huawei grant.

In this work, we use the terms transits and buses interchangeably. This proposed work first tracks the moving objects (i.e. buses) for a period of time. During the tracking phase, the locations and timestamps of the buses are recorded. The location of the buses can be determined by a location enabled smartphone [2] or from an onboard GPS device in the bus.

Given the location and timestamp data obtained from the location tracking phase, the transit artifacts consisting of bus stop locations, route geometries, and timetable schedule is derived automatically, using spatial and temporal mining on the collected GPS data. Using data mining technology, we propose three algorithms that can extract transit systems from raw GPS traces [2, 26, 27, 28]. The first is a supervised learning bus stop detection algorithm. The second is a spatial clustering route derivation algorithm. Finally, a temporal clustering scheme that produces a service schedule is proposed.

For bus stop detection, there is first a route aggregation phase where GPS data from the same transit route are processed together. With this aggregated route based data, a supervised learning technique that captures the mobility pattern of the probes using a density distribution histogram is considered for automated bus stop detection. This strategy first forms cluster points using a spatial constraint scheme. After this spatial constraining step, the results are then utilized in a subsequent aggregation step, which considers a new spatial requirement. This new spatial requirement enables pattern mining on the residuals, and also acts as a constraint on the distance between the predicted and the true bus stop location. Residuals are clusters associated with the initial spatial constraint. Finally, a density distribution histogram that captures the distribution of the residuals within the latter aggregation is constructed. The 10 normalized bins of the histogram and three other mobility patterns become supervised learning classification features for bus stop detection.

For route derivation, a simple yet effective six step spatial clustering algorithm is proposed. This algorithm coalesces the different GPS traces from different buses into a single line string (i.e. successive latitude and longitude pairs) representing the route. Spatial outliers that result from GPS inaccuracies are suppressed and pruned, while the remaining points are used to construct the route. The algorithm operates on historical probe archives or real time probe data streams using a spatial and temporal ordering strategy.

For service schedule extraction, the temporal property of the GPS data is mined using a K-means temporal clustering scheme. This scheme computes and utilizes the number of trips for a given bus stop as the K value in K-means.

In summary, the research and scientific contributions of this manuscript are as follows.

- A supervised learning algorithm for automatically deriving bus stop locations from the collected GPS data.

This algorithm requires no driver interaction or user input.

- A six stage spatial clustering algorithm that automatically derives the set of service routes and the geometries that represent them.
- A temporal clustering strategy that generates a given transit agency's service schedule for each bus stop.
- An evaluation of the proposed algorithms on the third largest transit system in the USA. Namely, the Chicago Transit Authority (CTA) [4].

The remainder of the paper is organized as follows. Next, we highlight other motivations and challenges. Section 3 deliberates the definitions and model, while the supervised learning bus stop detection strategy and its evaluation are presented in Sections 4 and 5. Section 6 addresses transit route derivation and Section 7 discusses service schedule derivation. The related work and conclusion are in Sections 8 and 9 respectively.

2. Motivation and challenges

In this section, the motivation for the research is presented. Additionally, the challenges are highlighted and discussed.

In this work, the minimum requirement for a real time transit tracking system is an in-vehicle location tracking device. This in-vehicle device, sometimes referred to as an Automatic Vehicle Locator (AVL), uses GPS or other localization systems to determine the bus's current location. The location information is then transmitted over a wireless link to the back office. The back office component is a server that processes the incoming location traces, and typically provides a live tracking website for the public and status monitoring for the transit agency's dispatchers.

Several motivations for this work exist. Transits agencies find it costly to derive and update the transit artifacts (route geometries, stop location, service schedules) of their service areas [1, 2]. These transit artifacts are not static, for example bus stops can be added or removed from a transit route for several reasons, such as, extreme weather or construction. Likewise, the geometry of a route may change due to bus re-routes. Thus, these transit artifacts must be kept updated in order for the correct information to be presented on the map. Using these proposed algorithms, we can derive the updated location of bus stops automatically by tracking the transits and then mining the spatial and temporal data obtained from the tracking phase. Additionally, the historical route geometry and updated deviations from the historical routes can be derived. Thus, the issue of transit agencies paying companies such as NextBus [1] or Clever Devices [3] millions of dollars to derive and monitor the transit network can be avoided. Instead, the transit agency can utilize the proposed algorithms. One transit agency with a budget of over USD $250M paid USD $24M for intelligent transportation services that monitors their transit system [2, 4]. Yet, they still cannot recreate routes on demand or automatically detect addition or removal of bus stops. This work targets these updating problems in mature transit systems. Additionally, the proposed solutions are effective for emerging smaller transit agencies to derive their initial transit information systems.

Several challenges exist in this research. These include for example:

- Localization via GPS can be inaccurate; hence the location that the buses report may not be the true location.
- Each bus on the same route will report completely different location points.
- Buses deviate from their true routes at times. For example, a bus may detour because of an accident.
- Buses submit GPS reports every 20-30 seconds. Generally, the reports are sparse. Thus, a bus may service a bus stop quickly without submitting a GPS report while at the bus stop.
- The data model is limited since we consider the minimum requirement which is location and timestamp.

3. Definitions and model

In this section, the definitions, model, and architecture are discussed.

Definition 1 – *Probe – We define a probe P to be a vector of the form (sys_id, route_id, bus_id, loc^{gps}). Where sys_id is the identification of the transit agency (e.g. CTA or San Francisco's BART), route_id is the route that the corresponding bus services, bus_id is the identification number of the bus, and loc^{gps} is the location property of the bus.*

In this work, in order to validate the algorithms we consider the third largest transit network in the United States; this is the Chicago Transit Authority (CTA). In this system, probes are submitted by each bus every 20-30 seconds.

Definition 2 - *GPS report – We define a GPS report loc^{gps} to be a vector of the form (lat, lon, time) where lat is the bus's latitude and lon and time is the longitude and timestamp of the report respectively.*

Definition 3 - *GPS trace – For a given bus b_i, a GPS trace is a temporally ordered sequence of GPS reports submitted by b_i.*

Given *n* buses b_1, b_2, b_3... b_n, where each active bus b_i submits a probe P_i^t at time t. Thus, over time, for each bus b_i we have a list of probes $P_i^t, P_i^{t+1}, P_i^{t+2}, P_i^{t+3} \ldots P_i^T$ where P_i^T is the last probe submitted by b_i. Each probe has a GPS report and from the list of probes $P_i^t, P_i^{t+1}, P_i^{t+2}, P_i^{t+3} \ldots P_i^T$ we can get a GPS trace for the bus b_i.

The probes that are submitted by the buses become the input for the algorithms. In order to evaluate the algorithms, we utilize over 10,000,000 probes. From the probes, we will use newly proposed supervised and unsupervised data mining techniques to extract the location of bus stops, route geometries, and service schedule.

Assumption: We assume that buses may stop periodically at the start or end of trips.

3.1 Architecture and data flow

The architecture is depicted in Figure 1. As illustrated in Figure 1, the buses possess a tracking component that periodically transmits its location data to the location server. This data can be transmitted over a cellular link. The in-vehicle location device may be permanently installed in the bus (e.g. on-board GPS) or carried by the driver (e.g. smartphone with localization capabilities). The traces from the buses are then archived.

Using the route aggregator, probes from the same routes are aggregated and then processed. After the route aggregation phase, the proposed supervised learning algorithm is then activated for bus stop extraction. Likewise, the spatial clustering route extraction and the temporal clustering schedule extraction algorithms are also activated.

The derived bus stop locations, route geometries, and service schedules are called transit artifacts and are archived for use in the transit agency's information system. Once the transit artifacts are

derived, the buses' spatial and temporal properties are then monitored for deviations. When deviation occurs, alerts are activated in the system. For example, using transit artifacts we are aware of the route geometry and bus stop locations. If buses deviate from the historical route geometry in the transit artifact, then a re-route alert is activated. We can also generate bus stop removal/addition alerts in real time. Further, if buses deviate from their historical scheduled time, we generate temporal based alerts such as bus delayed or running ahead of schedule.

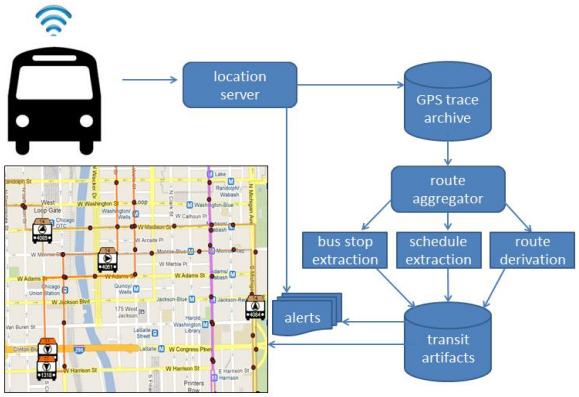

Figure 1- architecture overview

4. General algorithm for bus stop detection

In this work, stop extraction is the process of turning a set of raw GPS traces belonging to a given route into a small set of coordinates; indicating the locations of transit stops. The locations generated by the proposed algorithm are used for producing arrival times for schedule extraction or for drawing stop locations on a route map, this allows travelers to be aware of boarding locations.

Here, we define the concept of a mini-cluster.

Definition 4: *mini-cluster – In this work, a mini-cluster is a cluster formed with probe points that are a spatial requirement of 3m from each other*.

In this work, 3m is the distance selected, empirically, to balance between sufficient data density and reasonable run time. However, one can tune this spatial requirement and observe the bus stop detection accuracy of the machine learning model before system deployment. The cluster algorithm is purely spatial, and a cluster's center continues to shift when a new GPS point is added to a cluster. A GPS point can only exist in a single mini-cluster. In general each cluster has a center which is the weighted average of all the points within that cluster.

Definition 5 - c_{max} - *We define c_{max} to be a value corresponding to the size of the densest mini-cluster along the route. Thus, c_{max} varies per route and across routes.*

Based on experiments, we found that c_{max} is at the start or the end of routes. The maximum cluster size is obtained by comparing the number of GPS reports within each mini-cluster along the entire route. Mini-clusters provide information that can be used to distinguish a bus stop from other points. At bus stops, we expect a few mini clusters with high densities. For a stop light along a route, the distribution of the mini-clusters is sparser and density of mini-clusters is lower than at bus stops. Given these mini-clusters, we then form cluster sets which are 20m bounding boxes along a route that contains several mini-clusters. Thus, bus stop detection is approximately within 20m from the true bus stop location.

Definition 6 – *cluster set – a cluster set is a square bounding box with coordinates (x, x', y, y'). In this work, x'-x =20m and y`-y =20m.*

The value of 20m was chosen empirically and ensures that when the algorithm predicts a bus stop location it is reasonably close to the true bus stop location at most 20m away. One can tune this spatial requirement.

The intuition behind this step of mini-clusters being overlaid by cluster sets is that for bus stops, buses always stop close to the bus stop for passengers to board/alight. Consequently, at a bus stop the number of GPS reports should be high (i.e. dense mini-clusters). For other points on the route, since the buses are not stopping the number of GPS reports should not be as high as at a bus stop. On the other hand, regions around stop lights and stop signs may also observe a high frequency of reports. However, these objects (stop lights and stop signs) have different distribution patterns which we can use to identify and distinguish between them and bus stops. For one example, at stop signs buses hardly ever stop completely, for a long time. We observe that they (i.e. buses) mostly slow down or stop briefly. For another example, in the case of stop lights, buses stop at arbitrary points when adhering to any given stop light. In other words, for a given stop light, a bus maybe adhering and stops at 400m (e.g. in traffic), 40m, 50m, or 5m from the stop light. This is not the case for bus stops, instead, drivers tend to stop their buses as close as possible to the bus stop because of their business. Using spatial pattern recognition principles, machine learning models are then trained to automatically recognize and identify these mobility patterns.

The mobility patterns for classification are then mined from the cluster sets along the route. These mobility patterns capture the mini-cluster density and distribution within the cluster sets using 13 newly proposed classification features. Ten of the 13 classification features are bins of the histogram that represent the density distribution of the mini-clusters within a cluster set. The other 3 features are related to other mobility patterns. With these 13 classification features, multiple supervised machine learning models (e.g. Bayes Net, Random Forest, and Decision Trees) are trained. Then, when bus stop location is to be determined, the unlabeled data is used to create the same features that were used for training. These features are then fed to the machine learning model that is already trained. This trained model can then determine, in a probabilistic format, the presence of a bus stop. Given this trained supervised machine learning model, we can then utilize it on any transit system to discover or update bus stop locations.

4.1 Parameter tuning

In this work, the spatial requirements for mini-cluster and cluster set were chosen empirically. Realistically, one should tune these parameters while observing the accuracy of the supervised learning scheme in order to deduce the optimal spatial parameter settings.

4.2 Generate distribution histograms

A histogram for each cluster set layered along the route (See Figure 2) is generated. In Figure 2, the boxes represent the cluster set and within each cluster set are the mini-clusters. A distribution histogram is then generated for each cluster set. This histogram captures the distribution of the GPS reports within each mini-cluster in the cluster set. The histogram contains 10 bins. Each bin of the histogram is a fraction of the maximum cluster size c_{max}.

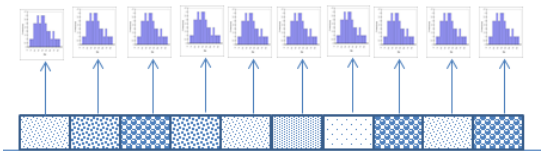

Figure 2- Generate histogram for each cluster set

More specifically, each bin of the histogram except the last bin contains 5% increment in terms of fraction of c_{max}. Thus, the first bin contains the number of mini-clusters in the cluster set with number of reports <= 5% of c_{max}, the second bin contains the number of mini-clusters within the cluster set with number of reports <= 10% of c_{max}, the third bin contains the number of mini-clusters within the cluster set with number of reports <= 15% c_{max} (i.e. 5% increments) the last bin contains the number of mini-clusters within the cluster set with number of reports > 45% of $c_{max.}$ As mentioned, based on our observation, c_{max} is at the start or end of the routes since the buses normally sit at these points. Given this histogram, a normalization step is considered.

4.3 Normalize histogram bin counts

Given the density distribution histogram, there is a normalization stage that follows. For each histogram, each bin frequency is represented as a fraction of the total number of points represented in that histogram. Subsequently, the ratios between the bins are maintained, but the scale of their magnitudes is normalized. Thus, this avoids bias when comparing samples with different densities. Observe that after normalization the sum of all bin frequencies for each histogram is 1.0.

4.4 Supervised learning feature vectors

We propose and utilize a new classification feature vector consisting of 13 new classification features. The 10 normalized bin counts from Section 4.2 become classification features. In addition, we also derive 3 other classification features. The other 3 features considered are:

1. For each mini-cluster, we compute the number of GPS reports with the bus stationary (e.g. <=2m/s) in each mini-cluster. The speed is computed from consecutive latitudes and longitudes of a given bus. Speed computed this way can be inaccurate since buses submit GPS reports every 25-30 seconds (i.e. sparse). However, computation of speed from raw GPS gives an indication of a bus sitting at a location for a period of time (i.e. stopped at a bus stop or stop light). Then, for a cluster set, the classification feature *low speed weight (lsw) = number of low speed reports in the cluster set / total reports across the cluster set.* A low speed report is one where the computed speed is low, for instance, less than 2m/s.
2. Computed *average speed across clusters (asac).* Each probe point has a computed speed which is derived from successive location changes, this allows for computation of the average speed for a cluster set.
3. Summation of *bin1+ bin2* of the histogram. This is a representation of the density of the first two bins. The intuition for this feature is that points on the route that are not bus stops, stop lights or stop signs, have sparse GPS reports. Thus, these lower level bins become special signals.

4.5 Machine learning component

In this section, we present the machine learning model, dataset, and the machine learning algorithms that we considered for evaluation.

4.5.1 Training model

First, the proposed algorithms are validated on the Chicago Transit Authority (CTA) transit system [4]. To build the training data we collected nine weeks of data from the over 1000 CTA buses. CTA buses submit probes every 25-30 seconds. In total we collected over 10,000,000 probes from buses. In the CTA system, over 12,000 bus stops and over 100 routes exists. The training data is constructed as follows. First, from CTA's bus route API [5], for any route of interest, we can obtain the true location of the bus stops from the agency (ground truth). We can also obtain true location points along the route that are not bus stops. Given these two types of points (bus stop points, non-bus stops points) and the mini-clusters formed from the raw GPS traces, we then form cluster sets along a route and then observe the mini-clusters within.

Within each cluster set that is overlaid along the route, there can be several mini-clusters. From each cluster set, we extract the 13 features from the mini-clusters within. Using this strategy, for each cluster set, we can extract the 13 classification features with the two labels for the training data (1) bus stops and (2) non-bus stops.

In addition to the data obtained from CTA's API, we also directly considered the location of stop lights and stop signs. Subsequently, from a spatial data set of over 20,000 stop lights and over 7,000 stop signs in Chicago, we obtained additional route points that are not bus stops, instead, these points are the locations of the stop lights and stop signs. We wanted to directly include stop lights and stop signs that are not close to bus stops into the model, this way the buses' mobility patterns at stop lights/stop signs, bus stops, non-bus stops can be understood and used to train the model. These stop light/stop sign points are labeled as non-bus stops and are used to make the model realistic and robust. Understanding and recognizing the mobility patterns of buses at stop lights and stop signs makes bus stop detection interesting since these points look deceivingly similar to bus stops in GPS data, because buses also stop at stop lights and stop signs.

Finally, in total, the training data has two labels, (1) bus stops, (2) non bus stops. Non bus stops are both regular route points and stop lights/sign points.

Based on the histograms generated, we observe a pattern. For regular points that are not bus stops, stop lights, or stop signs, the frequency is higher for the first bin/s of the histogram. This implies that the number of reports is much less than $c_{max.}$ This makes sense, since the buses are not sitting at these points, the number of reports from these location is small.

Example after normalization for a typical route point

1.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0

Additionally, for stop lights and stop signs, the pattern from the histogram is also obtained. Buses stop at these points briefly, not longer than at bus stops but longer than regular route points. Thus, the histogram's first few bins contain the most weight.

Example after normalization for a typical stop light/sign

0.66, 0.33, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0

Further, for the case of bus stops, since buses are stopping for passengers to board/alight the bus, the number of GPS reports at

these points is high and dense. Thus, the first set of bins is normally empty or has less information. Instead, the middle and last set of bins are active.

Example after normalization for a typical bus stop

0.0, 0.0, 0.5, 0.0, 0.375, 0.0, 0.125, 0.0, 0.0, 0.0

4.5.2 Machine learning algorithms and setup

We trained multiple machine learning models to recognize the mobility patterns of buses to be able to predict the location of bus stops. These trained machine learning models include Bayes Net, Decision Trees, and Random Forest. To test the accuracy of the models, we considered the 10 fold cross validation strategy.

5. Results on bus stop detection

In this section, the results of the proposed scheme are discussed. The results using the three machine learning algorithms (Bayes Net, Decision Trees, and Random Forest) are presented.

Training data distribution

In total we used 3663 training instances. Of these 3663 instances, 1955 are patterns of buses' at bus stop and 1708 are non-bus stop patterns. Of the 1708 non-bus stop patterns, 270 are the buses' patterns at stop light/stop signs. We used 270 because in the experimental region only 270 stop signs/stop lights exist that is not at bus stops. The remaining (i.e. 1438) patterns are not bus stops, stop lights, or stop signs. We call these regular route points. The results for the three models are presented in Figures 3(a, b, c).

Bayes Net

	Correct	Incorrect	Precision	Recall	F-Score
Bus stop	1451	504	73.80%	74.20%	74.00%
Non bus stop	1192	516	70.30%	69.80%	70.0%
Weighted avg					**72.10%**

Decision tree

	Correct	Incorrect	Precision	Recall	F-Score
Bus stop	1401	554	80.0%	71.7%	75.6%
Non bus stop	1358	350	71.0%	79.5%	75.0%
Weighted avg					**75.30%**

Random Forest comprised of 10 decision trees

	Correct	Incorrect	Precision	Recall	F-Score
Bus stop	1630	325	95.30%	83.4%	88.9%
Non bus stop	1627	81	83.40%	95.28%	88.9%
Weighted avg					**88.90%**

Figure 3 (a, b, c) – Bus stop detection evaluation results

In general, we can achieve over 89% accuracy for the newly proposed bus stop detection algorithm. The Random Forest model is the most effective and outperforms both Bayes Net and Decision Tree. This work is the first to have such high accuracy for bus stop detection using supervised learning from probe data. Previous work that derives bus stop locations from GPS traces only achieves 50% precision [2]. Also, the probe data considered in this work consists of only latitude, longitude, and timestamp. Higher order attributes such as speed and heading are not collected from the GPS device. We deliberately considered this limited data model.

5.1 Results on much larger training data set

In this section, we considered a much larger data set. We wanted to study the resilience of the model to training data size. In total, for these experiments, we used 12410 training instances. Of these 12410 instances, 6169 are patterns of buses at bus stop and 6241 are patterns of buses at non-bus stop. Of the 6241 non-bus stop patterns, 801 are the buses' patterns at stop light/stop signs. We used 801 because in the experimental region only 801 stop signs/stop lights exist that is not at bus stops. The remaining (i.e. 5440) patterns are not bus stops, stop lights, or stop signs. 10 fold cross validation using Bayes Net, Decision Tree, and Random Forest is studied. The results for the three classification models are presented next in Figure 4 (a, b, c). From these three Figures (See Figure 4), the classification model is resilient to training data size and Random Forest is still the most effective model. It can achieve over 88% bus stop detection accuracy. The Decision Tree is more effective on average than the Bayes Net. These results indicate that the proposed model is robust to size of the dataset and yields high accuracy under arbitrary but realistic data set sizes.

Bayes Net

	Correct	Incorrect	Precision	Recall	F-Score
Bus stop	4946	1223	72.30%	80.20%	76.00%
Non bus stop	4344	1897	78%	69.60%	73.6%
Weighted avg					**74.80%**

Decision trees

	Correct	Incorrect	Precision	Recall	F-Score
Bus stop	4592	1577	78.5%	74.4%	76.4%
Non bus stop	4986	1255	76.0%	79.9%	77.9%
Weighted avg					**77.20%**

Random Forest comprised of 10 decision trees

	Correct	Incorrect	Precision	Recall	F-Score
Bus stop	5102	1067	92.20%	82.7%	87.2%
Non bus stop	5810	431	84.50%	93.10%	88.6%
Weighted avg					**87.90%**

Figure 4 (a, b, c) – Bus stop detection evaluation results

5.2 Discussion on bus stop detection

Using the buses themselves as probes, we can detect bus stop location from raw GPS data with over 89% accuracy using these newly proposed classification features. This represents an increase of over 30% accuracy on the prior art [2]. Thus, bus stop detection from raw GPS data is one contribution of this paper. One strategy to handle the 11% misclassification is for transit agencies to allow their riders to fine tune the crude transit system that our algorithms derive. For example, the public, transit drivers, or transit riders can be considered to further report on the confidence of each detected bus stop and also identify bus stops that were misclassified or undetected by the proposed bus stop location detection supervised learning scheme. This strategy can increase the coverage beyond 89%.

5.2.1 Bus stop semantic naming

Reverse geocoding converts a latitude and longitude pair to the high level street address. Thus, we can reverse geocode the derived bus stop location to deduce a reasonable high level street name. For higher level bus stop semantic naming such as "Water Tower Mall stop", a naming service can be created to edit the reverse geocoded stop names.

6. Transit route derivation

Route derivation in this manuscript is the process of converting raw probe points into the set of service routes that is followed by the transit agency's moving objects (i.e. buses). As mentioned,

these probes are obtained from on board GPS devices on the, but also can be obtained from GPS enabled smartphones installed in the buses, for instance. This problem of transit route derivation is not trivial since all the buses on the route do not follow the exact same path. Likewise, GPS has errors (e.g. multipathing) that should be accounted for.

Route derivation is important, for example, commuters and drivers would like to have knowledge of their route's geometry before travel. Also, the transit agency may use the route's geometry in their information systems. Furthermore, some transit agencies already have knowledge of their historical route geometry. One challenge for these transit agencies is to reconstruct subsections of the route's geometry after there has been a spatial deviation (i.e. re-route) from the historical route due to an accident or construction for instance. This work on route derivation can be utilized to create these new routes or new route segments on demand. Next, the proposed six step algorithm is discussed.

6.1 Proposed six step algorithm

The algorithm for route extraction has six steps. These six steps are: (1) GPS trace seeding, (2) spatial clustering, (3) dynamic cluster pruning, (4) cluster aggregation, (5) spatial and temporal cluster ordering, and (6) map matching (*optional*)

6.1.1 Step 1 and step 2 - GPS trace seeding & spatial clustering

For a given route, let us call the set of traces t_{set}. For step 1, we pick a seed trace which is the longest GPS trace in t_{set} on the given route. The longest trace has the most GPS reports. Given the longest trace t_1 from step 1, we then perform step 2 as follows.

First, remove t_1 from t_{set}. For all GPS points $p_1, p_2, p_3 \ldots p_z$ on t_1, form a cluster with the GPS points on the traces in t_{set} that have GPS points within 40m. Intuitively, at each GPS point on t_1, we add the closest point on any other trace $t_i \in t_{set}$ unless the threshold of 40m is exceeded. After this step, we have a set of clusters along the route in t_{set}. Each cluster maintains a set of GPS reports and a centroid. The centroid is the weighted average of the reports.

6.1.2 Step 3 - Dynamic cluster pruning

Given the set of clusters in t_{set} from step 2, we compute t_{mean}, the mean number of GPS reports within the clusters. Let $c_1, c_2, c_3 \ldots c_m$ be the clusters along the trace. Then c^i_{size} is the number of GPS reports in cluster *i* and

$$t_{mean} = \frac{\sum_{i=1}^{m-2} c^i_{size}}{m}$$

(NB.) *Without the largest two clusters (i.e. start and end of trips).* The value t_{mean} is utilized as a threshold as follows. Given t_{mean}, clusters with sizes below a fraction (e.g. 1/4) of t_{mean} are pruned. This strategy removes outliers and is dynamic since the mean is always changing as we collect more probe data, thus the requirement for pruning also changes. Consequently, this threshold is not sensitive to the duration of probe data collection. The remaining clusters after step 3 are called t_{set}'. After step 3, inaccuracies due to say GPS inaccuracies are suppressed.

6.1.3 Step 4-Cluster aggregation

In step 4, in order to form routes, *we* aggregate or join clusters by connecting the centers of the clusters in t_{set}'. Intuitively, we want to convert points represented as clusters into polylines or track points. Each cluster has a center that represents the weighted average of all the GPS reports within. Subsequently, connecting the cluster centers produces a crude representation of the route's geometry.

Below in Figure 5a, the clusters formed along a candidate route in Chicago are shown after step 3. First, from the figure, the largest clusters are at the start and end of routes. In other words, the clusters at the start and end of the trips maintain a highest distribution of GPS reports than any other points on the route. This observation makes sense and is in line with the assumption that buses sit at these points. Since the clusters are still not ordered, the route geometry resulting from aggregating the clusters in step 4 is shown in Figure 5b. This route geometry in Figure 5b cannot be utilized.

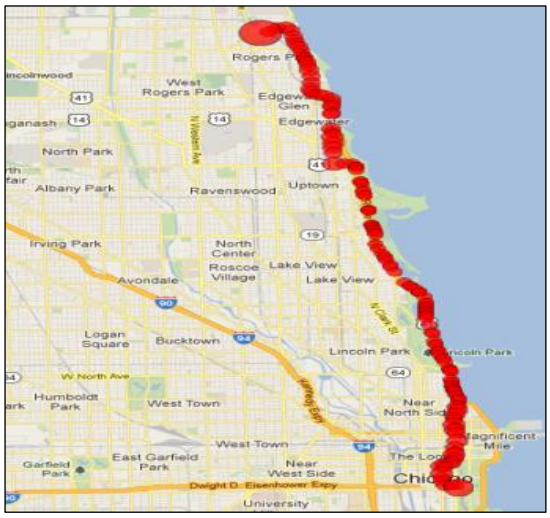

Figure 5 (a) – Before step 4

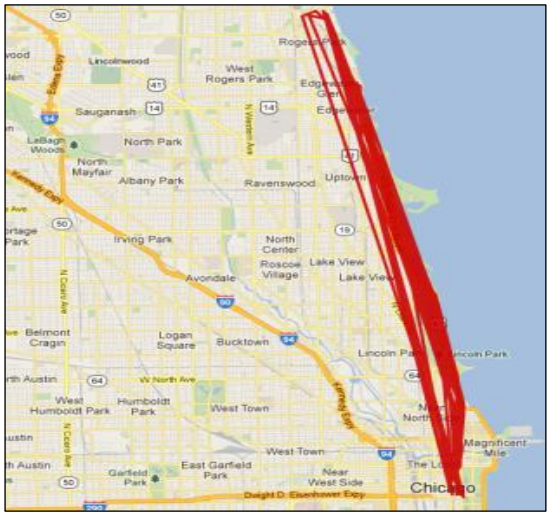

Figure 5 (b) – After step 4

6.1.4 Step 5 - Cluster ordering

As shown in Figure 5b, the aggregated clusters are not ordered after the 4th step in the algorithm. Hence, we cannot generate correct route geometries. In this section, we will explain how to order the aggregated clusters. We will utilize a spatial ordering strategy, discuss its weaknesses, and then utilize a spatial and temporal ordering strategy that addresses the shortfall of the spatial ordering technique.

6.1.4.1 Spatial ordering

Based on our observation, the start and end points of routes have the highest density clusters. Thus, for cluster ordering, the largest

cluster becomes cluster one and we then order the remaining clusters greedily relative to cluster one (i.e. the first cluster).

Formerly, let the set of clusters t_{set}' be represented as $\{ c_1, c_2, \ldots, c_n\}$ and let c_i represent the i^{th} cluster, then c_1 is the cluster with the highest number of GPs reports. Once a cluster is ordered, it is removed from t_{set}'. The algorithm for ordering the clusters spatially is simple and computes greedily as follows.

$$c_i = \begin{cases} c_1 \text{ , if } c_1 \textit{ cluster size is} \text{ maximim} \\ c_j \text{ that leads to min } dist\,(c_{i-1}, c_j \in t_{set}') \textit{ where } j \leq n, \textbf{otherwise} \\ s.t.\, dist\,(u,v) \textit{is the distance between cluster } u \textit{ and cluster } v \end{cases}$$

6.1.4.2 Spatial and temporal ordering

The spatial ordering strategy works well if the route is a simple geometry (e.g. a relatively straight line route). Let us now discuss a case where the spatial ordering scheme fails. Consider the circular route in Figure 6, the direction of the buses is assumed to be A, Q, B, C, and then D. Using a spatial ordering may result in the incorrect cluster ordering results if C's or D's Euclidian distance to Q is lower than the B's. Thus, spatial ordering is not effective for these routes. In general, a spatial and temporal ordering strategy is required.

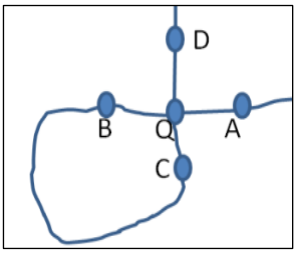

Figure 6 – Spatial ordering shortcoming

The spatial and temporal ordering operates as follows. First temporal ordering takes precedence over spatial and is based on the timestamp of the reports in the clusters. For example, if two clusters have GPS reports from the same bus on the same run on the same day, then we order these two clusters in ascending order with respect to the timestamp of these two GPS reports. This is done for all the clusters along the route. Transitivity of GPS reports from a given candidate bus run, across multiple clusters, is also considered. If, after temporal ordering, some clusters are still not ordered; we then activate the greedy spatial ordering only on those unordered clusters. After step 5, the final route is derived. For example, for the candidate route, the derived geometry by the proposed algorithm is shown in Figure 7(a, b). Figure 7a shows the derived route geometry. Figure 7b shows the zoomed version of the same derived route; it shows that the route's geometry is aligned with the road's geometry. After step 5, the clusters are aggregated and ordered. Consequently, we now have a polyline representing the route. Based on our studies, we found that the derived routes are aligned with the road's geometry in most cases as shown in Figure 7a and its zoomed counter part Figure 7b. In some cases, because of GPS positioning accuracies, the derived route is not well aligned with the road's geometry. This problem is then solved using map matching in step 6.

6.1.5 Step 6 - Map matching

This is the final stage and is optional. Given the derived route geometry as obtained from step 5, one can then use any map matching algorithm [8, 9, 10] to ensure that the derived route is well aligned with the road geometry. Map matching is the process of aligning route geometries with the underlying roadmap. The assumption is that, for the service area, a roadmap may be accessible. Our finding is that the algorithm can derive routes that are aligned with the roads geometry without map matching. However, in cases where the underlying roadmap is available, it should be used to improve the alignment and presentation. For example, for a derived route in the study we show the snapshot in Figure 8.

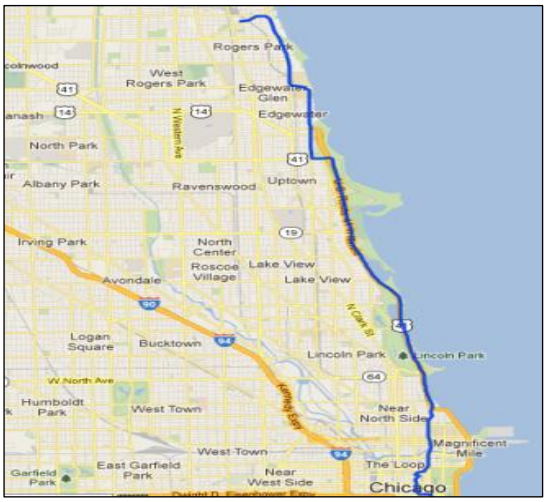

Figure 7 (a) Derived route after step 5

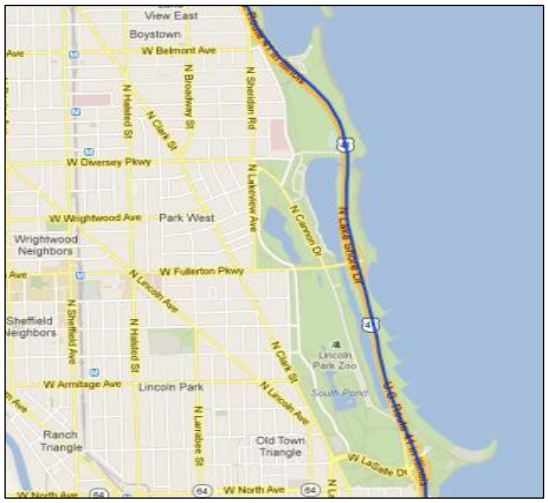

Figure 7 (b) Zoom of the derived route after step 5.

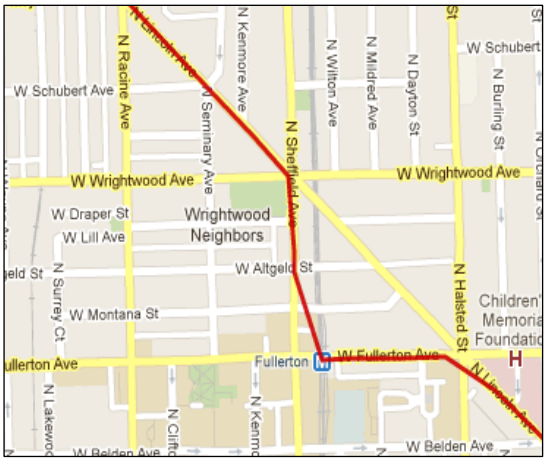

Figure 8- Derived route not aligned with the road's geometry

In Figure 8 it is evident that the derived route is not well aligned with the road's geometry at the intersection of W. Fullerton Ave and N. Sheffield Ave. Using map matching is useful in these cases to improve road and route geometry alignment.

6.2 Discussion on transit route derivation

The proposed route derivation algorithm can derive accurate route geometries from probe points as shown in Figure 7a and 7b. Additionally, Figure 9 (a, b) shows a derived route by the proposed algorithm and the ground truth of the route, as obtained from the agency. Using this algorithm, we derived the entire (over 100) routes of the Chicago Transit Authority.

In most of the cases, the road's and route's geometry are well aligned. In few cases, the derived route may not be well aligned with the road's geometry as shown in Figure 8 at Fullerton and N. Sheffield Intersection. Using the sixth step of the algorithm, map matching techniques should be considered to further align the route's and road's geometry.

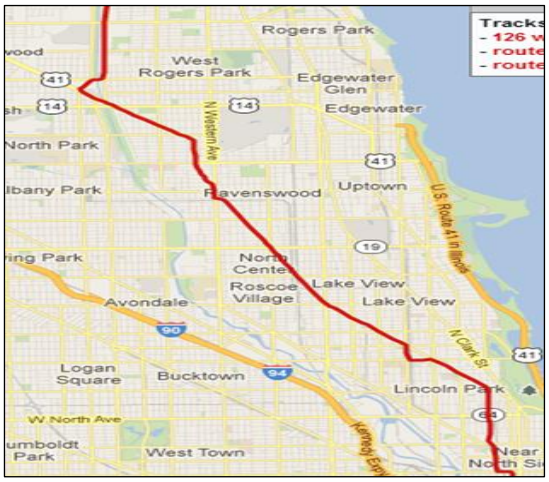

Figure 9 (a) – A derived route

Figure 9 (b) – Ground truth from the agency

7. Service schedule derivation

Service schedule derivation is the process of converting the raw GPS traces into the buses' service schedules for each bus stop. Service schedule derivation is important for transit agencies to update their service schedules. Often time buses deviate from their true schedules due to weather or traffic and the historical schedule needs to be re-computed to make it more accurate.

Given a set of bus stops $s_1, s_2, s_3, \dots s_n$ and a set of service trips $t_1^{di}, t_2^{di}, t_3^{di}, \dots t_m^{di}$ on an arbitrary day di along a route rt as obtained from the collected GPS traces such that any t_j^{di} contains a sequence of arrival times for each bus stop s_k. Each trip t_j^{di} maintains a set of arrival times $\{a_1^j, a_2^j, a_3^j, \dots a_n^j\}$ and each instance of the arrival time for bus stop s_p is represented by a_p^j. In general, from the sets of trips $\{t_1^{d1}, t_2^{d1}, t_3^{d1}, \dots t_m^{d1}\}$, $\{t_1^{d2}, t_2^{d2}, t_3^{d2}, \dots t_m^{d2}\}$, $\{t_1^{d3}, t_2^{d3}, t_3^{d3}, \dots t_m^{d3}\}$,…, $\{t_1^{d|d|}, t_2^{d|d|}, t_3^{d|d|}, \dots t_m^{d|d|}\}$ we derived two sets of schedules (i.e. weekday and weekend) for each s_k. $|d|$ is the number of days that we collected GPS traces. For each stop s_k along rt, the set of derived schedules $\{sch_1^{s_k}, sch_2^{s_k}, sch_3^{s_k}, \dots, sch_m^{s_k}\}$ is represented as $sch_{week-end}$ or $sch_{week-day}$. Finally, the entire service schedule is $\{sch_1^{s_1}, sch_2^{s_1}, sch_3^{s_1}, \dots sch_m^{s_1}\}$, $\{sch_1^{s_2}, sch_2^{s_2}, sch_3^{s_2}, \dots sch_m^{s_2}\}$, … $\{sch_1^{s_n}, sch_2^{s_n}, sch_3^{s_n}, \dots sch_m^{s_n}\}$ across all the routes in the transit service area.

Figure 10a shows the temporal distribution property of a fraction of GPS reports at a candidate bus stop over several days. The y-axis is a temporal property of the GPS report. From these GPS reports, it is evident that for each day buses arrive at different times at the bus stop. Also, there is no service between 1:30 and 4:00am.

For service schedule derivation, temporal clustering is performed on GPS reports that are associated with each candidate bus stop. The algorithm for service schedule derivation for a given bus is a simple two step algorithm. In the first step, the number of trips for the given bus stop is first computed. Let's call this T.

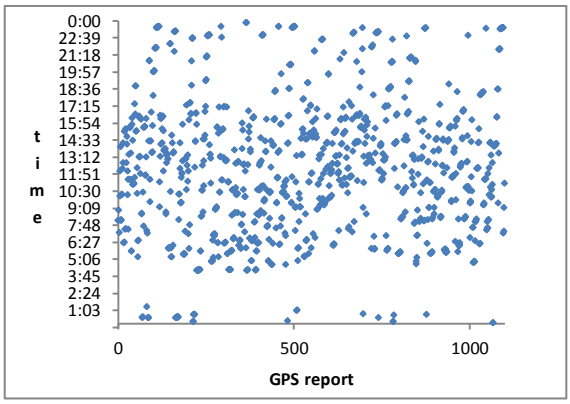

Figure 10(a) – Times of GPS reports at a bus stop

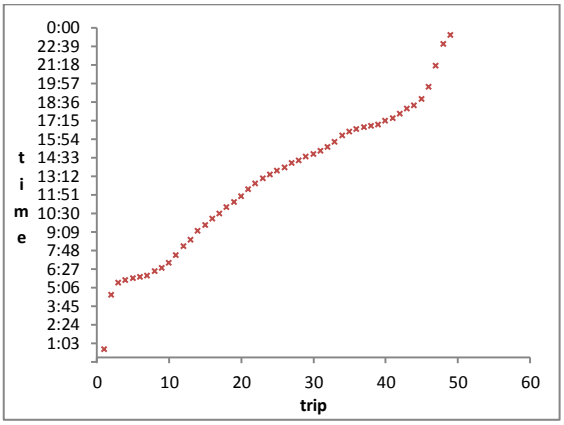

Figure 10(b) – Derived service schedule

Given T and the GPS reports that are associated with a particular bus stop, the algorithm then utilizes K-means clustering on the GPS reports' temporal properties. The value of K in K-means is equal to the number of trips (i.e. T).

Formerly, given a set of arrival or departure time observations ($\mathbf{a}_1$, $\mathbf{a}_2$, …, $\mathbf{a}_n$) for a single bus stop, where each observation is a *1-dimensional* real vector, representing the temporal property of the GPS reports that are associated with a bus stop, we utilize *K*-means unsupervised learning scheme to partition the *n* GPS observations into *K* sets. Each of the derived *K* clusters has information about the *K* service schedules for all trips of the candidate bus stop. We determine that a GPS report is related to s_i if it was submitted when the bus approaches or departs from s_i using a threshold *thres* with a route constraint. Recall, the bus stop locations are known from the bus stop derivation and buses report their true locations as they travel. The system parameter *thres* is a distance measure utilized to indicate when a bus approaches or departs from a bus stop. A subtle value, approximately 200m can be used.

For each stop s_k, the two inputs to K-means is determined. The two inputs are: (1) the number of service trips (i.e. L and L=K) and (2) the temporal property of the GPS reports associated with s_k. For s_k, the *K* clusters that are output by K-means maintains the properties of its *L* service schedules. More specifically, each cluster's mean is the derived service schedule (see Figure 10b). The variance of each cluster is an additional output of the algorithm and can be used as a confidence interval that improves the possibility of a traveler catching the desired bus. The derived transit service schedule, using the two stage algorithm, for the candidate bus stop in Figure 10a is shown in Figure 10b.

7.1 Service schedule evaluation and results

In this section, the effectiveness and efficiency of the schedules derived by the proposed algorithm is validated. For validation purposes, we simulate several people traveling and utilizing the service schedules that were produced. The evaluation algorithm then measures the percentage of time by which the travelers catch or miss their desired bus. Let's call this percentage value *TP*.

Using the schedule produced for an arbitrary bus stop along an arbitrary route, we then simulate travelers adhering to the derived schedule and proceeding to the bus stop at about 3 minutes before the scheduled arrival time. For example, if one of the derived schedules for a bus stop s_k along route W4 is 10:15am, we simulate a traveler arriving at the bus stop 10:12am. This simulation is realistic, because if one wishes to board a bus then they normally arrive a few minutes before they expect the bus to arrive. They (i.e. the traveler) also have an upper bound on the wait time. The upper bound wait time is the standard deviation for that bus's arrival time as computed by the proposed K-means strategy for each cluster.

If the bus arrives earlier than 3 minutes before the schedule or the traveler arrives later than the upper bound wait time, the traveler misses the bus. Otherwise, the traveler catches/boards the bus. The *TP* percentage value representing the probability of catching a bus is then computed.

Using this strategy of simulating multiple people adhering to a derived service schedule and attempting to catch a specific bus while we know the real time locations of the buses and when they arrive at bus stops [23], we found that the probability of catching a bus varies. While running several hundred experiments for arbitrary bus stops along five arbitrary routes, we found that the probability of catching a bus varies with the route. More specifically, for routes 12, 18, 60, and 8 we observed a probability of over 90%. For route 50, we get probability of over 82%. The schedule detection accuracy for 50 is lower than the others, we found this to be due to re-routes on route 50 during the validation phase. Re-routes causes buses to divert from their intended service schedule.

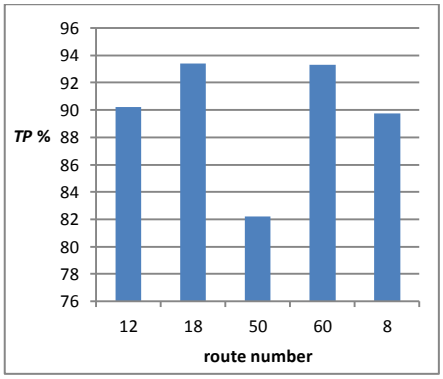

Figure 11 – Evaluation of service schedule derivation algorithm

8. Related work

The state of the art for transit system derivation is [2]. In [2], for bus stop detection the authors achieved 50% precision. Thus, this newly proposed algorithm achieves over 30% increase in the accuracy of bus stop detection. In [2], for bus stop detection, cells with density above the 80th percent quartile of non-zero density cell is considered as a potential stop. Then, using standard image processing technique the final decision, if whether or not a cell is a stop, is made. This proposed algorithm by us did not consider image processing; we used supervised learning via a density distribution histogram. Likewise, the route extraction in [2] is also different from the proposed six step algorithm. Another difference is that in this proposed work by us, for route derivation, Douglas-Peucker's [11] algorithm is not considered, nor is kernel density considered.

In [12, 13], assumptions are made that the transit agency is not willing to install tracking devices on their buses. Instead, tracking is done by crowd-sourcing through participatory sensing via traveler check-in. This implies that a traveler allows the system to track their mobile device while travelling. They also assume that the traveler is willing to report the location of the bus stops etc. While this work is interesting, it is different from our work in a number of ways, from the initial assumption to the idea of commuter check-in and crowd-sourcing.

There is a plethora of work that handles road map derivation from GPS traces. These schemes can be divided into three general categories: (1) *K-means* – Constructs road map using a series of cluster seeds [11, 14, 15, 16], (2) *trace aggregation* – Aggregates similar GPS traces based on location and bearing [17, 18], and (3) *kernel density estimation* - Which first computes a kernel density estimate of the raw GPS traces and then use image processing [2, 19, 20]. The proposed approach for route derivation is different and does not consider K-means, bearings, or kernel density estimation with imaging techniques. Some of these papers on road map generation take a more minute approach to map inference. For example, [14, 21, 22] place heavy emphasis on deriving individual lanes, lane splitting geometries, lane merging geometries, intersection geometries, and segment intersection boundaries. Currently, simple transit routes do not require such advanced modeling of road features. On the other hand, if these

features are desired in the future, the technology exists to support them.

There are works that aim to predict the real time arrival of buses at a given bus stop [24, 25]. This proposed work focuses on deriving the static service schedule for a transit agency. Real time bus arrival time prediction is a subject of our future work.

9. Conclusion and future work

This paper proposes algorithms that enable a transit system to be derived from raw GPS data. First, the transit system's buses are tracked using on-board GPS or smartphone localization technology. Given this location tracking data, the proposed algorithms can automatically derive the location of the bus stops, route geometries (i.e. transit path within the map), and service schedules. This requires no manual input or field visits.

More specifically, using a newly proposed density distribution histogram with ten bins and other temporal features, the proposed scheme captures the mobility patterns of the buses at bus stops, stop lights, and stop signs. We included stop sign and stop light locations into the model since these objects look deceivingly similar to bus stops in probe data. Bus stop locations can be detected with over 89% accuracy in this work. Using the public, transit drivers, or transit riders, one can use crowd-source techniques to address the misclassifications. This 89% accuracy is a 30% improvement on the prior art on bus stop detection from vehicle probes [2] which achieves only 50% precision.

Additionally, a route derivation algorithm that uses spatial clustering, spatial outlier pruning, cluster aggregation, and spatial and temporal ordering to derive accurate transit route geometries is introduced. The finding is that the derived route geometries are accurate and align with the road's geometry.

Further, using a temporal clustering scheme on the time component of the GPS data, effective service schedules of the transits are derived. Experimental analysis provides evidence that travelers can board the desired bus with high probability using the derived service schedules. This probability varies across routes.

In general, the algorithms proposed can be utilized and save transit agencies millions of dollars. There is a high cost for transit information systems. As mentioned, one bus agency budgeted over USD $20M for a transit information system [2]. Transit agencies in developing countries or emerging transit markets may not be able to afford such costly technology. Thus, using the proposed technology in this paper, the transit agencies can derive or update their transit information systems (i.e. stop locations, route geometries, service schedules) automatically from raw GPS traces. The proposed technology can be used by emerging transit agencies to initially determine stop locations and route geometries. Furthermore, mature transit agencies that already have knowledge of these transit artifacts can utilize the proposed technology to monitor and update their transit systems more effectively. Real time transit arrival time prediction [24, 25,] and evaluation of the current algorithms on other transit systems such as those of London and New York are subjects of future work.

10. References

[1] NextBus Website *http://www.nextbus.com*

[2] J. Biagioni, T. Gerlich, T. Merrifield, J. Eriksson. EasyTracker: Automated Transit Tracking, Mapping, and Arrival Time Prediction Using Smartphones. *ACM SenSys*, 2011.

[3] Clever Devices website *http://www.cleverdevices.com*

[4] Chicago Transit Authority website *www.chicagotransitauthority.com*

[5] Chicago Transit Authority Bus tracker *http://www.ctabustracker.com*

[6] I. Witten and E. Frank. Data Mining: Practical machine learning tools and techniques. *Morgan and Kaufmann*, San Francisco, 2005

[7] I. Guyon and A. Elisseef. An Introduction to Variable and Feature Selection. *Journal of Machine Learning Research 3*, 2003

[8] J. Zhou, R. Golledge. A Three-step General Map Matching Method in the GIS Environment. *Journal of Geographical Information Systems*, 2006

[9] H. Yin, O. Wolfson. A weight-based map matching method in moving objects databases. *16th SSDBM*, 2004

[10] P. Newson, J. Krumm. Hidden Markov map matching through noise and sparseness. *ACM SIGPATIAL GIS, 2009*

[11] D. Douglas and T. Peucker. Algorithms for the Reduction of the Number of Points Required to Represent a Line or Its Caricature. *The Canadian Cartographer*, (2), 1973

[12] A. Thiagarajan, J. Biagioni, T. Gerlich, and J. Eriksson. Cooperative transit tracking using GPS-enabled smartphones. *ACM SenSys*, 2010

[13] J. Biagioni, A. Agresta, T. Gerlich, and J. Eriksson. TransitGenie: A real-time, context-aware transit navigator (demo abstract). *ACM SenSys*, 2009

[14] S. Edelkamp and S. Schrodl. Route planning and map inference with global positioning traces. In R. Klein, H.-W. Six, and L. Wegner, editors, Computer Science in Perspective, volume 2598 of *Lecture Notes in Comp. Science*, 2003

[15] S. Worrall and E. Nebot. Automated process for generating digitized maps through GPS data compression. In *Australasian Conference on Robotics and Automation*, 2007

[16] G. Agamennoni, J. Nieto, and E. Nebot. Robust inference of principal road paths for intelligent transportation systems. *IEEE Transactions on Intelligent Transporation. Systems*, 2011

[17] L. Cao and J. Krumm. From GPS traces to a routable road map. *ACM GIS*, 2009

[18] B. Niehoefer, R. Burda, C. Wietfeld, F. Bauer, and O. Lueert. Gps community map generation for enhanced routing methods based on trace-collection by mobile phones. *In Advances in Satellite and Space Communications*, 2009

[19] J. J. Davies, A. R. Beresford, and A. Hopper. Scalable, distributed real-time map generation. *IEEE Pervasive Computing*, 2006

[20] W. Shi, S. Shen, and Y. Liu. Automatic generation of road network map from massive GPS, vehicle trajectories. In *Intelligent Transportation Systems*, 2009

[21] A. Fathi and J. Krumm. Detecting road intersections from GPS traces. *GIScience*, 2010

[22] Y. Chen and J. Krumm. Probabilistic modeling of traffic lanes from GPS traces. *ACM GIS*, 2010

[23] Chicago Transit Authority bus stop API: *http://www.ctabustracker.com/bustime/map/getStopPredictions.jsp?eta=true&route=all&stop=11575&key=0.10679383956930077*

[24] Y. D. S.I.J Chin and C. Wei. Dynamic bus arrival time prediction with artificial neural networks. *Journal of Transportation Engineering*, 2002

[25] Y. Bin, Y. Zhongzhen, and Y. Baozhen. Bus arrival time prediction using support vector machines. *Journal of Intelligent Transportation Systems*, 2006

[26] L. Modica and L. Stenneth. Method and apparatus for transit mapping (Nokia patent). Case number NC79749, Aug. 2012

[27] L. Modica and L. Stenneth. Method and apparatus for deriving spatial properties of bus stops and traffic controls (Nokia patent). Case number NC79747, Aug. 2012

[28] L. Modica and L. Stenneth. Method and apparatus for transit mapping (Nokia patent). Case number NC79746, Aug. 2012