Understanding Bus Travel Time Variation Using AVL Data

by

David G. Gerstle

B.S., Yale University (2009)

Submitted to the Department of Civil and Environmental Engineering
in partial fulfillment of the requirements for the degree of

Master of Science in Transportation

at the

MASSACHUSETTS INSTITUTE OF TECHNOLOGY

February 2012

Author .....
Department of Civil and Environmental Engineering
January 20, 2012

Certified by .....
Marta C. González
Assistant Professor of Civil and Environmental Engineering
Thesis Supervisor

Accepted by .....
Heidi M. Nepf
Chair, Departmental Committee for Graduate Students

Understanding Bus Travel Time Variation Using AVL Data

by

David G. Gerstle

Submitted to the Department of Civil and Environmental Engineering
on January 20, 2012, in partial fulfillment of the
requirements for the degree of
Master of Science in Transportation

Abstract

The benefits of bus automatic vehicle location (AVL) data are well documented (*see e.g.*, Furth et al. (2006)), ranging from passenger-facing applications that predict bus arrival times to service-provider-facing applications that monitor network performance and diagnose performance failures. However, most other researchers' analyses tend to use data that they acquired through negotiations with transit agencies, adding a variable cost of time both to the transit agencies and to researchers. Further, conventional wisdom is that simple vehicle location trajectories are not suitable for evaluating bus performance (Furth et al. 2006). In this research, I use data that are free and open to the public. This access enables researchers and the general public to explore bus position traces.

The research objective of this Master's Thesis is to build a computational system that can *robustly* evaluate bus performance across a wide range of bus systems under the hypothesis that a comparative approach could be *fruitful* for both retrospective and real-time analysis. This research is possible because a large number of bus providers have made their bus position, or AVL, data openly available. This research thus demonstrates the value of open AVL data, brings understanding to the limits of AVL data, evaluates bus performance using open data, and presents novel techniques for understanding variations in bus travel time. Specifically, this thesis demonstrates research to make the system architecture *robust* and *fruitful*:

- This thesis explores the exceptions in the various datasets to which the system must be robust. As academics and general public look to exploit these data, this research seeks to elucidate important considerations for and limitations of the data.

- Bus data are high-dimensional; this research strives to make them dually digestible and informative when drawing conclusions across a long timescale.

Thus, this research both lays the foundation for a broader research program and finds more visually striking and fundamentally valuable statistics for understanding variability in bus travel times.

Thesis Supervisor: Marta C. González
Title: Assistant Professor of Civil and Environmental Engineering

Acknowledgments

Thank you to my collaborators Marta González, Dietmar Bauer, and Peter Widhalm. Here it is. This work was partially supported by the Austrian Institute of Technology.

Thank you to the MBTA and the Transit Research Group for your insight and patience.

Thank you to my friends, lab-mates, the rest of my MST class, and the welcoming people of AIT for your kindness and friendship; especially Nick, who stayed up all night with me on gchat many times as I finished.

Thank you to my mother, father, and sister for your love and support; especially my mother, who gave thoughtful edits and feedback.

Thank you especially to Turner, Milo, and Fyodor.

Contents

List of Figures

List of Tables

List of Algorithms

Chapter 1

Introduction

The benefits of bus automatic vehicle location (AVL) data are well documented (*see e.g.*, Furth et al. (2006)), ranging from passenger-facing applications that predict bus arrival times to service-provider-facing applications that monitor network performance and diagnose performance failures. However, most other researchers' analyses tend to use data that they acquired through negotiations with transit agencies, adding a variable cost of time both to the transit agencies and to researchers. Further, conventional wisdom is that simple vehicle location trajectories are not suitable for evaluating bus performance (Furth et al. 2006). In this research, I use data that are free and open to the public. This access enables researchers and the general public to explore bus position traces.

The research objective of this Master's Thesis is to build a computational system that can *robustly* evaluate bus performance across a wide range of bus systems under the hypothesis that a comparative approach could be *fruitful* for both retrospective and real-time analysis. This research is possible because a large number of bus providers have made their bus position, or AVL, data openly available. This research thus demonstrates the value of open AVL data, brings understanding to the limits of AVL data, evaluates bus performance using open data, and presents novel techniques for understanding variations in bus travel time. Specifically, this thesis demonstrates research to make the system architecture *robust* and *fruitful*:

- This thesis explores the exceptions in the various datasets to which the system must be robust. As academics and general public look to exploit these data, this research seeks to elucidate important considerations for and limitations of the data.

- Bus data are high-dimensional; this research strives to make them dually digestible and informative when drawing conclusions across a long timescale.

Thus, this research both lays the foundation for a broader research program and finds more visually striking and fundamentally valuable statistics for understanding variability in bus travel times.

This thesis exploits transit agencies publicly available General Transit Feed Specification (GTFS) data and their real-time AVL eXtensible Mark-up Language (XML) feeds of bus Global Positioning System (GPS) coordinates. Here I specifically focus on the Massachusetts Bay Transit Authority (MBTA) bus network centered around Boston, MA using data from May 2011.

The computational architecture proposed here takes the data through many steps. Each step is designed to accommodate and correct for any biases introduced along the way. Figure 1-1 conceptually depicts the evolution and iterations of the data. Many of the iterations occur before before I have access to them, represented by the boxes reality, observation, and primary data. I am able to process and analyze the data from inside of the AVL analysis box. Understanding the errors and biases introduced by those iterations is essential to effectively projecting the AVL data onto the GTFS data.

Once the data are accessible, my computational architecture robustly transforms them into a form where they can be analyzed. Figure 1-2 shows how my system architecture interacts with various databases and datasets through computer algorithms. There, databases are represented by cylinders, the programs and programming languages used to transfer and transform the data, and the arrows show the mechanism for moving data around. Generally, tables and fields from a database are referred to using `fixed-width` text, and variables and functions from a program are referred to

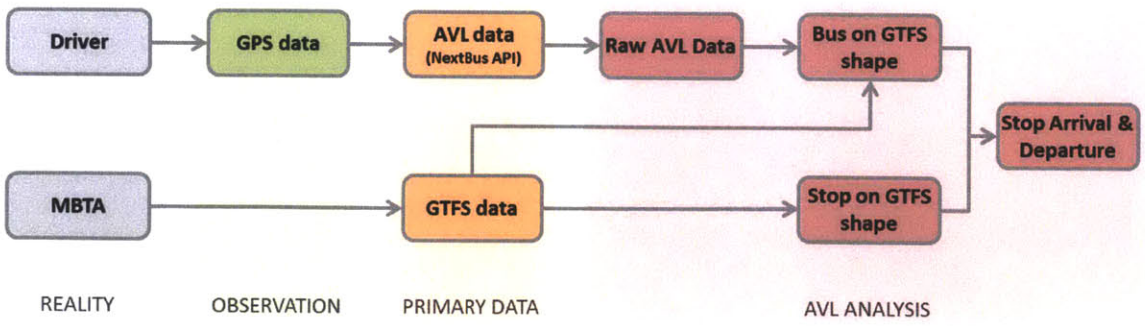

Figure 1-1: Schematic diagram of how the bus location data are generated and analyzed.

using sans-serif text.

Both Figures 1-1 and 1-2 depict the projection of the AVL data onto the geometric GTFS data, a procedure known as map-matching. After performing map-matching of real-time coordinates to the shapes of the road network, I can analyze travel time dynamics and, more importantly, estimate the time when each bus arrives and departs to and from each stop. I characterize disruptions at the bus level by comparing the statistics of travel times between stops, allowing us to detect the origin of delays in space and time and the emergence of bus bunching. This architecture provides a spatiotemporal analysis of the bus system performance.

Chapter 2 describes the dynamic bus location data and the static bus schedule data and places the bus location data within the broader context of using location data to study human mobility, and argues that it is necessary to also use the schedule data. Next, Chapter 3 presents my robust system architecture for transforming the static and dynamic bus data into a form that can be analyzed, especially my map-matching and stop arrival and departure interpolation methods. Then, Chapter 4 presents analysis and visualization of the bus data. Finally, Chapter 5 summarizes the thesis' findings and offers suggestions for further research.

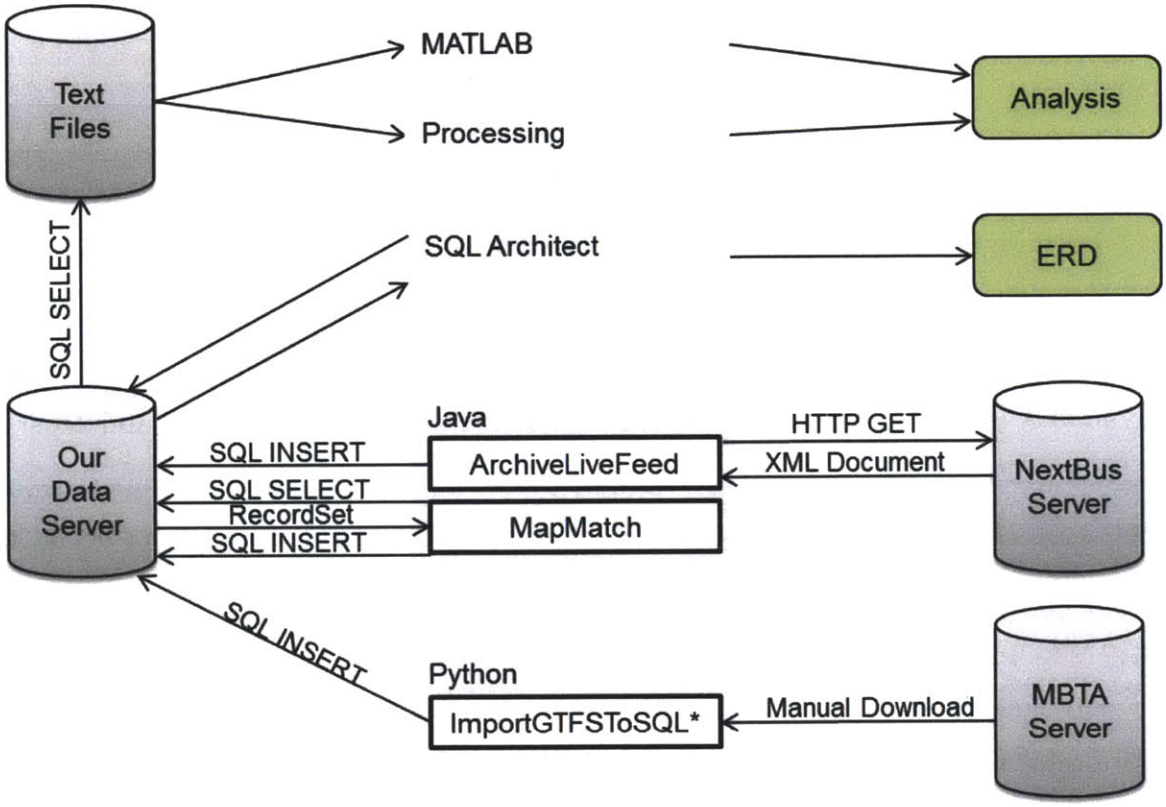

Figure 1-2: Schematic diagram of how the bus location information move between servers and computer programs. The ImportGTFSToSQL program was written by Bick and Damphouse (2010).

Chapter 2

Data

The central dataset of this research is the AVL data for the MBTA provided by NextBus. This Chapter introduces the NextBus MBTA AVL data not only to inform the below analyses but also to place the data in the broad context of human mobility research. As with any research, the research questions that can prudently be answered here are a function of the available data. As a number of scientists with backgrounds in other fields move toward researching human mobility, it is important to condense and codify the domain-specific knowledge necessary to make intelligent decisions about what different data can be used for and where they fall in the broad context of human mobility research.

Ultimately, AVL data are spatiotemporal tracing data. There is a wide range of research, summarized below, that uses trajectory data to different ends as their specifics allow. There are broadly two types of bus AVL data: location-at-time and time-at-location data (Furth et al. 2006). Location-at-time data are observations, usually collected using Global Positioning Satellite (GPS), of where the bus is at a particular time. They tend to give a latitude-longitude coordinate and a timestamp as a record on the bus' trajectory. Additionally, these data tend to be collected periodically, such as observing the bus' position every three seconds. Time-at-location data are the inverse. Time-at-location data are observations of when the bus passes a certain location. These data could also be recorded as a latitude-longitude coordinate and a timestamp, but the timestamps are not designed to be evenly spaced and the

coordinate likely identifies an important place, such as a bus stop or time point. So, time-at-location data are more likely to provide a location identifier and timestamp.

Location-at-time data exist in an abstract trajectory space whereas time-at-location data exist in human space. By trajectory space, I mean the mathematical three-dimensional space of latitude, longitude, and time. By human space, I mean roads, addresses, bus stops, and bus stations. More specifically, for buses, the human space is usually defined using General Transit Feed Specification (GTFS), which describes route and schedule definitions and is robust enough to describe bus, rail, gondola, and other transit systems. Obviously, trajectory space and human space are related. Map-matching is the process of projecting trajectory space onto human space. Geocoding is the process of projecting a street address in human space onto trajectory space.

The NextBus MBTA AVL data are location-at-time data, and the buses are polled for their positions every 60 seconds. Thus, the NextBus MBTA AVL data are native to the abstract trajectory space. However, as is shown in Section 2.1, the NextBus MBTA AVL data are not appropriate for the kinds of analyses generally done with data in trajectory space, and, rather, need to be projected onto human space, specifically GTFS space. With the scope of the research established, Section 2.2 introduces the GTFS data specifically and elaborates how I accessed them. Next, Section 2.3 discusses the NextBus MBTA AVL data and the data collection process for the NextBus data. Finally, Section 2.4 summarizes the content of this Chapter.

2.1 Use of Position Tracking Data

Human mobility is about when where, why, and how people move from one place to another. Unsurprisingly, location tracking data are the key data for most human mobility studies. Again, AVL stands for 'automatic vehicle location;' it is not lexicographically specific to buses. The amount of information AVL data carry is dependent on what kind of vehicle is being located.

This Section begins by exploring the use of location-at-time data in trajectory

space by imagining the 'vehicle' as a person or personal vehicle, ultimately finding that the NextBus MBTA AVL data are not appropriate for this use because they do not directly observe the movement of people. That is, Section 2.1.1 shows that the NextBus MBTA AVL data cannot be used for those purposes specifically because they are bus data. Thus, the location-at-time data need to be projected into human space. Section 2.1.2 shows that both the buses being buses and the polling time (60 seconds) of the NextBus MBTA AVL data prevent the data from being used for the traditional uses for data in human space. Ultimately, I argue that the best use for the NextBus MBTA AVL location-at-time trajectory space data is analyzing bus network performance in the context of the GTFS-defined human space.

2.1.1 Tracking a Person or Personal Vehicle

This Section shows that location-at-time data that directly observe people have been productively analyzed in trajectory space. That is, person and personal vehicle location tracking data provide for many different kinds of analyses of human mobility, and that those analyses do not necessarily require any information about the road network or other geographical information.

Within the context of a personal vehicle, such as a non-commercial automobile or bicycle, understanding where the *vehicle* is going is roughly equivalent to understanding where at least one *person* is going:

$$[\text{Where is this car going?}] \approx [\text{Where is this person going?}]$$

So, from the perspective of understanding human mobility, non-commercial automobile AVL data are valuable. However, they are only valuable if complete—if the entire extent of the trajectory is observed. This is because the trajectory is, at most, confined to the road network; these veicles are not running on any available schedule. For example, with half-complete AVL trajectories, I can only predict the remainder of the trip to accuracy proportional to the half-trip distance, subject to network topology (prediction in a rural area with very few roads would be better). Answering either

[Where is the vehicle going?] or [Where is the person going?] is a complex, if possible, task with partial AVL information when the vehicle is a personal vehicle.

AVL data for a personal vehicle capture direct observations of a human's mobility. Many researchers use these kinds of data, both those generated from observing a vehicle and from observing a person individually. The data are used to approach questions like: how far do people usually travel; what is a specific person's normal mobility routine; how predictable are people's mobility patterns; and what is the aggregate origin-destination (OD) matrix of a city? González et al. (2008) infer trip lengths and origin-destination predictability from location data generated by cell phone use histories. Song et al. (2010) looks at the distributions of travel distances, work that could have implications in activity-based transportation modeling. Trasarti et al. (2010) introduce a software package for visualizing GPS trajectories and calculating O-D matrices. Hariharan and Toyama (2004) use personal GPS trajectories to extract 'destinations' and destination transition probabilities from GPS trajectories. For Hariharan and Toyama (2004), a 'destination' is an important location such as home or work. Their research require large amounts of data for a single person and thus accordingly a stable unique identifier for each person who generates a location history. Andrienko et al. (2007) expand upon that work by adding some visualization tools and identifying common trips by algorithmically evaluating route similarity. They prefer the term 'significant places' over 'destinations.' Liao et al. (2007a) and Liao et al. (2007b) also explore algorithms for extracting significant destinations or places and activities, routes, or routines from GPS trajectories using hierarchically structured conditional random fields and Rao-Blackwellized particle filters, respectively.

Attacking different problems requires different levels of pre-processing. Interestingly, these research programs can take place in the abstract trajectory space. Location tracking data only include latitude-longitude coordinates and timestamps; they are a point in space and time. Yet, for many analyses, existence in abstract $\mathbb{R}^2$ is sufficient.

For example, calculating trip length does not require projection to human space.

For a course-grained measurement, evaluating trip length requires identifying the beginning and end of a trip (a non-trivial task in its own right), and calculating the geodetic distance between them. Of course, that calculation is likely to underestimate the actual trip length. Roads and trails are unlikely to provide a straight-line path from origin to destination. In the case of González et al. (2008), locations are approximated using the Voronoi cells defined by cellular phone signal towers. For them, the spatial fidelity is at best hundreds of meters, so trying to get finer estimates on the distance traveled is impossible.

Provided that there is at least one location data point between the origin and destination, trip lengths can be calculated more accurately. Making the calculation one level more detailed, researchers can sum the distance between intermediate location points from origin to destination. This approach would theoretically perfectly capture the contours and thus length of the trip. However, even GPS measurements are not perfect and can have 1 m to 40 m of uncertainty (Quddus et al. 2007). Accordingly, the GPS traces may not perfectly fit to the road taken. Calculating the distance traveled as dictated by the road network requires inferring the position along the road network that each GPS coordinate represents. Map-matching is this methodology for projecting a location trajectory onto the road network.

Ultimately, for many applications, studies using direct observations of human mobility do not require map-matching. Granted, the trajectories are often visualized by superimposing them on a map, but the data structures of the trajectory and human spaces are not more deeply linked. The patterns that those researchers are looking for are geometric patterns, not geographic patterns.

Observing personal vehicles or people themselves directly leads to a wide range of research opportunities. Those data typically do not require map-matching for analytic purposes. These data naturally lend themselves to studying human mobility. Unfortunately, there are important differences between these data and bus AVL data that prevent bus AVL data from directly informing human mobility.

2.1.2 Tracking a Bus

Now, shifting the definition of 'vehicle' in AVL back to a bus operating a scheduled service, the story is much different. 'Scheduled service' is in contrast to *ad hoc* service, not in contrast to high-frequency service. Bus location trajectories do not directly capture human mobility; rather, they observe the state of the transportation infrastructure, where the bus service itself (and its relative timeliness, etc.) are a part of the infrastructure. Like other studies that analyze transportation infrastructure, bus location data need to be map-matched. The precision of map-matching is limited by the polling time of location-at-time data. Ultimately, this Section demonstrates that the NextBus MBTA AVL data, because they are bus data, are most useful in human, not trajectory, space and that, because of the slow polling time, they are most prudently projected specifically onto the GTFS human space.

Bus location data are very different from personal and personal vehicle trajectory data. Here, the vehicle operator, through the lens of human mobility, is not a person. They are not traveling for any purpose other than employment itself; they do not have an origin and destination. Thus, if the bus is empty, it is not mobilizing any humans. The position of an empty bus offers no insights into human mobility except in that the bus is occupying road space (thus potentially affecting other travelers) and to any regard emptiness is interesting. It is not answering when, where, or why people are moving, and only vaguely how: not by bus. Accordingly,

$$[\text{Where is the bus going?}] \neq [\text{Where is this person going?}]$$

From the perspective of understanding human mobility, scheduled bus AVL data alone are not valuable. However, in contrast to automobile AVL data, partial information about where the bus is going is valuable. Much of a bus' public value come from the fact that it is scheduled to perform a predefined route at a predefined time; where the vehicle is going is *a priori* predictable. Analogizing buses to automobiles, asking where a bus is going is like asking where the road on which a vehicle is traveling is going because both can be looked up. This predictability is undesirable for human

mobility research. The value of the research described in the previous Section was finding regularities in the seemingly unpredictable data. In this way, the bus is a part of the infrastructure, not just a beneficiary of the road infrastructure.

Once the bus is seen as a part of the infrastructure, it follows that passenger, not just bus, data are required to better understand human mobility. These data are also collected by many bus systems as automatic fare collection (AFC) or automatic passenger count (APC) data. AFC data come in various forms. At their most basic, AFC data just count that someone has boarded a bus through the fact of their paying a fare. With payment cards such as MBTA's CharlieCard, the data reflect that a specific person, as identified by their card, has boarded the bus (Shireman 2011). Except in the few systems, like London's Underground, where users must also "tap out" of the transit infrastructure (Brakewood 2010), AFC does not directly capture alightings, only boardings. APC data, in contrast, reflect both boarding and alighting. However, APC data are fully anonymous and are unable to track when a specific individual enters and exits the transit system (unless there is only one passenger). Obviously, the inclusion of APC/AFC data along with AVL data opens a wider range of accessible research questions. However, APC/AFC data tend not to be openly available to the public, and so are excluded from this analysis.

Absent APC/AFC data, AVL data only inform [Where is this bus going?] and cannot get at [Where is this person going?]. The AVL data are not deterministic. Still, they need to be understood within the framework of their schedule in both the spatial and temporal dimensions. Thus, the AVL data need to be mapped onto the schedule data, onto human space, and so GTFS data are necessary. This is an important distinction between direct measurements human mobility and measurements of the transportation infrastructure: measurements of the infrastructure need to be linked back to the human space from the trajectory space.

Now that I have established that NextBus MBTA AVL data should be analyzed in human space, I argue that they specifically should be analyzed in the GTFS human space. Ideally, the bus AVL data could be used as sensors of all levels of the transportation infrastructure that they use and are a part of. Here, I show that,

unfortunately, they can only be used as sensors of the bus network itself. To do so, I look at past research that uses vehicles as sensors of the transportation infrastructure, moving from the most to least detailed studies of that infrastructure and show that the bus AVL data are unsuitable for pursuing similar lines of study.

Some research teams have used vehicle location data to study traffic microstructure. Oehlerking (2011) use vehicle position data to calculate fuel consumption and thus CO_2 emissions. They show that accurately predicting fuel consumption requires location data at a period of once per second, though a period of once every 2-3 seconds can calculate consumption to an accuracy of 2-5%. High fidelity data are required because fuel consumption is heavily dependent on the vehicle's acceleration profile. Interestingly, this more abstract look at traffic microstructure does not require map-matching. However, map-matching also would not be useless: accurately matching the GPS points to the road could help to more accurately calculate the distances traveled between GPS points thus more accurately calculate the velocity and acceleration.

Though bus AVL data could record their data frequently enough to be used for these CO_2 calculations, many do not. For example, the NextBus MBTA data used as a primary source in this research are only recorded every 60 seconds, an order of magnitude off the requisite frequency. Interpolating the NextBus MBTA AVL data up to the requisite frequency of 1-3 seconds would not be helpful. Modeling the CO_2 depends on integrating over the fine changes in acceleration over 1-3 second periods. As the NextBus MBTA data are only collected every 60 seconds, that detailed information is lost and cannot be recovered. Further complicating the calculations, many city buses run on liquid natural gas (LNG) or overhead electrical lines, and those data are not generally available. The bus AVL data used here are not appropriate for studying traffic microstructure and CO_2 emissions.

Moving a level less detailed, most research in this category uses vehicles as sensors of the traffic on road segments, generally called 'floating car' studies. There are a large number of 'floating car' studies. Chen et al. (2007) use Beijing taxi data to study travel time reliability over various time periods and scales. De Fabritiis et al. (2008)

implement a large-scale floating car study on the highway network around Rome. He et al. (2008) and Van Zuylen et al. (2008) improve floating car data methodology by more effectively treating road segments with traffic signals. Hellinga et al. (2008) suggest a method for inferring the behavior of floating cars between their consecutive polled positions. Toplak et al. (2010) propose a method for using historical floating car data and road segment similarity statistics to fill in traffic predictions for segments where data are very sparse. ITS Vienna Region (2012) uses real-time floating car data for multimodal trip planning. Dong and Pentland (2009) use a large database of historical vehicle GPS tracks to develop a traffic congestion prediction model. Thiagarajan et al. (2009) take that research farther by developing a mobile phone application called VTrack that simultaneously provides map-matching and routing for the driver, identifies and reroutes the driver away from delay-prone segments, and minimizes the power drain on the mobile device. Cathey and Dailey (2001) use buses as a traffic sensor. However, Cathey and Dailey (2001) do not use GPS data; rather they use time-at-location data in the form of 'time point' data which record when a bus passes through sequential sensing gates in the road. They only study a bus route that travels on the interstate, so the bus travel times are unaffected by stopping for passenger boardings and alightings.

Floating car research does require map-matching. The purpose of floating car research is to report on and possibly predict the traffic state on such-and-such road segment. This pursuit lacks accuracy if the researchers cannot identify whether a vehicle is on that road segment or not.

Unfortunately, the NextBus MBTA AVL data are not suitable for use as floating car data either. As suggested by Cathey and Dailey (2001), AVL data is not appropriate to be used as floating car data for intracity driving. First, scheduled bus stops would need to be removed from the data, as they do not reflect traffic conditions. D'Acierno et al. (2009), for example, attempt to use bus AVL data as floating car data in urban Naples, Italy by separately treating regions of the road that are in the neighborhood of bus stops or traffic lights. However, their results are not conclusive and have not been broadly adopted. Further, the infrequent 60 second polling on

the MBTA data precludes them from such analyses. Even with perfect information about the bus' location, a city bus could not be used as a floating car. Some bus stops are co-located with stop lights, and there is no iron-clad way of knowing whether the length of a stop there is due to the traffic light or to serving passengers. Additionally, after making a stop, a bus must often merge back into traffic, again slowing its progress in a way not experienced by general traffic. Buses traveling in exclusive bus lanes are also not suitable for use as floating car data, as general traffic are explicitly excluded from those lanes. However, as practiced by Cathey and Dailey (2001), if buses have long stretches where they behave just like other vehicles without stopping for passengers, such as along highways, AVL data are appropriate. In sum, AVL data are not necessarily the best to be used as city driving floating car data.

Thus, the NextBus MBTA AVL data, despite being location-at-time data, cannot be used to directly study human mobility, traffic micro-structure, or traffic generally. However, the NextBus MBTA AVL data can be used to study the status of the bus itself as infrastructure. Consequently, this is where most of the research presented here is pointed, and so I will provide a more thorough literature review of this subject in Chapter 4. To study the bus infrastructure, I do need to map-match the bus AVL data from trajectory space to human space. As I am not studying traffic generally, I do not need to map-match them to the general road network, but rather only the bus network. These data are generally provided in the General Transit Feed Specification, the subject of the next Section.

2.2 General Transit Feed Specification

As argued above, the NextBus MBTA AVL data are best understood in the context of the MBTA GTFS data. Fittingly, I introduce the MBTA GTFS data first here. This Section covers the genesis and evolution of the GTFS and summarizes how to access and generally understand GTFS data.

The General Transit Feed Specification (GTFS) "defines a common format for public transportation schedules and associated geographic information"

(Google 2011). Simply, GTFS is a standardized machine-readable format for transit schedule data for multi-modal trip planning. It comprises of thirteen comma-separated values files that are described as a relational database. GTFS data, when complete, provide all of the data about planned bus service. They are the standard against which the AVL data can be judged once the AVL data are projected onto the GTFS data.

The inspiration for GTFS came from an entrepreneurial transit provider. GTFS began when Portland's TriMet's Bibiana McHugh approached Google in 2005 to create Google Transit for Google Maps (TriMet 2011). Thus, the original motivation was to put schedule data into a format that Google could use to provide potential passengers with schedule and trip-planing information. The specification was originally called the Google Transit Feed Specification, and after widespread acceptance, the 'G' was changed from Google to General (Google 2011). GTFS is now the *de facto* standard for schedule and route data (*see e.g.*, Raschke (2011)).

The GTFS is well-documented by Google (2011). However, that description is geared more toward transit providers to help them to put their schedule data into GTFS format, not toward bus service analysts. Here I summarize the procedure for accessing the MBTA data and key terms and tables in GTFS for analyzing bus performance both to make GTFS data more accessible to the reader and to introduce important terminology for the rest of the thesis.

Acquiring the GTFS data is easy. The MBTA publishes their GTFS data on their website and maintains an archive of old GTFS files. I downloaded the data directly from MBTA (2012). The collection of text-files is best leveraged as a SQL relational database. I chose to use the open-source PostgreSQL database platform, commonly referred to as Postgres (PostgreSQL 2011). Next, I used Bick and Damphouse (2010)'s Python-based tool to import the MBTA GTFS files into the Postgres database. Bick and Damphouse (2010)'s tool creates a database that conforms with an older form of the GTFS standard, so I needed to make some changes so that it would conform to the current standard set by Google (2011).

The GTFS itself elegantly represents transit schedule data. As the GTFS are a

relational database, it is useful to view the relationships between the files of GTFS. Chen (1976) introduced the entity-relationship diagram (ERD), a helpful and popular way of visualizing a relational database. Figure 2-1 depicts the full ERD of the GTFS, plus two additions. I built the ERD using the open-source Java-based tool SQL Power Architect (SQLPOWER 2011).

ERD notation conveys all of the characteristics of a database. Here, I introduce the ERD notation as implemented by SQLPOWER (2011). In this notation, each data table is represented as an individual rectangle. At the top of the rectangle the table name is listed in a gray background. For specificity, I focus on the table `gtf_stop_times`, depicted in the upper-left corner with the page rotated landscape with the ERD text right-side up. In that box, the table title `gtf_stop_times` is shown with a gray background at the top. The remainder of the rows in the box are the data table field names, which label all of the data that are included in each record of the table. Directly below the table title is the primary key, a collection of one or more fields which when specified together identify a unique record. In SQLPOWER (2011) notation, the fields which comprise the primary key are designated in two ways. First, they are listed between the title and the horizontal line at the top of the table box. Second, they have the notation [PK], short for primary key, or [PFK], short for primary and foreign key. In table `gtf_stop_times`, the primary key is composed of two fields, `trip_id` and `stop_sequence`.

A line connecting two tables indicates that they are related. For example, `gtf_stop_times` has relationships with the entities `gtf_trips`, `gtfs_pickup_dropoff_types`, and `gtf_stops`. These relationships are foreign key relationships, notated as [FK] or [PFK]. First, let's look at the relationship with `gtf_trips`. The relationship is identifying, as indicated by the solid as opposed to dashed line, meaning that the relationship is between fields that are part of their respective tables' primary keys. Further, the relationship shows that for each single `trip_id` in `gtf_trips`, there are zero, one, or many records in `gtf_stop_times` with that `trip_id`. For short, this is a one-to-many relationship. The one-to-many relationship is notated by the single hash on the line between them on the `gtf_trips` side, indicating the phrase 'for each

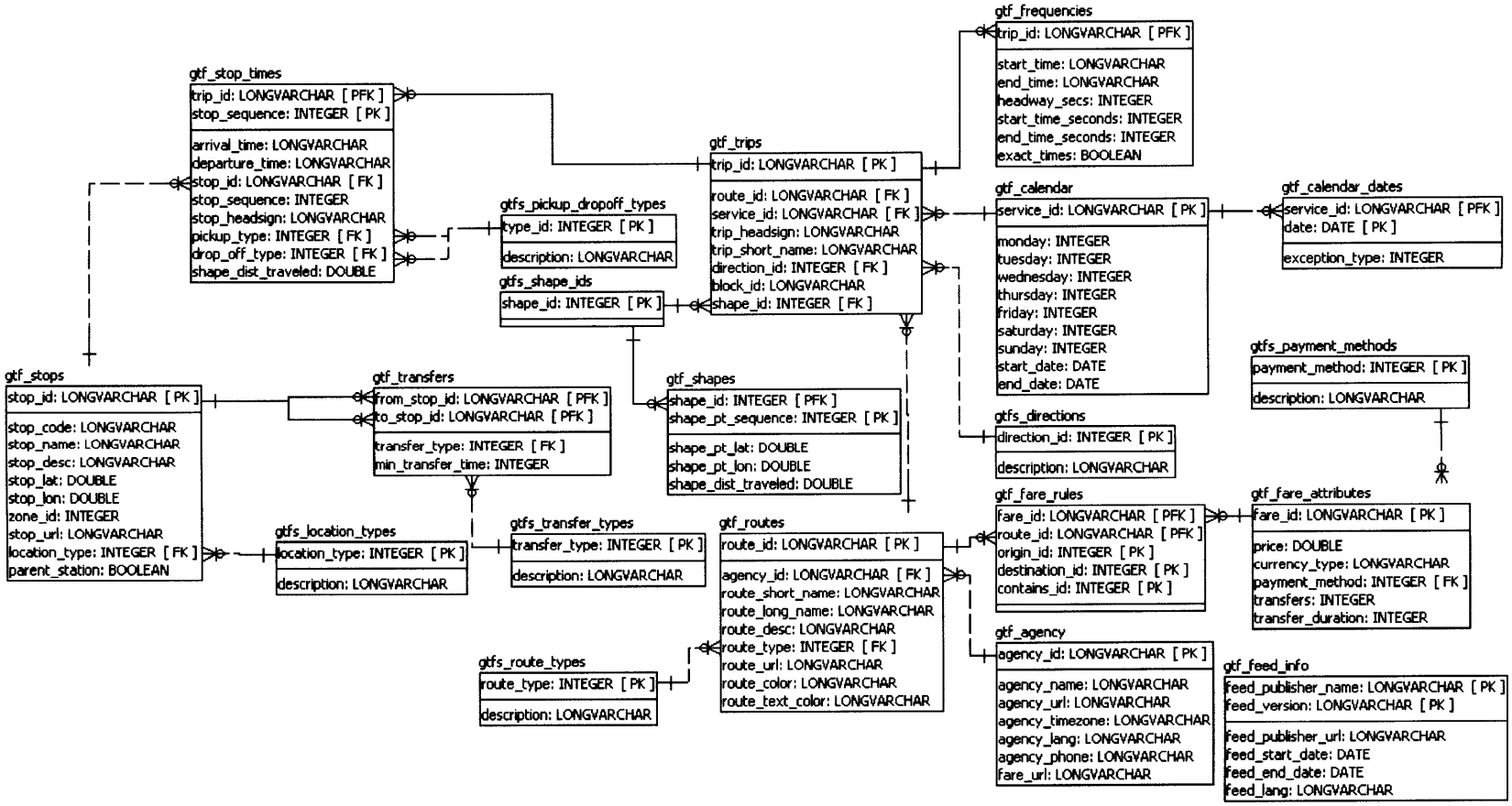

Figure 2-1: Entity-relationship diagram (ERD) of the GTFS files. Built using the open-source Java-based tool SQL Power Architect (SQLPOWER 2011). This reflects the GTFS according to Google (2011) per the 6 September 2011 revision. Additionally, the various lookup tables defined by the specification (*e.g.* `gtfs_payment_methods`) and also a table of only the unique shape ID's in order to represent the relationship between `gtf_trips` and `gtf_shapes`. `gtf_feed_info` does not have a relationship with the other tables.

single `trip_id`,' and the combination of an open circle or zero, a single hash, and the fanning hashes (sometimes called crow's feet), indicating the phrase 'zero, one, or many `trip_ids`,' respectively. The line between them give the phrase between of 'there is.' So, all together, the line between `gtf_stop_times` and `gtf_trips` indicates that 'for each single `trip_id` in `gtf_trips` there are zero, one, or many records with that `trip_id` in `gtf_stop_times`.'

There are two important additions to the ERD depicted in Figure 2-1 which are not part of the official GTFS:

1. The ERD includes the look-up tables for `pickup_dropoff_types`, `location_types`, `location_types`, `route_types`, `directions`, and `payment_methods`. The values for these look-up tables are defined by Google (2011), but they are not explicitly stated as additional tables.

2. I added the table `gtfs_shape_ids`. `gtfs_shape_ids` only contains the unique `shape_ids` so that the relationship between `gtf_trips` and `gtf_shapes` may be properly depicted. Otherwise, there is not a table in which `shape_id` alone is the primary key, which prevents it from having a well-defined foreign relationship.

Only a subset of the tables of the GTFS are relevant to analyzing bus performance. For my purposes, the trip is the most fundamental concept, so the `gtf_trips` table is the central table. A trip is one scheduled run of a bus along a route of a specific shape, also called a half-cycle. The unique `trip_ids` in `gtf_trips` can then be used to look up everything about that run:

- The time when the trip is scheduled to begin and make all of its stops are provided in `gtf_stop_times`. The `gtf_stop_times` table thus also lists all of the stops that the trip is to make with the `stop_id`. The location and name of each `stop_id` can then be looked up in the `gtf_stops` table. The MBTA uses `gtf_stop_times` for both 'scheduled trips' and 'frequent trips.' The 'frequent' trips `stop_ids` are defined in `gtf_stop_times`.

- 'Frequent' trips are reported in the `gtf_frequencies` table. These trips are scheduled to take place continuously during the period defined by `start_time`

and `end_time` with a target headway of `headway_secs`. The MBTA does not schedule many routes as 'frequent' service.

- The route that the trip is running. The route name and description can be looked up in the `gtf_routes` table using the `route_id`.

- The direction of the trip. For example, the routes of the MBTA are either inbound or outbound with respect to central Boston. The `direction_id` captures the trip direction as an integer whose value is reflected in the look-up table `gtfs_directions`.

- The days on which the trip is run as indicated by the service. `service_ids` are defined in `gtf_calendar`. `gtf_calendar` defines the range of dates and the days of the week that a particular `service_id` operates. The `service_id` is helpful for separating weekend and weekday service.

- The shape of the route that the bus is running. The `shape_id` identifies the shape of the route as defined in the table `gtf_shapes`. There, the shape is denoted as a poly-line of latitude-longitude coordinates as `shape_pt_lat` and `shape_pt_lon`, respectively. The order of the coordinates (thus defining the poly-line) are represented by the `shape_pt_sequence`. In the MBTA data, each `route_id` has at least two `shape_ids`, one for each direction of travel, but they can have as many as 20 `shape_ids` to capture the various large and small deviations necessary. The shape variations tend to be identified by sequential `shape_ids` that also encode the route. For example, the `shape_ids` for MBTA Route 1 are 10016 (inbound) and 10017 (outbound). There do not seem to be any other regular patterns for which `shape_id` is inbound or outbound, by odds/evens or otherwise, as the variations increase: for example, for Route 66 the seven variations are 660015 (inbound), 660017 (inbound), 660018 (outbound), 660019 (outbound), 660020 (outbound), 660021 (inbound), and 660022 (outbound). As the `shape_ids` capture the scheduled trajectory of the bus, the AVL data, when map-matched, are projected onto the appropri-

ate shapes. Interestingly, there is not a relationship between gtf_shapes and gtf_stops in the GTFS: the stop_lat, stop_lon coordinates also need to be map-matched to the appropriate shape_ids as dictated by gtf_stop_times and gtf_trips. However, the MBTA data do appear to use shape_pt_sequence to show part of the relationship between gtf_stop_times and gtf_shapes. The shape_pt_sequence progress, using Route 1's 10016 as an example, as

$$\{10001, 10002, 10003, ..., 10012, 20001, 20002, ...\}$$

Once the stop_ids are map-matched to the shape, I joined gtf_stops and gtf_shapes to show that the stop at stop_sequence = 2 (in this case the stop_id is 2168) lies between shape_pt_sequence 10012 and 20001. This suggests that the ten-thousands place of shape_pt_sequence can be used to identify stop locations, eliminating the need for map-matching the stops to the shapes. This pattern holds broadly for the MBTA but would not necessarily hold for another agency. Accordingly, this trick should not be used in a system architecture spanning multiple agencies. gtf_stops and gtf_shapes were joined here on shape_dist_traveled. The MBTA does not provide shape_dist_traveled for either table, but it can be inferred through map-matching and simple calculation, respectively.

In short, a route is driven along a shape. One run of the route from beginning to end is a trip. Along a trip, the bus makes stops. Trips can be defined with scheduled stops along the route, as defined in gtf_stop_times, or a range of times over which consecutive buses are to maintain a scheduled headway, as defined in gtf_frequencies. All of these attributes are interrelated.

The GTFS is a helpful way of conceptualizing transit data. However, there are some difficulties in working with GTFS data. First, though GTFS is standardized, there is substantial room for variation. Second, the GTFS can make maintaining a database across multiple agencies difficult. Third, the GTFS itself is not static.

Though GTFS is the standard for reporting static transit data, there is still a lot of

room for variation. For example, many of the tables and fields shown in Figure 2-1 are not required by Google (2011), such as the tables `calendar_dates`, `fare_attributes`, `fare_rules`, `shapes`, `frequencies`, and `transfers`. Further, there are variations in the underlying logic for the GTFS. For example, the MBTA does not ever schedule dwell time, *i.e.*, scheduled time to load and unload passengers, at stops. Also, the MBTA s `shape_pt_sequence` pattern with regard to stop locations and `shape_id` naming convention may not be replicated elsewhere.

Maintaining a single database of GTFS data which spans long time periods and multiple agencies is not trivial. Though generally static, agencies tend to update their GTFS as they make adjustments to their service. For example, the MBTA updates their GTFS quarterly. Incorporating these different versions is relatively simple. The `gtf_calendar` provides fields for a start and end date to a service. Any attempt to insert a duplicate `trip_id`, `shape_id`, etc. should be prevented by the SQL database as it preserves key integrity (assuming that an INSERT SQL command is used, not an UPDATE command). Combining the data from multiple agencies is more complicated. Many tables do not already have the `agency_id` and `feed_version` fields, whether explicitly or somehow encoded into the existing primary keys, complicating the task of differentiating which agency is operating which trips. Further, there is no guarantee that any of the `*_id` fields would be unique after combining various agencies. Managing a database across multiple agencies and multiple agency schedule definitions is a non-trivial task.

Complicating matters still, the GTFS standard itself is evolving, with fields and relationships being added and removed over time. As the GTFS changes, GTFS databases need to be restructured and queries on those databases need to be rewritten.

Despite any challenges with using or maintaining GTFS data, they are an invaluable resource for bus AVL research. They are the data onto which the AVL data need to be projected through map-matching.

2.3 Automatic Vehicle Location Data

There is not a generally agreed upon standard for AVL data, nor is there even a standard for what data precisely should be included in AVL data. This Section serves to both review different forms of AVL data, in Section 2.3.1, and introduce the NextBus MBTA AVL data specifically, including an introduction to their collection, in Section 2.3.2.

2.3.1 Types of AVL Data

As discussed at the beginning of the Chapter, there are two different types of AVL data: location-at-time and time-at-location. Then, within these categories, there are various formats for reporting the AVL data.

AVL data were historically collected through the use of special sensors positioned along the bus route (Parker 2008). These data by definition were time-at-location. Now, AVL systems tend to be Global Positioning System (GPS) based. GPS data can be either location-at-time or time-at-location.

In addition to NextBus, there are many other different formats for reporting AVL data. The European standard for AVL data is the Service Interface for Real time Information (SIRI) (Raschke 2011). SIRI is XML-based and provides very complete information, with the flexibility to provide time-at-location and location-at-time data.

OneBusAway (OBA) is an open-source AVL data service (Ferris 2011). OBA provides the data in a variety of formats, including a format that is compatible with SIRI (Ferris 2011).

The Chicago Transit Authority (CTA) developed its own data format to deploy AVL data. It is an XML format that is very similar to the MBTA NextBus format, though it does provide more information, including effectively map-matching ahead of time by providing the shape distance traveled (Chicago Transit Authority 2011). Further, the CTA publishes the XSD documents as well.

There is a wide range of AVL data types and formats.

2.3.2 NextBus MBTA AVL

This Section introduces the NextBus MBTA AVL data, covering everything from the generation of position data inside of a bus to NextBus publishing the data through the Internet. Understanding the data generation through publishing should help to inform the challenges of integrating data from numerous transit agencies and the irregularities that the system architecture must correct and that could affect results.

NextBus, Inc., a wholly owned subsidiary of Webtech Wireless (Webtech Wireless 2011), provides data for a large number of US and Candadian transit providers, as is shown in Table 2.1. The NextBus service can involve a wide array of functions, including bus arrival predictions and bus locations.

The NextBus MBTA AVL data go through many steps before I can access them. As is depicted in Figure 1-1, the data pass from reality to observation to being the primary data. Each of these steps affect the data and the inferences that can be drawn from them. Understanding the errors and biases introduced by those iterations is essential to effectively projecting the AVL data onto the GTFS data. Here, I trace through the data generation process, elucidating impacts of each step on the data.

First, AVL data are generated by a bus driver driving the bus. In the MBTA network, each bus is equipped with a GPS unit and a TransitMaster system (Barker 2011). The TransitMaster system, made by Trapeze ITS (2011), knows each driver's driving schedule for the day (Barker 2011). Each driver signs in to the TransitMaster unit at the beginning of the shift. The schedule information for the driver is integrated with the GPS so that the on-board computer knows what route the bus is driving to manage the bus's head sign and give accurate information about the route and trip being traveled. When everything is working perfectly, the TransitMaster handles the bus headsign text and on-board audio stop announcements, and reports the bus' location back to MBTA every 60 seconds. Errors anywhere in this chain can lead to errors in the AVL data. Trips could be missing because a driver did not sign in to the TransitMaster, because the TransitMaster is not turned on or is broken, because the GPS is not turned on or broken, or the link between them is not established or is

Tag	Title	Region
actransit	AC Transit	California-Northern
calu-pa	California University of Pennsylvania	Pennsylvania
camarillo	Camarillo Area (CAT)	California-Southern
chapel-hill	Chapel Hill Transit	North Carolina
charles-river	Charles River TMA - EZRide	Massachusetts
charm-city	Charm City Circulator	Maryland
oxford-ms	City of Oxford	Mississippi
collegetown	Collegetown Shuttle	Maryland
dc-circulator	DC Circulator	District of Columbia
da	Downtown Connection	New York
georgia-college	Georgia College	Georgia
glendale	Glendale Beeline	California-Southern
south-coast	Gold Coast Transit	California-Southern
lametro	Los Angeles Metro	California-Southern
mbta	MBTA	Massachusetts
mit	Massachusetts Institute of Technology	Massachusetts
moorpark	Moorpark Transit	California-Southern
brooklyn	NYC MTA - Brooklyn	New York
sria	Pensacola Beach (SRIA)	Florida
portland-sc	Portland Streetcar	Oregon
reno	Reno RTC Washoe	Nevada
rutgers-newark	Rutgers Univ. Newark College Town Shuttle	New Jersey
rutgers	Rutgers University	New Jersey
sf-muni	San Francisco Muni	California-Northern
seattle-sc	Seattle Streetcar	Washington
simi-valley	Simi Valley (SVT)	California-Southern
stl	Societe de transport de Laval	Quebec
temple	Temple University	Pennsylvania
thousand-oaks	Thousand Oaks Transit (TOT)	California-Southern
ttc	Toronto Transit Commission	Ontario
unitrans	Unitrans ASUCD/City of Davis	California-Northern
umd	University of Maryland	Maryland
vista	Ventura Intercity (VISTA)	California-Southern
wku	Western Kentucky University	Kentucky

Table 2.1: The list of the agencies using the NextBus AVL data as of writing. These data were found using the NextBus XML API with the command `agencyList`, though the `shortTitle` attribute of each `agency` was excluded.

broken. The radio used for reporting the positions could also be broken. Trips could also be missing because a trip was not run for any number of reasons, such as the vehicle being needed elsewhere in the network. There is no way of knowing directly from the AVL data why a trip is missing. These missing data mean that the NextBus AVL data always may be under-estimating bus performance. An observed headway of 20 minutes could mean that a third bus in between was missing, making for two ten minute headways. Moreover, the GPS data have accuracy within 1 m to 40 m at a 95% level (Quddus et al. 2007).

Second, the MBTA sends the data to NextBus and NextBus prepares to publish them. As stated above, each of the active MBTA buses report their positions every 60 seconds. Those positions are sent to the NextBus servers. Before publishing the data, NextBus does some preliminary map-matching (Barker 2011). NextBus closely guards all of their analytic and processing methods, regarding them as proprietary secrets (Barker 2011), so it is difficult to say precisely how they map-match. My intuition from examining the locations of GPS points is that NextBus matches them to the general street network rather than to the GTFS shapes. Generally, this map-matching is helpful for me and expedites the map-matching process because the AVL points have moved closer to the road network. However, there are two common problems that arise from this map-matching. First, matching to the street network is problematic when the buses are traveling on exclusive busways. The exclusive busways do not appear in the road network, and so the buses appear to be traveling on neighboring streets, such as shown below in Figure 3-16, often going the wrong way on a one way street or hoping from one side to the other. Second, the bus trajectories can appear to wander to neighboring streets in regions where the street network is particularly dense, such as shown below in Figure 3-15. The NextBus matching can cause the bus to 'hop' to a neighboring street. There is no way of knowing whether the bus actually took a detour or if NextBus erroneously matched the GPS point from the data.

Finally, NextBus publishes the data through the Internet. At this point, the data are freely open to the public, and any biases introduced into the data are caused by

those, like me, who access the data. So, now I shift from examining the process by which the data become accessible to discussing how precisely I access and archive them.

I access the data, as is shown in Figure 1-2, using the Java program ArchiveLiveFeed. NextBus provides its data as a REpresentational State Transfer (REST) eXtensible Markup Language (XML) document through their Application Programing Interface (API). This is commonly referred to as a REST-ful XML API. Unfortunately, the schema information, in the form of a XML Schema Definition (XSD), is not published. REST enables users to specify exactly which data they want (*e.g.* MBTA instead of San Francisco's Muni) by specially formatting a Uniform Resource Locator (URL) delivered through a HyperText Transfer Protocol (HTTP) GET request. NextBus (2011) provides full documentation of the data and respective commands that are available. For example, the base URL for the NextBus feed is `http://webservices.nextbus.com/service/publicXMLFeed`; to complete a query, various constraints are placed preceded by a question mark `?`.

The command for the raw AVL data (as opposed, for example, to stop predictions) is `vehicleLocations`. To request vehicle locations for MBTA vehicles, ArchiveLiveFeed sends HTTP requests such as:

```
http://webservices.nextbus.com/service/publicXMLFeed?
    command=vehicleLocations
    &a=mbta
    &t=0
```

Breaking down the URL, ArchiveLiveFeed is using the `publicXMLFeed service` to execute the `vehicleLocations command` for the `mbta a`(-gency, see Table 2.1), requesting the most recent locations of every active bus that has updated its position since epoch `t`(-ime) `0`. Epoch time is the number of milliseconds since epoch, January 1, 1970 UTC, so, assuming that every active bus has updated its location since January 1, 1970, using `0` requests all active buses' last positions.

After successfully receiving that HTTP request, NextBus sends an XML document

```
<body copyright="All data copyright MBTA 2011.">
    <vehicle
        id="2228"
        routeTag="9"
        dirTag="9_1_var0"
        lat="42.3392193"
        lon="-71.0513772"
        secsSinceReport="16"
        predictable="true"
        heading="310"
        speedKmHr="0.0"
    />
    .
    .
    .
    <lastTime time="1324007633490"/>
</body>
```

Figure 2-2: A portion of a sample NextBus MBTA XML document, including all of the possible tags and fields.

back to ArchiveLiveFeed. Figure 2-2 shows a portion of an XML document that NextBus sends back.

Figure 2-2 shows the full richness of the data. There are `body`, `vehicle`, and `lastTime` tags, with `vehicle` and `lastTime` both children of `body`. A `vehicle` has the attributes `id`, a unique vehicle ID; `routeTag`, the route the vehicle is traveling; `dirTag`, which here triply encodes the route ID, direction ID, and shape ID (*e.g.* `28_0_var3` is route 28 direction 0 (outbound) on the fourth (the index starts from zero) shape variation (280022)); `lon`, the longitude; `lat`, the latitude; `secsSinceReport`, the number of seconds between the query and when the bus was at that location; `predictable`, a Boolean for whether stop predictions are available for the bus; `heading`, a measurement of the direction in which the bus is pointed; and `speedKmHr`, which would be the bus speed if it were not always zero. AVL heading data are notoriously unreliable, especially at low speeds (Furth et al. 2006). The `lastTime` has one attribute, `time`, the NextBus server's epoch time when I queried it.

The `time` attribute of `lastTime` is very important for the Java program. As explained above, `t` is one of the fields of the REST-ful, and using `0` effectively asks for the most recent position of all of the active buses. The field `t` tells the server to give all of the positions of active buses that were updated after `t`; so giving `lastTime` in the XML document lets the user know when they last queried the server.

As mentioned above, NextBus does not publish an XSD for their XML feed. The standard Java Architecture for XML Binding (JAXB) is cumbersome to use without an XSD. So, instead of writing the XSD and using JAXB, I use the Processing XML library to parse the XML document (Fry and Reas 2011). Then, the Java program inserts the parsed data into the Postgres database where the GTFS tables live into the `rawdata` table. Then, to ensure that I capture all of the bus positions, ArchiveLiveFeed queries the NextBus server every 30 seconds, updating the `t` parameter at each iteration using the value from the previous `lastTime time`. That is the full extent of the functionalities of ArchiveLiveFeed.

Generally, ArchiveLiveFeed is robust enough to handle data from any of the agencies listed in Table 2.1. However, there are differences in the data NextBus provides for each of those agencies and across time for a single agency. The San Francisco Muni, for example, encodes its `dirTag` differently and gives non-zero `speedKmHr`. The MBTA `dirTag` has gone through at least two iterations, where the data `28_0_var3` was encoded as `28_280022v0_0` prior to 15 June 2011. Additionally, for at least a month and a half, the MBTA provided the GTFS `trip_id` as well, making the map-matching problem much easier.

Though the NextBus data provide a lot of information, they are not complete. Importantly, the NextBus location data do not provide information on when a bus arrives or departs from a stop. Most bus performance metrics require information on stop arrival times. As discussed above in Section 2.2, the GTFS data provides the latitude and longitude coordinates that define the bus stops and the poly-line shapes along which the buses travel. Thus, stop arrival and departure times have to be interpolated after map-matching the bus locations to the GTFS shapes.

2.4 Data Summary

In this Chapter, I introduced the datasets used for this analysis. First, I discussed location tracking data broadly in the context of human mobility and argued that bus location data, absent passenger data, can only be used to monitor the bus performance as a part of the infrastructure. I showed that, resultantly, I need a second dataset with static bus schedule data. Then, I discussed the GTFS as the static bus schedule data. Finally, I introduced the NextBus MBTA data, in addition to discussing AVL data generally.

Chapter 3

Data Processing and Cleaning

The previous Chapter covered the process of obtaining the location-at-time data that live in trajectory space and of obtaining the relevant data defining the human space. This Chapter serves to describe how I projected the trajectory space data onto the human space through a fast, robust system architecture. This projection has two steps. One, the trajectory data need to be map matched, a literal projection of the trajectories onto bus route shapes. Two, the data are transformed into time-at-location data by calculating the most likely stop arrival and departure times. The purpose of this Chapter is to describe how I completed each of these tasks and to advise on how the tasks might be done differently. Throughout, I note where the system is robust to processing another city's data and where it is not.

As described above, part of the objective of this research is to make analyzing AVL data more accessible to both practitioners and citizens. Accordingly, I will err on the side of verbosity, describing many of the less technical details thoroughly so as to gently initiate the uninitiated.

Taken together, these transformations prepare the NextBus MBTA AVL data for analysis. This preparation is broken into four basic steps:

1. Determine what shape the trip is traversing

    - With the trip_id and GTFS, this task is trivial, as shown in Figure 2-1. The NextBus MBTA AVL feed provided the `trip_id` at least during the

period from May 1, 2011 to June 15, 2011. The analyses here focus on this period. All of the NextBus MBTA AVL data have been archived since March 1, 2011 through the time of writing.

- Without either, this is more difficult. Even determining when a trip begins and ends can be non-trivial. However, as the MBTA NextBus feed provides the `vehicle_id`, `route_id`, and direction_id, breaking the records into trips should be possible. Some work has been done toward determining which trip a bus is traversing given its trajectory. In an earlier version, the NextBus MBTA AVL data did not provide the `route_id`. In response, Zeldovich (2010) developed a web-application that predicts the route of MBTA buses in real time. Learning the `trip_id` effectively adds the third dimension of time to the map-matching problem; time is the dimension that differentiates different trips on the same `shape_id`. Cathey and Dailey (2003) and Biagioni et al. (2011) use both time and location data to determine the `trip_id`. In the case where the `route_id` has many `shape_id`s, either the time data could be used to learn the `trip_id` or space data could be used to learn the correct `shape_id`. I do not present a solution to this problem; it is an important area for further research.

2. Determine the best projection of each coordinate to its `shape_id`. This is the map-matching problem. As described above, in the case where the `shape_id` is not directly accessible from the AVL data, this step is concurrent with determining the `shape_id`. To simplify the subsequent analyses, the map-matching does not simply transform the AVL data from one set of latitude-longitude coordinates that do not necessarily lie on the `shape_id` to another set that do. Rather, the latitude-longitude coordinates are translated to a distance-along-shape (or, in GTFS language, `shape_dist_traveled`). Projecting to the distance-along-shape is preferable because it is one- rather than two-dimensional, and it simplifies calculating most statistics about a trip, including the bus speed and the proximity of a bus to a stop.

3. Determine what stops is the bus is supposed to service and the locations of those stops. With the GTFS data and an accurate `trip_id`, this problem is also trivial. As the `trip_id` and GTFS data are necessary for the above stops, I skip describing this step in detail.

4. Determine when the bus serviced each of the stops. Generally, this is done by comparing the `shape_dist_traveled`s of the AVL trajectory with those of the shape's `stop_id`s.

This Chapter describes these methodologies, tracing the data from a raw data file to an analyzable format. First, Section 3.1 prescribes a map-matching procedure for projecting NextBus MBTA AVL data onto GTFS data, or, using the terms from Figure 1-1, it describes merging the raw AVL data and GTFS data into bus-on-GTFS-shape and stop-on-GTFS-shape data. Second, Section 3.2 offers two different methods for calculating bus stop arrival and departure times. These methods finish Figure 1-1 by combining bus position and scheduled stop data into stop arrival and departure data.

3.1 Map-matching

This Section covers the map-matching. First, Section 3.1.1 gives an overview of the map-matching problem, map-matching terminology, and other approaches that have been used to map-match location traces. Second, Section 3.1.4 shows the ways in which the NextBus MBTA AVL map-matching fail so as to inform the decisions I made in formulating my map-matching algorithm. Third, Section 3.1.5 describes the map-matching procedure itself. Finally, Section 3.1.6 presents the results of my map-matching algorithm.

3.1.1 Map-Matching Overview

Map-matching is the process of assigning a measured vehicle position to a road segment in a digital map (Hummel 2006). GPS devices have errors of 1 meter to 40

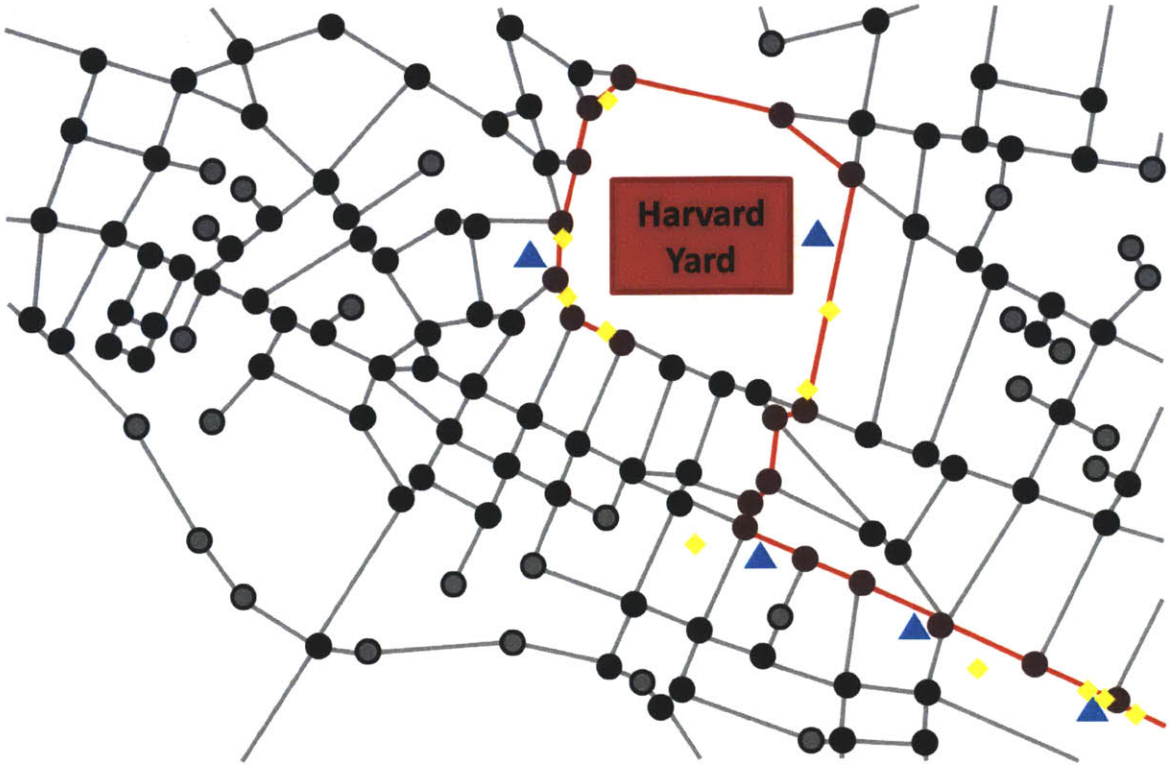

Figure 3-1: A full map of the area surrounding Harvard Yard in Cambridge, MA, including the MBTA Route 1 inbound shape, stops, and sample bus GPS points. Nodes are shown as circles, where black circles are intersections, gray circles are not, and crimson circles are bus shape nodes. Edges are shown as lines, where red lines are bus route edges. Bus locations are yellow diamonds, and bus stops are blue triangles.

meters (Quddus et al. 2007), so the task is not trivial. Others refer to the task as snapping the GPS coordinates to the road. This overview is divided into two parts. First, Section 3.1.2 summarizes the general terms and concepts of map-matching. Second, Section 3.1.3 reviews previous map-matching literature.

3.1.2 Map-Matching Terms

The map-matching problem has been formulated and thought of in many different ways. As is shown in Figure 3-1, map-matching is generally thought of as projecting onto a graph of the street network. There are edges, shown in Figure 3-3, as defined by the roads, and nodes, shown in Figure 3-2, as defined by street intersections and termini. In the context of GTFS map-matching, the edges are referred to as shape

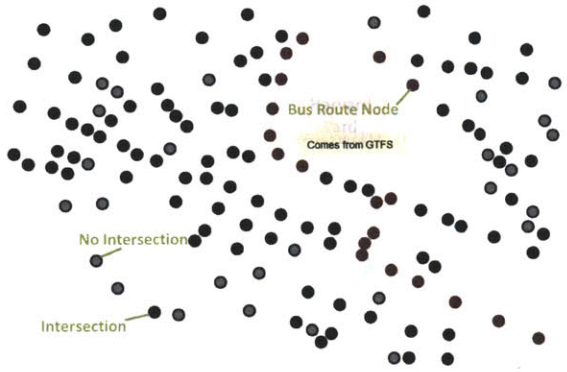

Figure 3-2: Harvard Square area nodes in the road network. Intersections are shown in black, mid-street nodes are shown in gray, and nodes for the MBTA Route 1 inbound are shown in crimson.

Figure 3-3: Harvard Square area edges in the road network.

Figure 3-4: Harvard Square area bus GPS locations.

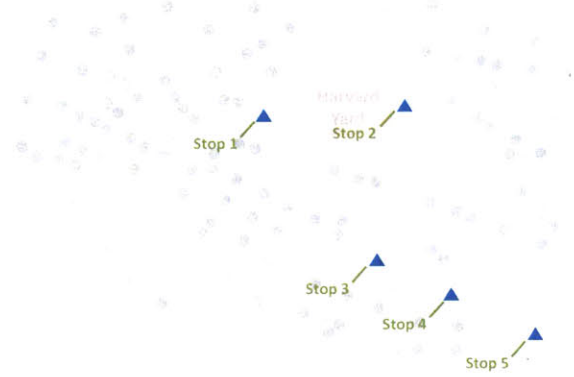

Figure 3-5: Approximate locations of the Harvard Square area bus stops for the MBTA Route 1.

segments, and the nodes are referred to as shape points. The task is to map the GPS traces, in this case from buses shown in Figure 3-4, onto the road network. Finally, in bus performance research, the bus stop locations, shown in Figure 3-5, also need to be map-matched.

There are many different procedures for map-matching. Broadly, all of the procedures are a combination of point-to-point, point-to-curve, and curve-to-curve matching, which are depicted in Figures 3-6, 3-7, and 3-8, respectively. Each method is what it sounds like: in point-to-point matching, each GPS coordinate is simply matched to the nearest node; in point-to-curve matching, each GPS coordinate is matched to the nearest edge; and in curve-to-curve matching, the trace history is used to find

the best fit to the shape of the road. Most map-matching algorithms involve a combination of these methods; curve-to-curve matching itself explicitly involves at least point-to-curve matching if not also point-to-point matching.

Generally, regardless of the map-matching type, GPS coordinates are not just matched to the best fit. Rather, some coordinates are discarded because they do not fit well. For example, a GPS coordinate located in Connecticut would be thrown out when matching the Boston bus data. Usually, a maximum threshold distance is specified, and points that lie outside of the threshold are discarded.

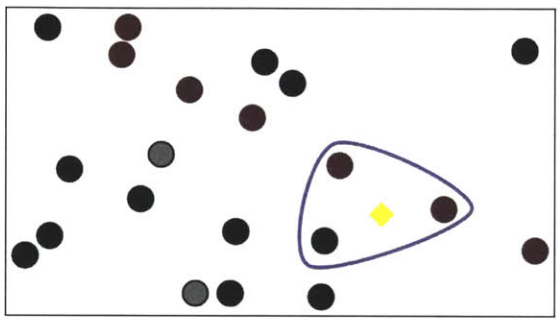

Figure 3-6: Point-to-point map-matching

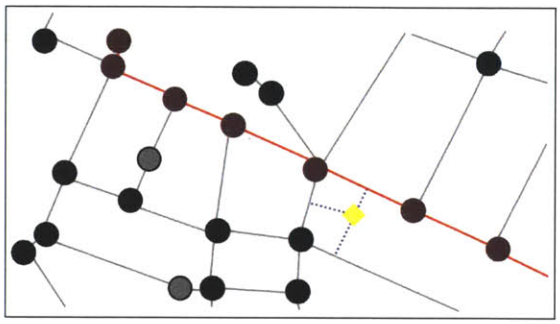

Figure 3-7: Point-to-curve map-matching

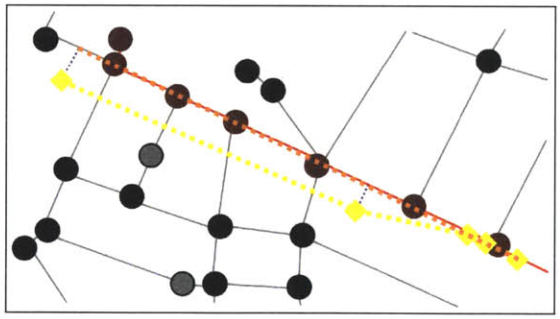

Figure 3-8: Curve-to-curve map-matching

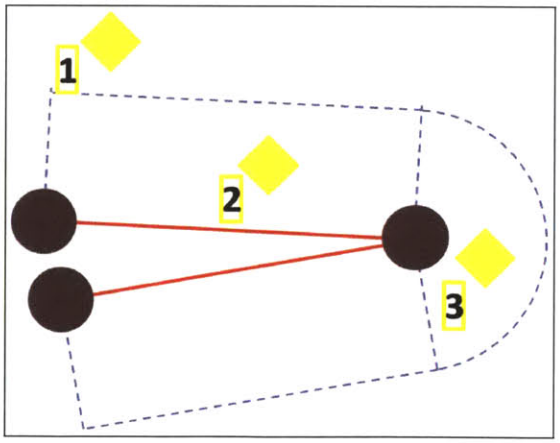

Figure 3-9: A comparison of point-to-point and point-to-curve map-matching. Point 1 does not successfully match. Point 2 matches as point-to-curve. Point 3 matches as point-to-point.

Figure 3-9 displays the codependent relationship between point-to-point and point-to-curve matching. As in Figures 3-1 through 3-8, the red lines indicate bus

shape edges, the crimson circles indicate bus shape nodes, and the yellow diamonds indicate bus AVL points. The dashed purple lines define the region, determined by some arbitrary threshold distance from the poly-line, outside of the acute angle within which a point must lie to be successfully matched to the shape. The two rectangular regions are regions in which a point may successfully project onto an edge by point-to-curve matching. The pie-shaped region is a region in which a point may successfully project onto a node by point-to-point matching. Thus, point 1 does not successfully match; point 2 matches as point-to-curve; and point 3 matches as point-to-point. This Figure shows that while point-to-curve matching generally projects points onto the shape more precisely, point-to-curve matching also needs point-to-point matching to pick up regions inside of reflex angles. If point-to-point matching were not used here to complement point-to-curve matching, point 3 would not be successfully matched.

In sum, map-matching uses point-to-point, point-to-curve, and curve-to-curve matching. These methods are generally dependent on one another. Some points that are too far from any roads in the human space data are discarded.

3.1.3 Map-Matching Literature

There are a wide range of methodologies for combining point-to-point, point-to-curve, and curve-to-curve matching. Map-matching algorithms tend to be differentiated depending on the desired numerical calculation speed of map-matching and the polling frequency of the location data. First, the desired speed of map-matching affects the algorithm used. If pure speed is the most important characteristic, point-to-point matching alone is used; however, this is rare. More commonly, algorithms are separated by the polling time of the GPS coordinates. The polling time is the frequency with which GPS points are recorded. Polling times tend to be either on the order of seconds (one coordinate every 1 to 5 seconds) or minutes (one coordinate every 30, 60, or 120 seconds). Conservatively estimating city driving speeds at 30 kph, this is a difference between getting one point every 8 to 40 meters and one point every 250 to 1,000 meters–quite a significant difference.

Most map-matching literature concerns high-frequency GPS polling, which is what

automobile GPS routing uses. As discussed in Section 2.3.2, the MBTA data are polled every 60 seconds, so my treatment of high-frequency polling will be brief. Quddus et al. (2003) give a helpful tabular summary of different map-matching algorithms, and Quddus et al. (2007) provide a formal and more up-to-date review of various map-matching techniques. However, most of those techniques seem to have been supplanted by Hidden Markov models (HMM) using the Viterbi algorithm introduced by Hummel (2006) (*see e.g.*, Dong and Pentland (2009) and Thiagarajan et al. (2009) for examples of HMM used for map-matching). Thiagarajan et al. (2009) use an interesting hybrid of cell phone GPS and the WiFi access point fingerprint to reduce battery drain.

Less map-matching literature is devoted to low-frequency GPS polling. Many of the sophisticated high-frequency methods, such as Quddus et al. (2003), rely heavily on the immediate past of the trace, heavily weighting the probability that a vehicle has stayed on the same road. With low-frequency polling, the vehicle may have made two or more turns between polls, making such probabilities all but useless. Yang et al. (2005) and Wu et al. (2007) both approach the low-frequency polling map-matching problem by first collecting the subset of likely matches, found by point-to-point and point-to-curve matching, for each GPS coordinate and then searching for the most likely route through them. Yang et al. (2005) note that there could be many routes between two coordinates and suggest using the shortest path, calculated using Djikstra's algorithm, between them as the most likely route. My algorithm borrows heavily from Yang et al. (2005).

Map-matching buses is then a different subject still, as alluded to in Section 2.1, discussing what bus GPS data can and cannot be used for. Specifically, it is generally not a mystery where a bus is going to go: it has a route that it is assigned to follow and times when it is supposed to travel along that route. In GTFS terminology, the bus has a `trip_id`. Buses may have to make detours for various reasons, but the general assumption is that they will follow their route. This difference greatly simplifies the map-matching process from projecting onto a graph as defined by the road network to projecting onto a poly-line as defined by the GTFS data. Bonzon (2007) exploits

this for his map-matching of the Lausanne, Switzerland bus lines near EPFL by only searching over the bus route. Bonzon (2007) has a very elegant algorithm which is accessibly described. However, Bonzon (2007) polls every 3 seconds, and so some different tactics are needed for the NextBus data. Some work has pushed to eliminate the need for GTFS shape definition information by learning the shapes from historical GPS data. Biagioni et al. (2011) develop a system architecture that estimates the route shapes from the bus GPS traces themselves, in addition to map-matching and completing trip prediction on the GPS points. Biagioni et al. (2011) also use high-frequency GPS data and use a HMM for their map-matching and trip prediction algorithms.

Given that the data only need to be mapped to one poly-line, some may suggest simply using PostGIS (Refractions Research 2011), an open-source GIS-specific database software built on top of Postgres, to snap the GPS coordinates to the GTFS poly-line. However, I do not think this gives enough control over the map-matching process and could lead to confusing results when a route overlaps with itself.

Map-matching involves projecting location coordinates onto a road network. Map-matching the NextBus MBTA AVL data is a unique problem because the data have a long polling time and because I know *a priori* where the bus is traveling.

3.1.4 Data Errors and Biases

As a result of both the way that the bus GPS positions are handled and the uncertainties endemic to GPS data, some points either cannot be successfully projected onto the shapes or project onto the shapes in an illogical way. There are a number of different ways in which the bus AVL data match to their GTFS shape poorly. I describe these symptoms here so that I can look for diagnoses in the data generation, publishing, and archiving processes described above. Further, these errors are used to inform the map-matching algorithm, described below in Section 3.1.5, capturing exceptions that the algorithm needs to handle. The errors are GOES_BACKWARDS, NO_MATCH, TOO_FEW_POINTS, and DOUBLE_TRIP, and they can be interpreted literally. They are discussed in detail below.

I found these errors by inspecting the quality of map-matching visually. To accomplish this, I developed a simple tool that reads a file dump of trips with errors in them that is generated by the map-matching algorithm. The tool is built in Processing (Fry and Reas 2011), an open-source Java-based programming language designed for visualizations, and uses the unfolding library (Nagel 2011) to access OpenStreetMap contributors (2011, 2012) maps which are licensed under CreativeCommons (2012). An earlier version of the tool used ModestMaps for the maps instead of unfolding (Carden 2011,2012); I moved to unfolding because ModestMaps did not support using OpenStreetMap for map tiles at the time, though ModestMaps does as of writing.

The tool operates by reading two text files: one that defines the shapes and a second that gives the AVL points. These text files are generated by SQL SELECT statements on the database. The tool allows the user to play through the trips uninterrupted, pause play to investigate interesting cases, move time in one-step increments while paused, zoom in and out, manipulate the map center position as a 'slippy-map,' increase or decrease the number of AVL points displayed at one time, and take a snapshot of the current scene. The tool automatically changes the zoom level and map centering to follow the AVL points' trajectory, though, as mentioned above, these attributes can be customized.

The maps generated by the tool, used in Figures 3-10 to 3-17, denote the bus shape by a gray line and the shape segment to which a GPS point was matched with a magenta line. AVL points that successfully match to the shape are shown with a green circle; AVL points that were not successfully matched are colored red, orange, or pink depending on if they are a GOES_BACKWARDS, NO_MATCH, or TOO_FEW_POINTS, respectively. Finally, the tool displays the `shape_id` in the upper-right corner of the map. The arrows showing the direction of travel and numbers indicating the order of the AVL points are added manually to snapshots of the map program.

There are four general error types in the NextBus MBTA AVL data:

1. **GOES_BACKWARDS**: As depicted in Figure 3-10, GOES_BACKWARDS is an error when the bus travels backwards along the shape. The shapes are signed poly-lines and in the map-matching process, the AVL points are projected to a

distance along that poly-line. Being a bus service, the bus is expected to have a non-decreasing distance along shape across its trip. There are a few reasons that a bus may go backwards:

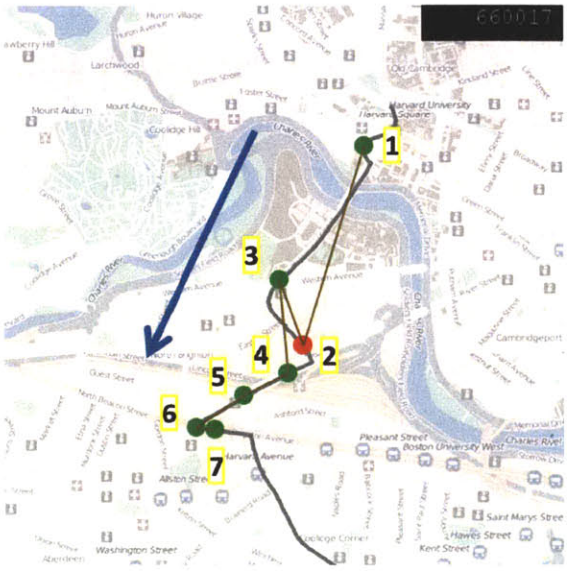

Figure 3-10: Example of the map match error Goes Backwards shown in red. Specifically, this is a 'jump back' error. Map tiles ©OpenStreetMap contributors (2011, 2012), CC-BY-SA (CreativeCommons 2012).

(a) **'Jump back'**: Some buses appear to 'jump' backwards. Figure 3-10 is an example of a bus 'jumping' backwards. In these cases, the bus shows a point backward along the shape within about 20 seconds of another point that is farther along the shape; thus it has 'jumped' backwards in less than 20 seconds. This situation is especially odd because the bus locations are only polled every 60 seconds. The obvious solution is to throw out the first point with the intuition that the quickly following point is a correction of the first one. This is the solution adopted in Figure 3-10. However, time is confusing here. Up to now, the time I have referred to is the 'real' time calculated by subtracting the `secsSinceReport` attribute of the vehicle record in the XML document from the `t` attribute of the query, or 'query' time. In 'real' time the bus jumped back, but in 'query' time, they are in the correct order (both distance on shape and 'query' time increase

monotonically). This confusion of times throws into question all of the reported times. It is not clear which time is correct.

(b) **'Small movement'**: Some buses appear to move a very small distance backwards, less than 10 meters. These cases are likely due to GPS errors and so are recoded as WAS_GOES_BACKWARDS with a distance along shape identical to the point they moved slightly back from. By labeling them as WAS_GOES_BACKWARDS, I give more information for future analyses so that these records can be excluded if so desired.

(c) **'Going backwards'**: Some buses appear to actually be moving backwards along the shape, with a sequence of two or more consecutive points traveling backwards along the shape. Many of these are very early in the morning and appear to be the bus driving to the beginning of its route. Others just turn around in the middle of a route and then 'disappear' from the trip. Clearly, in these cases the bus is not providing bus service during these periods of going backwards, so the points are thrown out of the dataset.

2. **NO_MATCH**: As depicted in Figures 3-11 thru 3-16, NO_MATCH is an error where the distance between the AVL point and its projection onto the shape is too large. The threshold distance is a normative choice. There are many different cases where buses get NO_MATCH errors:

   (a) **Before Beginning**: One very common NO_MATCH occurs when there are points before the beginning of the shape. Figure 3-11 shows an example of this case. This error is likely the result of a disconnect between the TransitMaster definition of the shape and the GTFS definition of the shape. Even though these points are deemed too far from the shape, they can be used to estimate when the bus started its trip when there are not any well-matched points close to the beginning of the shape. Many other researchers have had problems with matching the beginning of trips (Furth et al. 2006).

   (b) **Entering Shape**: A less common NO_MATCH comes from a bus entering

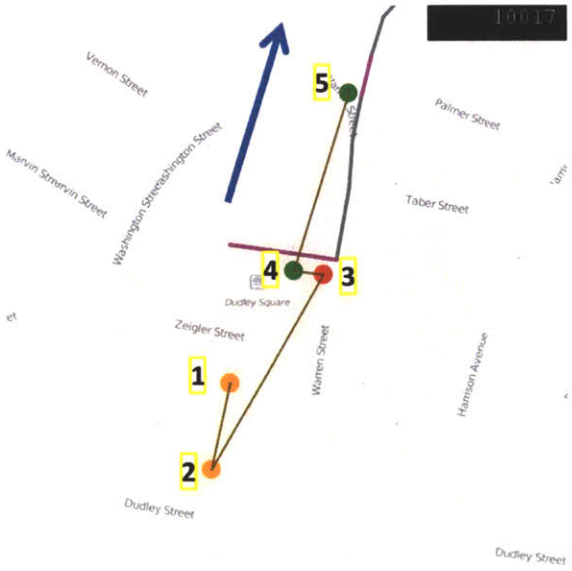

Figure 3-11: NO_MATCH because point is before the beginning of the shape, shown in orange. Map tiles ©OpenStreetMap contributors (2011, 2012), CC-BY-SA (CreativeCommons 2012).

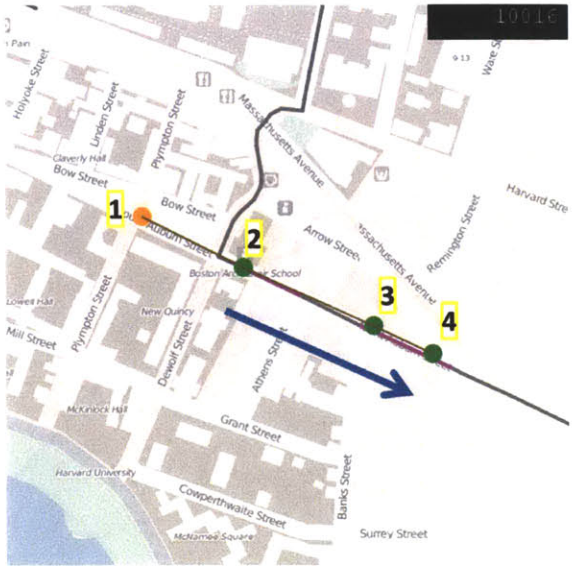

Figure 3-12: NO_MATCH because point is before entering the shape, shown in orange. Map tiles ©OpenStreetMap contributors (2011, 2012), CC-BY-SA (CreativeCommons 2012).

a shape somewhere in the middle of the trip, as shown in Figure 3-12. This bus trip has a single point too far from the shape just before entering the shape.

(c) **After End**: Another very common NO_MATCH happens when there are points after the end of the shape. Figure 3-13 shows an example of this case. This error is very similar to the before the beginning error, and so similarly, these points can be useful for determining when a bus finishes its trip when those data are unavailable. Many other researchers have cited problems with collecting data on the end of the trip (Furth et al. 2006).

(d) **Leaving Shape:** Another less common NO_MATCH comes from a bus leaving the shape without completing the trip, as shown in Figure 3-14. This bus trip has a single point too from the shape just after leaving the shape.

(e) **On a Parallel Street**: Some buses appear to hop onto a neighboring

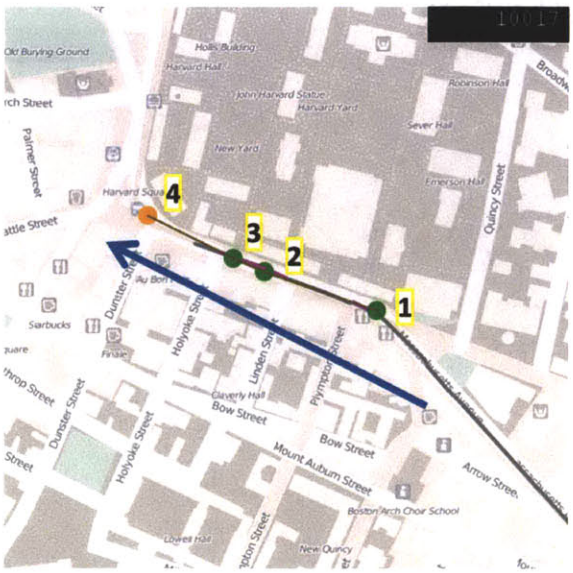

Figure 3-13: NO_MATCH because point is after the end of the shape, shown in orange. Map tiles ©OpenStreetMap contributors (2011, 2012), CC-BY-SA (CreativeCommons 2012).

Figure 3-14: NO_MATCH because point is after exiting the shape, shown in orange. Map tiles ©OpenStreetMap contributors (2011, 2012), CC-BY-SA (CreativeCommons 2012).

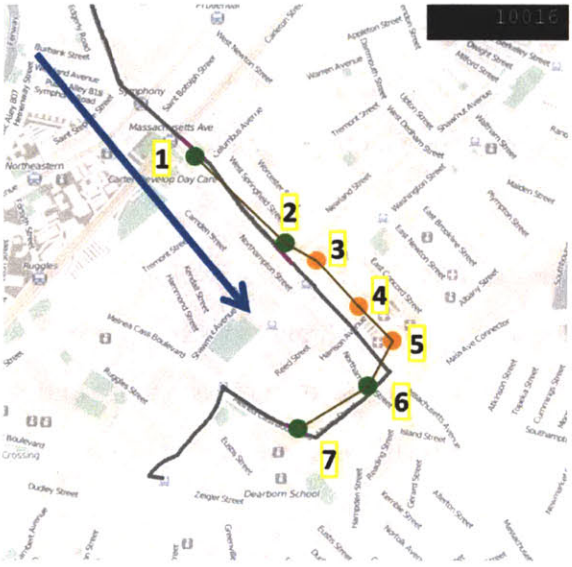

Figure 3-15: NO_MATCH because points are on a parallel street, shown in orange. Map tiles ©OpenStreetMap contributors (2011, 2012), CC-BY-SA (CreativeCommons 2012).

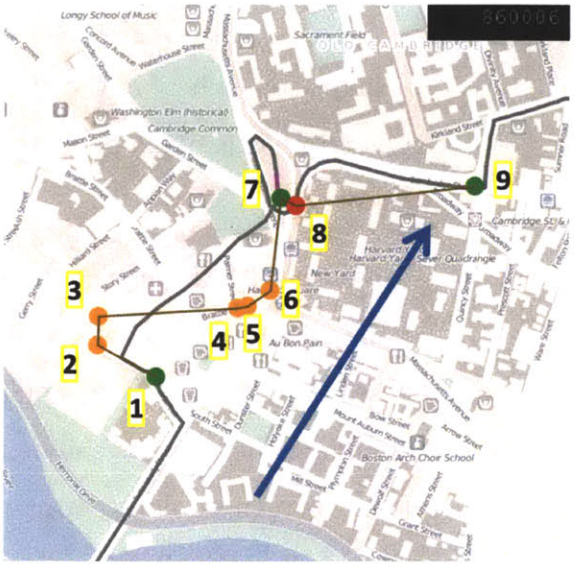

Figure 3-16: NO_MATCH because bus is on an exclusive bus-way, shown in orange. Map tiles ©OpenStreetMap contributors (2011, 2012), CC-BY-SA (CreativeCommons 2012).

parallel street for a little while, as is shown in Figure 3-15. Sometimes the

bus will even appear to be traveling the wrong way on a parallel one-way street. Ultimately, there is no way of knowing whether this error is a result of the bus actually taking a detour, a GPS problem, or NextBus' preliminary map-matching without directly contacting the service provider. If the data span many providers, contacting the service providers is not a reasonable mechanism for resolving errors. Rather, these data should be discarded. If the bus did take a detour, then the dataset should reflect that the stops passed while on the detour were not served; however, I have not implemented this.

(f) **On an Exclusive Bus-way**: In some cases, the bus will appear to jump around to many different places while it is traveling on an exclusive bus-way, as is shown in Figure 3-16. There, the bus is entering the Harvard Square bus station, which has both an above- and underground exclusive bus-way. This error is likely to be caused by either GPS errors as the bus travels underground or the NextBus preliminary map-matching. It is impossible to tell in what directions the data have been distorted by either mechanism, so rather than accept their best projection to the poly-line, I discard them. Additionally, it would not be trivial to algorithmically determine regions of the shape where the bus is on an exclusive bus-way, though one-dimensional clustering could help.

3. **TOO_FEW_POINTS**: Some bus trips have very few AVL points in them and are often traveling the wrong way on the shape. TOO_FEW_POINTS where the bus is headed in the correct direction is shown in Figure 3-17. These trips likely reflect that a vehicle was reassigned or that the TransitMaster is exhibiting other erroneous behavior; these trips should be discarded entirely.

4. **DOUBLE_TRIP**: On a few rare occasions, a trip appears to be run twice in a day, sometimes by the same vehicle and sometimes not. The second time is usually hours after the first, and the first is at the scheduled time. One example of this error occurred on Sunday before Memorial Day, suggesting that

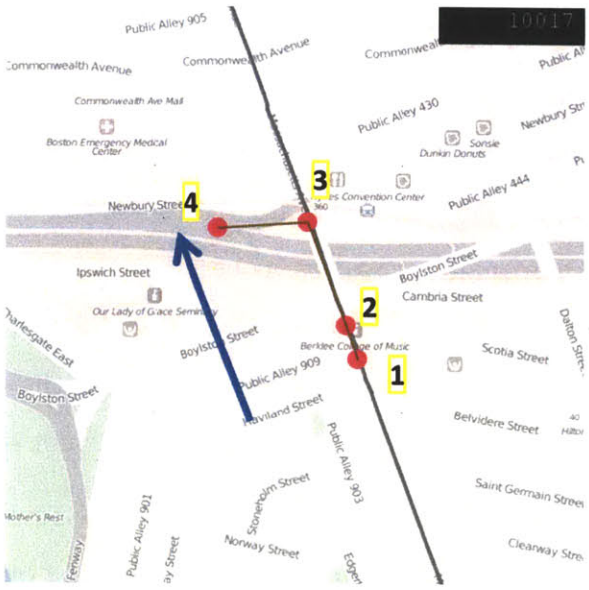

Figure 3-17: Example of the map match error Too Few Points, shown in pink. Map tiles ©OpenStreetMap contributors (2011, 2012), CC-BY-SA (CreativeCommons 2012).

sometimes trips are doubled as emergency service to accommodate crowds. It is not obvious how to handle these cases. Throwing out the second trip would underestimate the service level on that route and so is undesirable. The simplest solution would be to define a new `trip_id`, such as `XXXXXXX.1` for the second run. However, that solution would violate the integrity of the GTFS database. Ultimately, there are so few instances of this occurring that it is a relatively unimportant error for the map-matching procedure to handle. But the effects of double trips are significant. Stop arrival times are often approximated as the first time the bus passes into the stop region, and stop departure times as the last time it moves out. So if the same trip is run twice in a day, it could appear that the bus dwells at each stop for hours and then travels back in time to get to the next stop.

Though map-matching buses seems much simpler than map-matching an automobile that can go anywhere, there are a variety of data errors that the map-matching procedure needs to tolerate and handle. The existence of these errors shape the map-matching algorithm.

3.1.5 My Map-Matching Procedure

My map-matching algorithm is built specifically to robustly tolerate the errors discussed in Section 3.1.4. The purpose of this Section is to describe my map-matching methodology.

The map-matching procedure begins with my Java program, depicted in Figure 1-2 as MapMatch, querying the SQL database for a Java RecordSet. The RecordSet returns records with the fields vehicleID, tripID, latitude, longitude, realTimestamp, and tripDay sorted first by vehicleID then by realTimestamp. The procedure then

1. Breaks the RecordSet into chunks of records, each representing a single trip as defined by tripID, tripDay, and vehicleID. Two records are labeled as originating from the same trip if they have the same tripID, tripDay and vehicleID.

2. For each trip

   (a) Match the coordinates (composed of the latitude, longitude and realTimestamp of each record) to the corresponding shape as closely as possible, noting whether each record was matched to the shape well or not. This algorithm, described in Algorithm 1, hierarchically uses point-to-curve then point-to-point matching as shown in Figure 3-9. To speed matching, PointMatch assumes the coordinate will match to the shape well, and so only looks forward along the shape. This procedure is described in Algorithm 2, GoodMatch. If a good match is not found, the algorithm looks from the beginning of the shape and considers any possible match, not just those near the shape. This procedure is described in Algorithm 3, AllMatch.

   (b) Investigate whether the poor matches are actually better than first estimated and vice versa. This procedure is broken into Algorithm 4 and Algorithm 5. This step directly addresses the errors, whereas the previous step is only structured to enable this one.

   (c) Write each record back in the database with its best match to the shape and a code that notes whether it should be regarded as a good match for

analytic purposes or not.

```
input  : trip = {(latitude, longitude, timestamp)}, shape =
         {(latitude, longitude, sequence)}
output: PointMatchedTrip
1 foreach coordinate in trip do
2 |   populate candidates with GoodMatch(coordinate,shape);
3 |   if found at least one candidate then
4 |   |   choose winner and code as GOOD_MATCH;
  |   |   // Code for GoodMatch() is in Algorithm 2
5 |   else
6 |   |   find the coordinate's best match though bad match using
  |   |   AllMatch(coordinate,shape);
  |   |   // Code for AllMatch() is in Algorithm 3
7 |   end
8 end
```

Algorithm 1: The PointMatch algorithm

Map-matching at the trip level takes place in two stages:

1. As winning candidates are read in from PointMatch (Algorithm 1), they are classified by ReadTrip using their score (*i.e.*, distance from the shape) and their relationship to the directly previous matched coordinate, as described in Algorithm 4.

2. Then the errors are checked by TripFixer to see if their information can be used to reclassify them as good matches, as described in Algorithm 5.

There are four different possible causes for a coordinate not matching to the shape well. Two of them suggest that the point should be matched to the shape regardless and two that they should not:

- Do NOT match to the shape:
  - The bus took a detour
  - The GPS is malfunctioning
- Do match to the shape:

```
input  : coordinate, shape
output: candidates
1  while more shape segments and (just added a candidate or candidates is
   empty) do
2  |  increment the shape segment;
3  |  if coordinate projects onto this segment within threshold then
4  |  |  add this projection to the list candidates;
5  |  else if distance from coordinate to first node of segment then
6  |  |  add this node to the list candidates;
7  |  else if within a reasonable distance of the first node that the bus could
   |  have driven then
8  |  |  note that, though not a new candidate, pretend as if there was a new
   |  |  candidate;
9  |  else
10 |  |  no reason to keep searching if there are candidates;
11 |  end
12 end
```

Algorithm 2: The GoodMatch algorithm

```
input  : coordinate, shape
output: candidates
1  while more shape segments do
2  |  increment the shape segment;
3  |  if coordinate projects onto this segment at all then
4  |  |  add this projection to the list of candidates;
5  |  end
6  |  add the first node of the segment as a candidate, regardless of the
   |  distance;
7  end
```

Algorithm 3: The AllMatch algorithm

- NextBus map-matching distorted the bus position
- The bus is before the beginning or after the end of the shape, just outside the threshold to be a GOOD_MATCH

The trip-level map-matching tries to take this into account. However, as the present time my procedure does not match points that are likely a NO_MATCH because of NextBus map-matching. This deficiency is partially because it is difficult to prove that the NextBus map-matching is the culprit. However, some errors, such as those seen near the Harvard Square bus station where buses are traveling on exclusive busways shown in Figure 3-16, are regular enough to make this diagnosis. For example, one could employ one-dimensional clustering to see if there is a region of the shape to which points are never successfully projected and reasonably hypothesize that NextBus map-matching caused the problem.

```
   input  : coordinate
   output: RECORD_CLASSIFICATION
 1 if distance from shape is outside of a threshold then
 2 |  classify as NO_MATCH;
 3 else if the shape_dist_traveled has decreased relative to previous coordinate
   then
 4 |  if only went backwards a little bit then
 5 |  |  classify as WAS_GOES_BACKWARDS;
 6 |  else
 7 |  |  if this is the beginning of the shape and the previous coordinate also
   |  |  went backwards then
 8 |  |  |  classify as GOOD_MATCH;
   |  |  |  /* By this logic, the previous coordinate was marked as
   |  |  |     GOOD_MATCH, so:                                          */
 9 |  |  |  classify previous as GOES_BACKWARDS;
10 |  |  else
11 |  |  |  classify as GOES_BACKWARDS;
12 |  |  end
13 |  end
14 else
15 |  classify as GOOD_MATCH;
16 end
```

Algorithm 4: The ReadTrip algorithm

```
   input  : PointMatchedTrip
   output: TripMatchedTrip
 1 if PointMatchedTrip has very few coordinates then
 2 |  all coordinates are marked as TOO_FEW_POINTS;
 3 else if there are any NO_MATCH or GOES_BACKWARDS then
   |  // Try to fix these errors
   |  // Fix the beginning of the trip
 4 |  if there are not any GOOD_MATCH at the beginning of the trip then
 5 |  |  if a NO_MATCH is near the beginning then
 6 |  |  |  classify that NO_MATCH as GOOD_MATCH to capture the
   |  |  |  beginning;
 7 |  |  else if a GOES_BACKWARDS is near the beginning then
 8 |  |  |  classify that GOES_BACKWARDS as GOOD_MATCH to capture
   |  |  |  the beginning;
 9 |  |  end
10 |  end
   |  // Fix the end of the trip
11 |  if there are not any GOOD_MATCH at the end of the trip then
12 |  |  if a NO_MATCH is near the end then
13 |  |  |  classify that NO_MATCH as GOOD_MATCH to capture the end;
14 |  |  end
   |  |  // matching a GOES_BACKWARDS does not make sense
15 |  end
   |  // Fix any remaining GOES_BACKWARDS
16 |  if there are still some GOES_BACKWARDS in PointMatchedTrips then
17 |  |  if there is a sequence of multiple GOES_BACKWARDS then
18 |  |  |  classify all in sequence as GOES_BACKWARDS;
19 |  |  end
20 |  |  if the GOES_BACKWARDS is a 'jump back' then
21 |  |  |  use the one that is later in time as the GOOD_MATCH;
22 |  |  end
23 |  end
   |  // Fix any remaining errors
24 |  if still any GOES_BACKWARDS or NO_MATCH then
25 |  |  finalize their current category;
26 |  end
27 else
   |  // The trip is good
28 |  the PointMatchedTrip is good to begin with, TripMatchedTrip ←
   |  PointMatchedTrip ;
29 end
```

Algorithm 5: The TripFixer algorithm.

	Good Match	Was Goes Back-wards	Was No Match	Goes Back-wards	No Match	Too Few Points	Total
01	98.7%	0.4%	0.0%	0.1%	0.7%	0.1%	100.0%
66	97.5%	1.2%	0.0%	0.9%	0.3%	0.1%	100.0%
80	96.5%	0.5%	0.0%	0.5%	2.3%	0.2%	100.0%
85	99.4%	0.2%	0.0%	0.1%	0.2%	0.1%	100.0%
86	96.5%	0.3%	0.0%	0.6%	2.5%	0.1%	100.0%
87	97.1%	0.6%	0.0%	0.8%	1.2%	0.2%	100.0%
91	97.0%	0.3%	0.0%	1.3%	1.3%	0.2%	100.0%
94	97.5%	0.3%	0.0%	0.2%	2.0%	0.0%	100.0%
96	96.6%	0.2%	0.0%	0.1%	2.9%	0.1%	100.0%
Total	97.6%	0.6%	0.0%	0.5%	1.2%	0.1%	100.0%

Table 3.1: Map-matching success rates on the reported months for the month of May 2011. Total sample size across all of the routes is N=930,109.

3.1.6 Map-Matching Results

Ultimately, the map-matching algorithm is very successful, as is shown in Table 3.1. Across 10 routes in the month of May 2011, over 97% of the points were successfully matched to GTFS shapes. Additionally, the map-matching algorithm is fast; all of the results reported in Table 3.1 required less than twenty minutes.

3.2 Data Interpolation

The data interpolation is thought to be the most important step of the entire process and the crux of Furth et al. (2006)'s argument that location-at-time data cannot be used to monitor bus network performance. The stop arrival and departure times are interpolated from the observed data. I use two different interpolation methods. The first, described in Section 3.2.1, is naïve interpolation. Dietmar Bauer, my collaborator, created the second interpolation, described in Section 3.2.2, which uses quadratic optimization techniques.

3.2.1 Naïve Interpolation

The naïve interpolation is calculated in a simple SQL SELECT statement by linearly interpolating the time at which the bus arrived and departed from the stop as the time at which it was 50 meters before and after the stop, respectively. 50 meters is an arbitrary measure. Accordingly, the first and last stops only have a departure and arrival time, respectively. For trips when the bus does not appear both before and after a stop, the naïve method does not extrapolate an arrival or departure time for that stop. Resultantly, many trips do not visit stops at the beginnings and ends of the shapes, a common problem in AVL studies (Furth et al. 2006).

3.2.2 Resampling Interpolation

Many works have used statistical techniques, such as Kalman filters, to smooth AVL data (*e.g.* Cathey and Dailey (2003)). Assuming that there is non-zero distance between all consecutive AVL points, naïve interpolation assumes that the bus is always moving. Even the Kalman filter works to smooth the velocity profile over the bus trip. While this method would intuitively more accurately capture the bus speed along the trip, part of that smoothing would serve to eliminate times when the bus stops. The most important attribute of bus performance is knowing when it is stopped at its designated stops. These problems are exacerbated for AVL with infrequent polling.

To combat the biases introduced by any graph smoothing technique, Dietmar Bauer, my collaborator, suggests explicitly modeling bus movement to re-sample the trajectories. Specifically, Bauer suggests using non-parametric techniques to better represent dwell time at stops.

Whereas linear interpolation and other smoothing techniques assume that the bus is always moving, Bauer takes the opposite approach and assumes that the buses do stop. The re-sampling method uses a combination of historical data and current observations. The historic data are used to see where along the shape a bus is more likely to be. The bus locations are sampled evenly in time, every 60 seconds, so the histogram of bus observations along the route reflect the probability of observing a

bus in that location. Logically then, buses spend more time in locations they are more likely to be; the probability of a bus being somewhere is inversely proportional to the dwell time of buses being in that segment.

These probabilities can then be used to reconstruct the actual velocity profile of an individual bus trajectory. The average travel velocities and dwell times close to stops (*i.e.*, within a region of 15m about the stop) can be inferred by using this proportionality. Even in cases where the trajectory does not have any observations near a stop, the probability function can be used to estimate the bus' dwell time at the stop. If there are observations near a stop, they can be used to more tightly constrain the most likely dwell time.

The historical information can be incorporated using a simple model for the progression of buses along the shape. Bauer suggests a simple two mode model for the bus trajectory: buses are either dwelling at a stop or traveling at constant speed (the buses do not accelerate). The model then fits the trajectory data as to the best possible re-sampled trajectory that incorporates only those two modes. The calculated parameters in this model are the trip start time, the dwell times at the stops, and the average velocities between each of the stops. These are the most important outputs from translating the location-at-time data to time-at-location data. The model finds the best fit between the raw trajectory observation and the historically most likely trajectory, with the important restriction that observed dwell times are always respected.

The model reconstructs the most likely trajectory of the trip. The fit of the trajectory is calculated by comparing the observed travel time between two consecutive AVL observations and the modeled travel time between those points in the shape. The model parameters are chosen so as to minimize the squared sum of these deviations plus the sum of squared deviations from the most probable stop dwell times, while respecting any observed stop dwell times. The solution to this least square problem thus delivers as an output the trajectory describing the progression of the bus along the shape which subsequently can be sampled at given time or location points. These re-sampled trajectories are used for most of the analyses in Chapter 4.

Chapter 4

Data Visualization and Analysis

Scagnetti and Schechtner (2010) suggest that, in most cases, the beauty and technical content of a visualization are inversely related to each other. Though my research may not yet transcend this axiom, I do approach the AVL data from both perspectives. Keeping the two separate, I provide visualizations that give an impressionistic account of the bus systems in Section 4.1 and provide new methods to analyze variations in bus travel times in Section 4.2.

4.1 Visualization

Often, the purpose of a data visualization is more to strike the viewer with awe and wonder for the data rather than to concretely convey insights from the data (Scagnetti and Schechtner 2010). Similarly, I present impressionistic displays of the AVL data in this Section.

Fry and Reas (2011) provide an invaluable tool for data visualization in the Processing language. All of these visualizations are built using the Processing language and use Nagel (2011)'s unfolding Processing library to render and manipulate map tiles. Many other visualizations require a simple static map that is individually manually downloaded rather than dynamically calling the tiles *ad hoc*. Because the unfolding library load the map tiles, the visualizations can be used for bus location data anywhere in the world without making adjustments to the programs other than

Figure 4-1: The reported positions and headings of all MBTA Route 1 buses at 7:30am on January 13, 2012. Map tiles ©OpenStreetMap contributors (2011, 2012), CC-BY-SA (CreativeCommons 2012).

loading in different data. It accommodates coordinates anywhere.

Section 4.1.1 presents a real-time visualization of bus locations and headings.

4.1.1 Real-time Visualization

Figure 4-1 displays an example of the real-time visualization capabilities. Each bus is represented by a black arrow pointed in the direction of the heading. This analysis does not require the map-matching or interpolation techniques discussed in Chapter 3; the bus locations are simply plotted on the map using their latitude-longitude coordinates.

This visualization, in addition to informing the viewer about the locations of buses, shows just how modular the NextBus data are. Figure 4-1 displays the positions of MBTA's route 1, but that is only because of the settings `agencyName = "mbta"` and `routeName = "1"`. Swapping `agencyName` for any of the tags in Table 2.1 and `routeName` for a valid route of that agency produces the logical result. Alternatively, setting `routeName = "all"` also produces the logical result without the complication of needing to know the specifics of, say, the Charm City Circulator's

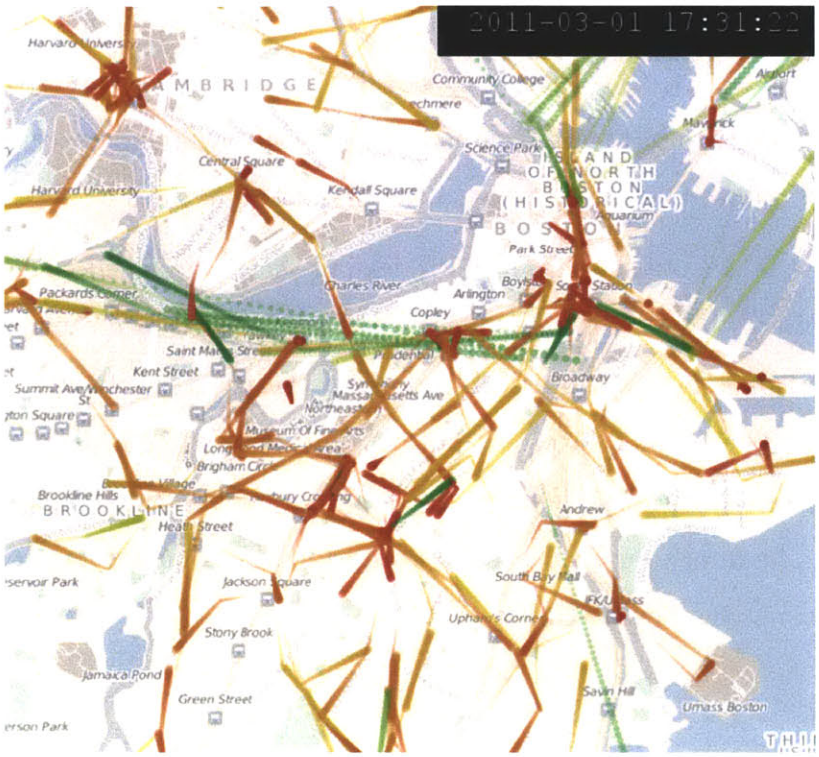

Figure 4-2: A depiction of the data for 1 March 2011 at 5:30pm across the MBTA network zoomed in to show central Boston and Cambridge. These data are raw AVL data. Buses are color-coded by speed, with red the slowest and green the fastest. The speeds are capped at 30kph, so the brightest green represents speeds of 30kph and higher. Map tiles ©OpenStreetMap contributors (2011, 2012), CC-BY-SA (CreativeCommons 2012).

routes. However, with large networks like the MBTA, displaying `"all"` buses can result in an indistinguishable mess of arrows.

4.1.2 System Visualization

The simplest visualizations of the archived data for MBTA's bus service do not require, though they would benefit from, map-matching. For example, as is shown in Figure 4-2, I can visualize all of the bus' trajectories and speeds only using their latitude-longitude positions.

Figure 4-2 depicts the data for 1 March 2011 at 5:30pm across the MBTA network zoomed in to show central Boston and Cambridge. These data are raw AVL data. Buses are color-coded by speed, with red the slowest and green the fastest.

The speeds are capped at 30kph, so the brightest green represents speeds of 30kph and higher. Generally, this visualization will show the buses in slightly inaccurate positions because they have not been map matched and will underestimate the speeds of the buses because they are unable to travel in straight lines.

Figure 4-2 is a snap-shot from a Processing sketch designed to visualize a day of bus trajectories. In the sketch, time progresses forward as a movie, but the user can zoom and pan the map interactively, pause the progress of time, and step time incrementally. The default is to zoom out only far enough to see all of the AVL points with the map centered on the center of the four extreme latitude-longitude coordinate boundaries. The image for Figure 4-2 was captured by zooming in to central Boston and then pausing the flow of time.

4.2 Analysis

In addition to visualizations, the map-matched and interpolated data can be used to analyze bus network performance. A wide body of research exists on using archived AVL data to understand and improve bus network performance. However, there are still areas where those methodologies could be improved.

For simplicity, I focus on one route for this Section. Specifically, all examples and analyses will be completed on MBTA's Route 1, which is depicted in Figure 4-3. Route 1 runs right through MIT, making it a favorite example of MIT transportation researchers. Route 1 is a high-frequency route with scheduled headways of 7.5 minutes during the evening peak period. These analyses were all completed using MATLAB, the only non-open source package used.

This Section is broken into two subsections. First, Section 4.2.1 reviews the literature on using archived AVL data to study bus performance and reproduces a few analyses using the NextBus MBTA AVL data. Second, Section 4.2.2 introduces new insights and analyses that my collaborators and I have developed.

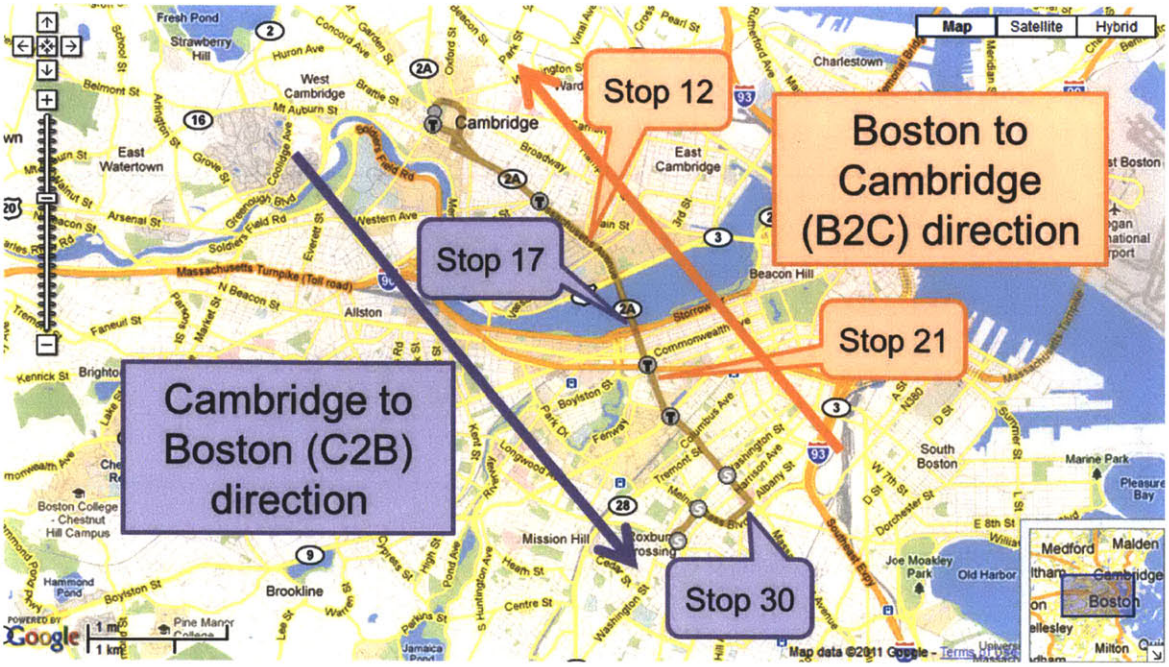

Figure 4-3: A map of MBTA Route 1 including the locations of key stops. Map generated by Baldwin (2006).

4.2.1 Traditional Analyses

Furth et al. (2006) provide an invaluable review of the analytic methods developed for archived AVL data through the time of its publication. Furth et al. (2006) divide bus service into two groups: scheduled service, where fewer than 6 buses service the route per hour; and walk-up service, where 6 or more buses service the route per hour. For scheduled service, adhering to the schedule is the most important metric because the buses come infrequently. For walk-up service, the most important attribute is maintaining even spacing between buses so as to minimize expected wait time; it is not too disruptive to passengers for a bus to leave a stop a few minutes early or late with regard to the schedule if there is another bus a few minutes behind it.

Furth et al. (2006) suggests that there are four different analyses that one can do with archived AVL data alone (*i.e.*, without AFC/APC data). First is run-time analysis, which looks at the distribution of the time it takes for buses to complete their trips. Second is schedule adherence; as could be expected, this only applies to scheduled service. Third is headway adherence, which predictably only applies to

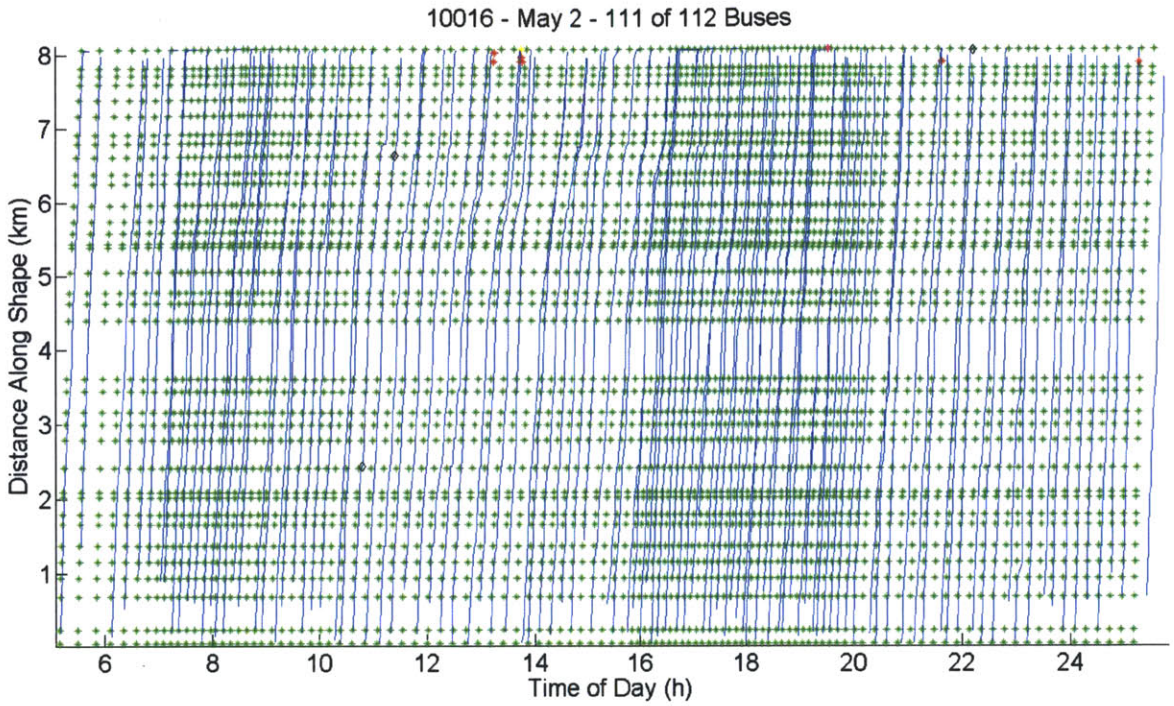

Figure 4-4: Space-time diagram of route 1 inbound on May 2, 2011. The blue lines represent the traces of buses and the green stars show scheduled bus stops.

walk-up service. Finally, passenger wait time can be calculated for walk-up service without passenger data if it is assumed that passengers arrive at the stops randomly.

The NextBus MBTA AVL data can be used for any of these analyses. One popular analysis that Furth et al. (2006) mentions is the space-time diagram, which can be used for both schedule and headway adherence analyses. Space-time diagrams plot the positions of buses throughout the day with distance along the shape as a function of time. Figure 4-4 shows the space-time diagram for MBTA Route 1. In Figure 4-4, we can see that there is poor headway adherence during the evening peak period, especially at 6pm (hour 18 of the day), as the bus trajectories are not evenly spaced. This phenomenon is generally called bunching. Bunching is undesirable because it increases the passengers' expected waiting time.

However, the space-time diagram has a few drawbacks. First, it captures only one day of service on one direction of one route; the space-time diagram is not appropriate for looking at general trends in service reliability across a month across the entire MBTA bus network. Even looking at the whole day of the space-time diagram is

a little daunting. Second, the space-time diagram does not endemically have any statistics in it that could alert a transit provider that one part of one space-time diagram is particularly interesting.

One important study published after Furth et al. (2006) is Byon et al. (2011). Byon et al. (2011) suggest an index for measuring how bunched a bus route is. Then, Byon et al. (2011) plot the index in a grid to span across the shape distance and time of day. By this point, it should be clear that AVL data are very high-dimensional. There is an observed headway for each stop of each trip of each direction of each route for each day of the year. The plot that Byon et al. (2011) suggest has four dimensions plotted: distance along the shape, time of day, bunchiness, and day of the year.

Liao and Liu (2010) provide a schematic for a computational system architecture and software tool for understanding bus network performance. This work is meant to be a competitor to the popular TriTAPT system developed by Delft University. Liao and Liu (2010) similarly have a SQL-based database server design. However, Liao and Liu (2010) use timepoint data (*e.g.* time-at-a-location data) rather than trajectory data. They generate interesting results from looking at the times when the bus needed to operate the wheelchair lift.

4.2.2 Analyzing Variation in Travel Time

In this section, I show analyses that my collaborators and I developed. These analyses were completed on the NextBus MBTA AVL data for the weekdays of May 2011 along Route 1.

Ideally, we would like to find a metric which can be mapped, producing both an informative and striking visualization of the AVL data. Then, we would like to study whether and how good and bad service spread through the bus network to see if and how good and bad service on one route affects service on another route. These analyses aspire to that ideal.

To look at bus performance metrics on a map, the data can only be two dimensional (distance along shape and a color), or, if played as a movie, three dimensional. Yet, even if there were a one-dimensional perfect statistic that captured the day-

to-day and diurnal variation in service levels along the shape, plotting it on a map would not be trivial because that distance along shape is also occupied by the other shapes of that route, and perhaps even the shapes of other routes. Displaying service performance for many routes on a map is problematic.

With regard to that goal, the results presented here are only preliminary and, hopefully, foundational. Yet, the results are interesting and insightful in their own right. In short, our analyses try to move beyond only identifying a service disruption toward identifying why service was disrupted and where the disruption started. Accordingly, the analyses focus on variations in travel times between stops.

Modeling the Spread of Travel-Times

This work was predominately done by my collaborator Peter Widhalm. In order to build a model for travel time prediction and planning, Widhalm analyzes the correlations between bus speeds along different segments of the route. In this analysis, Widhalm chops the shape into 10m segments and calculate the bus speed for each segment.

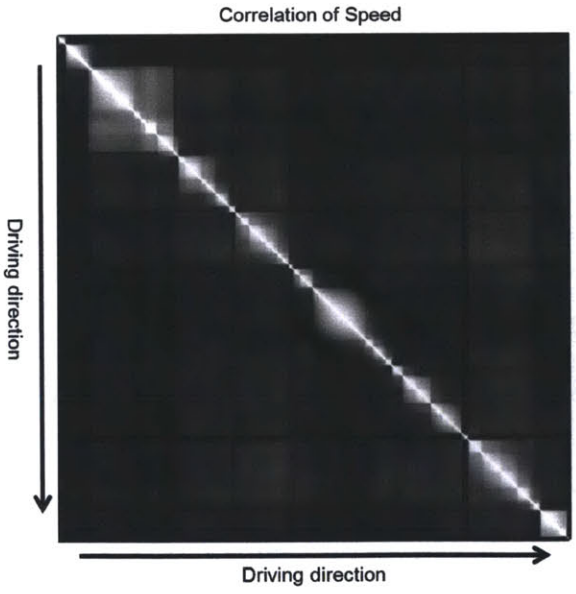

Figure 4-5: Correlation matrix of speeds on the MBTA Route 1 during the month of May 2011. White indicates high absolute correlation and black indicates low absolute correlation.

Figure 4-5, shows the correlation matrix for the Boston to Cambridge direction where white indicates high absolute correlation and black low absolute correlation. This Figure shows that there are discrete segments of the shape over which bus speed is highly correlated, seen in the white-to-gray squares. Interestingly, there is very little correlation between these blocks. This suggests that regions of the road have traffic patterns that are independent of one another. Also, the dark bands that separate different segments of high correlation are located at bus stops and stop lights. This makes sense; whether or not a bus has to stop at a stop light is independent of its speed anywhere else along the route; it is random. So, from the bus' perspective, the traffic state has reset each time that it goes through a major stop or stop light.

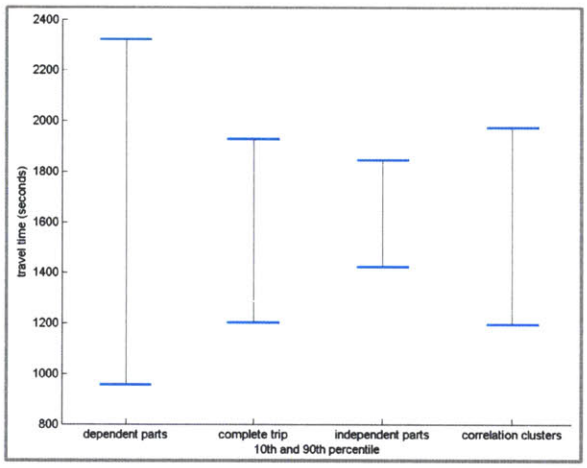

Figure 4-6: Spread of predicted travel times under different assumptions about the speed correlation structure.

Widhalm then sought to leverage the highly-correlated segments to make accurate predictions about the half-cycle time (time it takes to make one trip) for the route. Figure 4-6 shows the results. Figure 4-6 shows the spread (here as the difference between the 90th and 10th percentile values) of the half-cycle times under various assumptions: at the far left, assuming that every 10m segment of the shape is fully correlated with (dependent on) every other; next to the right, the observed spread of the half-cycle times; next to the right, assuming that every 10m segment of the shape is fully independent from every other; and at the far right, assuming only full correlation inside of the segments observed to be highly correlated in Figure 4-

5. This result suggests travel time variation can be well modeled using only local dependencies.

Travel-Time Matrix

Building on Widhalm's analysis, I try to determine the causes of travel-time variation. This effort is two-fold. First, I examine the variations in travel times between the stops of a route. Second, I look at the day-to-day variations in travel times between nearby stops.

First, I study the variations in travel times between the stops of a route in a diagram which I call the travel-time matrix. Creating the travel-time matrix involves many steps.

To make the travel-time matrix, I begin by calculating the observed travel times between each pair of `stop_sequence`s for each direction of the route. For routes with more than one shape for each direction, this can be somewhat more complicated, though they generally share most of the same `stop_id`s. If a shape has different stops than another shape, then the `stop_sequence`s can be used to determine the overlap of the shapes and appropriately re-number the `stop_sequence`s of the shapes. These calculations are completed with a simple SQL SELECT statement.

Next, the `stop_sequence`s from one direction are mapped onto the `stop_sequence`s of the other. Often, this is simple because two stops are directly across the street from each other and are not near any other stops, removing any space for ambiguity. Fittingly, the first pass at pairing stops is completed by searching for the `stop_id` from the opposite direction shape(s) that is nearest to each stop. These pairs are accepted if they are within a threshold distance and neither stop has competition to be in another pairing. Where this does not achieve a one-to-one pairing, whether because of multiply assigned stops or no assigned stops, I manually inspect the pairs and use judgment to assign pairs. The final pairs are then used to re-code the `stop_sequence`s for at least one of the directions. Often, both directions are recoded because at least one stop of each direction does not have a mate, and so the total number of stops (counting each pair as a single stop) in the route is larger than

the total number of stops in either direction.

Finally, I calculate the statistics. At this point, the data are arranged as records of a departure time from a stop, for example stop 12, and an arrival time at another stop, for example 2; they are a pair of stops and a departure and arrival time. For each pair of stops, I calculate the mean, trimmed mean, median, standard deviation, trimmed standard deviation, and spread of the travel times between them. I then refer to these as the 'global' statistics (*i.e.*, the global median) because they are calculated across the entire day. Then, I divide the day up into half-hour windows (*e.g.*, 5:15pm to 5:45pm) and calculate the same statistics for each pair of stops. Though the standard deviation of travel times between 5:15pm and 5:45pm can be understood without context, though some suggest using the coefficient of variation instead, the mean travel time between stops during the period from 5:15pm to 5:45pm requires context. So, the global mean, median, and trimmed mean are subtracted each of their respective time-window counterparts.

It may be possible to look at the travel times with respect to the scheduled travel times; however, these analyses were generally not as fruitful. Specifically, the travel-time matrices calculated with respect to scheduled travel times did not exhibit the kind of structure that these travel-time matrices have. Rather, they only show that longer trips are generally later (trip length is proportional to lateness), not that a specific stop or part of the route disproportionately contributes to delays.

Here, I show two pairs of travel-time matrices. Each matrix shows the value of a statistic through color for each from-to stop combination. The 'from' stops are shown on the horizontal axis, and 'to' stops are shown on the verticle axis. Figures 4-7 and 4-8 show the trimmed mean and trimmed standard deviation for travel-times during the half-hour around 9:00pm, and Figures 4-9 and 4-10 show the same for the half-hour around 5:30pm. In each Figure, the direction from Cambridge to Boston, as shown in the lower-left corner of Figure 4-3, is in the upper-left half of the travel-time matrices and vice versa. That is, the diagonal from lower left to upper right of blank squares is the dividing line between traveling in one direction and the other. These blank data points represent the travel times from stop 1 to stop 1, and so on.

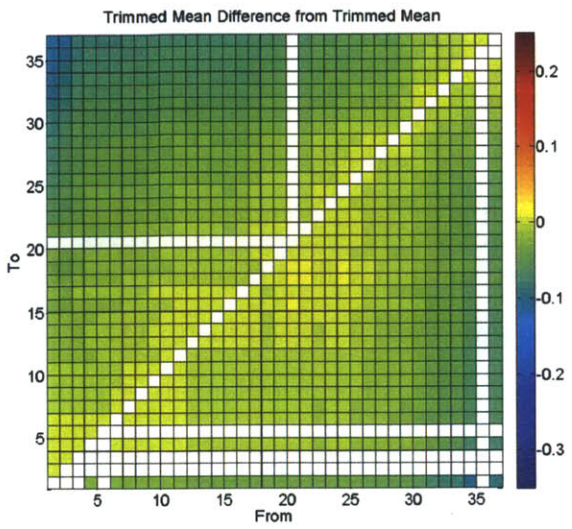

Figure 4-7: The travel-time matrix of the difference between the trimmed mean during 8:45pm to 9:15pm and the trimmed mean across the entire day. Time is reported in hours.

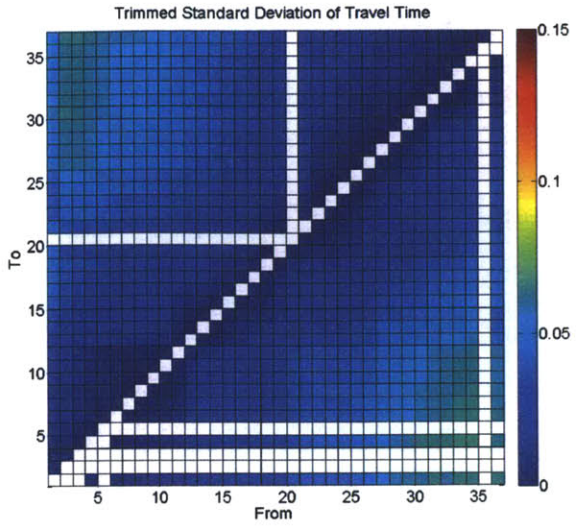

Figure 4-8: The standard deviation of the travel-time during the same time period. Time is reported in hours.

Other stop combinations are blank because a stop exists in one direction but not the other. In each pair of travel time plots, I show the difference between trimmed means (during this time period and all day) and the standard deviation of trimmed travel times.

The pair of Figures 4-7 and 4-8 show the travel time variability during off-peak travel, specifically between 8:45pm and 9:15pm. There, the travel-times are about average; the difference between the trimmed-mean in this period and the global trimmed-mean is about zero for all stop pairs in both directions, with some of the longer trips (those that are farther from the diagonal dividing directions) even clocking in a little faster than average. Additionally, there is relatively little variation in the travel times during this period, though the variability does increase for the longer trips, which is logical. Further, there is a little structure in the variability matrix, with the variability suddenly increasing at certain stops, seen as rectangles of similar colors in Figure 4-8.

However, the travel-times are much more interesting during the peak period. For the peak period, I use the half hour around 5:30pm, as shown in Figures 4-9 and 4-10.

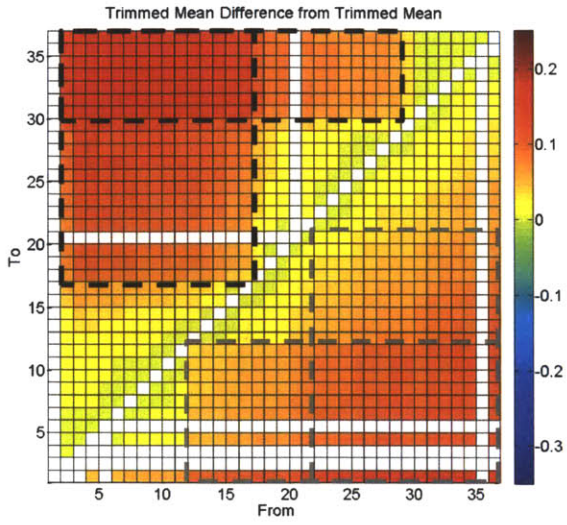

Figure 4-9: The travel-time matrix of the difference between the trimmed mean during 5:15pm to 5:45pm and the trimmed mean across the entire day. Time is reported in hours.

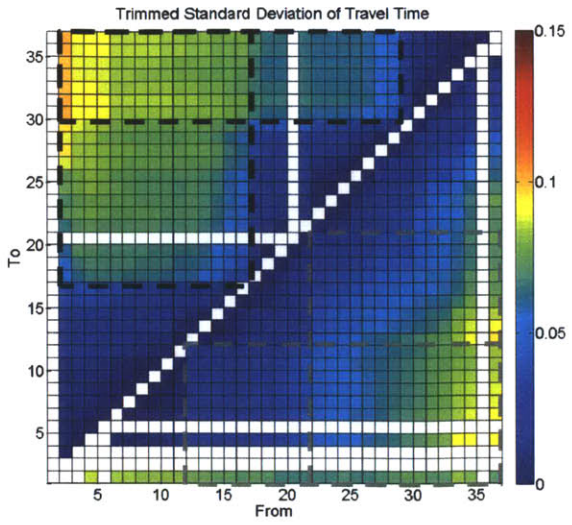

Figure 4-10: The standard deviation of the travel-time during the same time period. Time is reported in hours.

These figures show that service disruptions (marked by the black dashed boxes) in the Cambridge to Boston begin at stops 17 and 30. Similarly, in the Boston to Cambridge direction, travel times are significantly affected after stops 12 and 21 (marked by gray dashed boxes). Examining Figure 4-3, it shows that stops 17 and 30 in the Cambridge to Boston direction are located just over the Harvard Bridge into Boston and at the Boston Medical Center, respectively. Stops 12 and 21 in the Boston to Cambridge direction are located at Hynes Convention Center and MIT, respectively. The structure seen in both the trimmed mean and trimmed standard deviation plots at these points indicate that the buses are on average being significantly slowed with respect to the average travel time, though not always at the same magnitude.

Though the standard deviation of the travel times is high, this does not necessarily mean that the route has uneven headways because there is not information about the autocorrelation between successive headways. These stops are leverage points in Route 1 where bus service disruptions are introduced. However, as these data are averaged across the entire month of May, it is not possible to determine on what time scale these disruptions occur. This again raises the complications of working

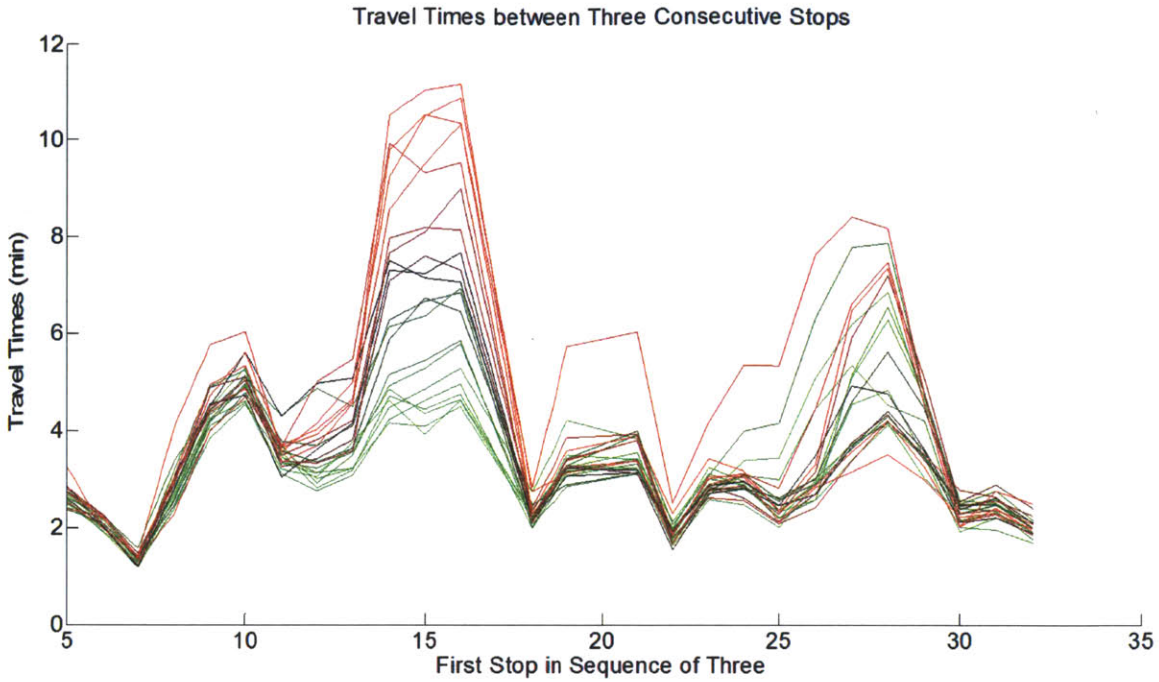

Figure 4-11: Mean daily travel-times between three consecutive stops on Route 1 during the evening peak period of 4:30pm to 6:30pm for May 2011.

with such high-dimensional data. I can, however, determine the time scale of the disruptions by pairing these plots with another set of plots.

Travel-Time Variability by Day

Here I focus on the Cambridge to Boston direction, especially the disruption observed at stop 17 during the evening peak period. Specifically, I look at the daily service variability over May 2011 weekdays. In order to make the statistics more robust, I increase the time period to 4:30pm to 6:30pm. There are 16 trips scheduled for this period each weekday. Outliers were removed.

For this analysis, I focus on short trips just off of the diagonal so as to focus on stop 17. Specifically, I look at the travel time between three consecutive stops. In Figure 4-11, the daily average travel time between the stops is shown as a function of the first stop in the sequence of three. For example, the value for stop 5 reflects the travel time from stop 5 to stop 8. Each line captures the average travel time for each weekday of May. The day colors are assigned from red to green as ordered by

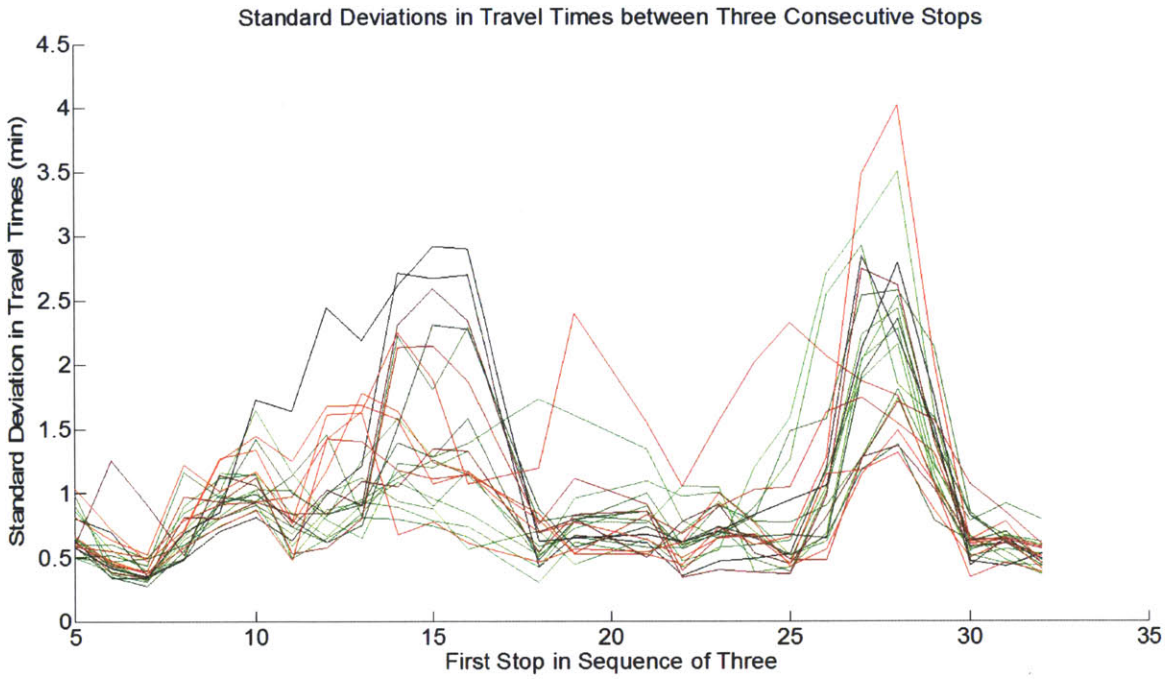

Figure 4-12: Daily standard deviation of travel-times between three consecutive stops on Route 1 during the evening peak period of 4:30pm to 6:30pm for May 2011.

the observed average travel time from stop 15 to stop 18, with the longest travel time the brightest red and shortest travel time the brightest green. The days hold this assigned color through Figures 4-12 and 4-13.

In Figure 4-11, there is a noticeable jump in absolute travel time and the spread of the daily travel times for traveling between stops 14 and 17. The travel times fall again and fall back together only at the travel time between stops 18 and 21. There is a similar, though less pronounced, split at stop 30. Interestingly, as was seen in Widhalm's work in Section 4.2.2, travel times in one region of the route do not necessarily predict travel times elsewhere; though the days are sorted by color at stop 15, they do not retain this sorting anywhere else in the shape.

Figure 4-12 shows the daily standard deviation of travel times in the evening peak period. Each day is captured as a single line, colored as in Figure 4-11. Comparing Figures 4-11 and 4-12, it is clear high standard deviations do not necessarily mean high local travel times. Further, there is not a universal bump in standard deviation approaching stop 17, whereas there is a universal bump at stop 30. Together, this suggests that perhaps there are certain days when there are large service disruptions

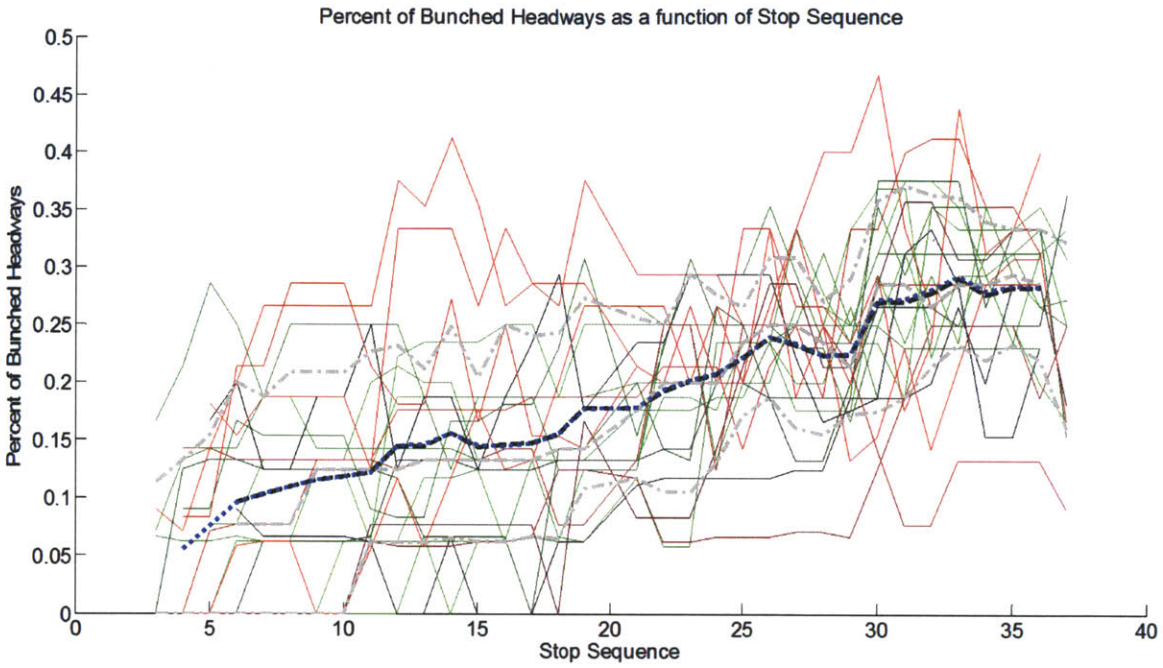

Figure 4-13: The percentage of bunched headways as a function of the stop along the route separated by day. Headways are considered to be bunched if they are less than or equal to two minutes.

around stop 17, and not that bunching significantly increases at stop 17. We can investigate this by looking at the bunchiness at each stop individually.

Looking again at bunching as a possible cause of disruptions at stops 17 and 30, I consider the daily headways of the same cohort of buses at each stop in Figure 4-13. Here, the days are colored similarly as in Figures 4-11 and 4-12. In Figure 4-13, I show the percentage of bunched headways as a function of stop sequence. I define a headway to be a bunched headway if the observed headway is two minutes or less (26.7% of the scheduled headway). Each solid line is a day. Additionally, I calculate aggregate statistics across the days: the mean in dashed black, median in dashed gray, and 15th- and 85th percentiles in dash-dot gray. We also show the average percentage across all observations independent of day in the dotted blue. This line is different because some days have different numbers of trips.

Here in Figure 4-13, bunching increases loosely linearly as the bus moves along the route. However, there are not any points where service consistently degrades daily, whether due to a systematic increase in the travel time or a systematic increase in

the variability of travel times.

4.3 Analysis Discussion and Conclusion

Ultimately, this collection of figures shown in Sections 4.2.2 and 4.2.2 summarize many aspects of the service disruptions experienced by the MBTA Route 1 during the month of May 2011. These figures are successful in that they manage to display the variation in the data across many different dimensions. The figures show variation in bus performance along the shape, through the diurnal cycle, and in daily differences. Additionally, the figures analyze both absolute degradation of travel time and travel time variability across each of these cuts. These figures are not successful in that there are so many of them; indeed the travel-time matrices shown here are a small subset of the total number of generated travel-time matrices. Each figure requires concentrated effort to understand and interpret, and, moreover, the figures are best understood in aggregate. That is, while these figures are powerful for understanding variability in bus travel times, they do not achieve the goal of finding a one-dimensional parameter to plot on a map. Indeed, the results of these figures do not even obviously lend themselves to automated interpretation.

Chapter 5

Conclusion

This thesis expands the usefulness of bus AVL data by demonstrating how publicly available AVL data, such as the NextBus MBTA AVL data used here, can be used to evaluate bus performance. Using these data lowers the entry cost to doing AVL research, ultimately enabling studies that span multiple cities and a large timescale.

The thesis discusses the datasets used. I demonstrate that the NextBus AVL data, which capture the bus' location as a GPS coordinate at set intervals, are best used for evaluating bus service, and that, resultantly, GTFS data are also necessary. Further, I provide a useful primer for using GTFS and NextBus AVL data.

Next, I successfully integrate the AVL data and GTFS data. My map-matching algorithm is able to robustly handle a wide range of erroneous behavior that the GPS trajectories may exhibit. The map-matching algorithm and its computational architecture, built in Java, are robust and fast. Errors in map-matching can be quickly and clearly investigated using a tool developed in Processing. With my collaborators, I introduce a new method for inferring bus stop arrival and departure times using a non-parametric model.

Finally, I present new analytic methods for bus AVL data. Some of these analyses are deeply technical, others only leave the viewer with an interesting new perspective of bus movement. The technical analyses demonstrate a suite of analyses that can be used to diagnose points along a route that disproportionately contribute to service disruptions.

This thesis demonstrates the power and versatility of low-frequency bus AVL data to understand variations in bus transport travel times. The thesis uses Bostons MBTA bus service as a case study, presenting here the results for Route 1. I show that the AVL data can be used to find parts of the route on which service deteriorates. Additionally, I am able to identify regions of the route wherein service is very stable and highly predictable. Travel time distributions are very accurately reconstructed by only using dependence on travel time within these regions. Further, I am able to investigate possible reasons for service to deteriorate in those regions.

In the future, researchers should expand these analyses to the entire MBTA network to detect weak points of the system. Future research could find critical stops that cause the system to collapse and then propose and simulate improvement strategies.

Additionally, future research could expand beyond examining bus performance with respect to current ridership to examining bus performance with respect to potential ridership. Wu and Murray (2005) use population density data to evaluate the Columbus, OH bus network's ability to effectively satisfy travel demand. This model could be improved by integrating both AVL data and cell phone tracking data. With accurate passenger origins and destinations from the cell phone data and accurate bus performance data from the AVL, the research presented here could be used to evaluate how effectively the Boston bus network distributes travelers.

With this information, future research could better understand the disaggregated (*i.e.* individual level) demand for transportation in the greater Boston area and then propose adjustments to the existing system those satisfy those demands. Similarly, potential riders can then better understand how well the system can meet their requirements theoretically increasing the likelihood that they will chose public transportation.

Bibliography

Gennady Andrienko, Natalia Andrienko, and Stefan Wrobel. Visual analytics tools for analysis of movement data. *SIGKDD Explor. Newsl.*, 9:38–46, December 2007. ISSN 1931-0145. doi: http://doi.acm.org/10.1145/1345448.1345455. URL `http://doi.acm.org/10.1145/1345448.1345455`.

Laura Baldwin. MBTA Bus Routes (work in progress), 2006. URL `http://web.mit.edu/boojum/www/Map/`.

Dave Barker. Interview with Mr Barker, Manager of Operations Technology at MBTA, 19 September 2011.

James Biagioni, Tomas Gerlich, Merrifield, and Jakob Eriksson. EasyTracker: Automatic Transit Tracking, Mapping, and Arrival Time Prediction Using Smartphones. In *9th ACM Conference on Embedded Networked Sensor Systems (SenSys)*, Seattle, WA, November 2011.

Colin Bick and Robert Damphouse. Gtfs sql import tool, 2010. URL `http://cbick.github.com/gtfs_SQL_importer/html/index.html`.

François Bonzon. Internet-based tracking of gps-equipped buses. Sin, École Polytechnique Fédérale de Lausanne, TOPO Laboratory, February 2007 2007.

Candace Elizabeth Brakewood. Contactless Bankcards and Prepaid Cards in Transit Fare Collection Systems. Master of Science in Transportation and Master of Science in Technology and Policy, Massachusetts Institute of Technology, Department of Civil and Environmental Engineering and the Engineering Systems Division, May 2010.

Young-Ji Byon, Cristian E. Cortes, Francisco Javier Martinez, Marcela Munizaga, and Mauricio Zuniga. Transit Performance Monitoring and Analysis with Massive GPS Bus Probes of Transantiago in Santiago, Chile: Emphasis on Development of Indices for Bunching and Schedule Adherence. In *TRB 2011 Annual Meeting Compendium of Papers CD-ROM*, 2011.

Tom Carden. ModestMaps for Processing, 2011,2012. URL `https://github.com/randomEtc/modestmaps-processing`.

F.W. Cathey and D.J. Dailey. Transit vehicles as traffic probe sensors. In *2001 IEEE Intelligent Transportation Systems Conference Proceedings*, pages 579–584, Oakland, CA, August 2001.

F.W. Cathey and D.J. Dailey. A prescription for transit arrival/departure prediction using automatic vehicle location data. *Transportation Research Part C*, 11:241–264, 2003.

Kun Chen, Lei Yu, Jifu Huo, and Huimin Wen. Characteristics analysis of road network reliability in beijing based on the data logs from taxis. In *TRB 86th Annual Meeting Compendium of Papers CD-ROM*, 2007.

Peter Pin-shan Chen. The entity-relationship model: Toward a unified view of data. *ACM Transactions on Database Systems*, 1:9–36, 1976.

Chicago Transit Authority. Bus Tracker API Documentation, 2011. URL `http://www.transitchicago.com/assets/1/developer_center/BusTime_Developer_API_Guide.pdf`.

CreativeCommons. Attribution-ShareAlike: CC BY-SA, 2012. URL `http://creativecommons.org/licenses/by-sa/3.0/`.

Luca D'Acierno, Armando Carten, and Bruno Montella. Estimation of urban traffic conditions using an automatic vehicle location (avl) system. *European Journal of Operational Research*, 196(2):719 – 736, 2009. ISSN 0377-2217. doi: 10.1016/j.ejor.2007.12.053. URL `http://www.sciencedirect.com/science/article/pii/S0377221708003482`.

Corrado De Fabritiis, Roberto Ragona, and Gaetano Valenti. Traffic estimation and prediction based on real time floating car data. In *2008 11th International IEEE Conference on Intelligent Transportation Systems*, pages 197–203. IEEE, 2008. URL `http://ieeexplore.ieee.org/lpdocs/epic03/wrapper.htm?arnumber=4732534`.

Wen Dong and Alex Pentland. A network analysis of road traffic with vehicle tracking data. In *AAAI Spring Symposium*, pages 7–12, 2009.

Brian Ferris. *OneBusAway: Improving the Usability of Public Transit.* PhD dissertation, University of Washington, Computer Science and Engineering, 2011.

B. Fry and C. Reas. Processing library for visual arts and design, May 2011. URL `http://processing.org`.

Peter G. Furth, Brendon Hemily, Theo H.J. Muller, and James G. Strathman. *TCRP Report 1133: Using Archived AVL-APC Data to Improve Transit Performance and Management.* TRB, Washington D.C., 2006.

Marta C. González, César A. Hidalgo, and Albert-László Barabási. Understanding individual human mobility patterns. *Nature*, 453(7196):779–782, June 2008. ISSN 0028-0836. doi: 10.1038/nature06958. URL `http://dx.doi.org/10.1038/nature06958`.

Google, 2011. URL `http://code.google.com/transit/spec/transit_feed_specification.html`.

Ramaswamy Hariharan and Kentaro Toyama. Project lachesis: Parsing and modeling location histories. In Max J. Egenhofer, Christian Freksa, and Harvey J. Miller, editors, *Geographic Information Science, Third International Conference, GIScience 2004, Adelphi, MD, USA, October 20-23, 2004, Proceedings*, volume 3234 of *Lecture Notes in Computer Science*, pages 106–124. Springer, 2004. ISBN 3-540-23558-2. doi: http://springerlink.metapress.com/openurl.asp?genre=article&issn=0302-9743&volume=3234&spage=106.

Shuyan He, Wei Guan, Wei Qiu, Lan Wang, and Jihui Ma. Link travel time estimation at signalized road segments with floating car data. In Baohua Mao, Zongzhong Tian, Haijun Huang, and Ziyou Gao, editors, *ASCE Conf. Proc.*, number 322, pages 880–889, Nanning, China, 2008. ASCE. doi: doi:10.1061/40995(322)83.

Bruce Hellinga, Pedram Izadpanah, Hiroyuki Takada, and Liping Fu. Decomposing travel times measured by probe-based traffic monitoring systems to individual road segments. *Transportation Research Part C*, 16:768–782, 2008. doi: 10.1016/j.trc.2008.04.002.

Britta Hummel. *Dynamic and Mobile GIS: Investigating Changes in Space and Time*, chapter Map Matching for Vehicle Guidance, page Ch. 10. CRC Press, 2006.

ITS Vienna Region. AnachB.at, 2012. URL `http://www.anachb.at/`.

Chen-Fu Liao and Henry X. Liu. Development of a Data Processing Framework for Transit Performance Analysis. In *TRB 2010 Annual Meeting Compendium of Papers CD-ROM*, 2010.

Lin Liao, Dieter Fox, and Henry Kautz. Extracting places and activities from gps traces using hierarchical conditional random fields. *Int. J. Rob. Res.*, 26:119–134, January 2007a. ISSN 0278-3649. doi: 10.1177/0278364907073775. URL `http://dl.acm.org/citation.cfm?id=1229555.1229562`.

Lin Liao, Donald J. Patterson, Dieter Fox, and Henry Kautz. Learning and inferring transportation routines. *Artif. Intell.*, 171:311–331, April 2007b. ISSN 0004-3702. doi: 10.1016/j.artint.2007.01.006. URL `http://dl.acm.org/citation.cfm?id=1238140.1238288`.

MBTA. Massachusetts Bay Transit Authority General Transit Feed Specification data, 2012. URL `http://www.mbta.com/uploadedfiles/MBTA_GTFS.zip`.

Till Nagel. unfolding: interactive maps for processing, July 2011. URL http://code.google.com/p/unfolding/. Newer verions are available.

NextBus. Public xml feed, August 2011. URL http://www.nextbus.com/xmlFeedDocs/NextBusXMLFeed.pdf.

Austin Louis Oehlerking. Streetsmart: Modeling vehicle fuel consumption with mobile phone sensor data through a participatory sensing framework. Master of science in mechanical engineering, Massachusetts Institute of Technology, Department of Mechanical Engineering, September 2011.

OpenStreetMap contributors. Openstreetmap tiles, 2011, 2012. URL http://www.openstreetmap.org/.

Doug J. Parker. *TCRP Synthesis 73: AVL Systems for Bus Transit: Update.* TRB, Washington D.C., 2008.

PostgreSQL. PostgreSQL, 2011. URL http://www.postgresql.org/.

Mohammed A. Quddus, Washington Yotto Ochieng, Lin Zhao, and Robert B. Noland. A general map matching algorithm for transport telematics applications. *GPS Solutions*, 7(3):157–167, December 2003. ISSN 1080-5370. doi: 10.1007/s10291-003-0069-z. URL http://dx.doi.org/10.1007/s10291-003-0069-z.

Mohammed A. Quddus, Washington Y. Ochieng, and Robert B. Noland. Current map-matching algorithms for transport applications: State-of-the art and future research directions. *TRANSPORTATION RESEARCH PART C-EMERGING TECHNOLOGIES*, 15(5):312–328, OCT 2007. ISSN 0968-090X. doi: {10.1016/j.trc.2007.05.002}.

Kurk Raschke. Open standards for avl and other real-time transit data. *Raschke on Transport*, 12 April 2011. URL http://transport.kurtraschke.com/2011/04/open-standards-for-avl.

Refractions Research. PostGIS, 2011. URL http://postgis.refractions.net/.

Gaia Scagnetti and Katja Schechtner. Mobility Patterns in Cities - Information Visualization. Austrian Institute of Technology Internal Report, December 2010.

Matthew Thomas Shireman. Using Automatically Collected Data for Bus Service and Operations Planning. Master of Science in Transportation, Massachusetts Institute of Technology, Department of Civil and Environmental Engineering, January 2011.

Chaoming Song, Tal Koren, Pu Wang, and Albert-Laszlo Barabasi. Modelling the scaling properties of human mobility. *Nature Physics*, 6(10):818–823, September 2010. ISSN 1745-2473. doi: 10.1038/nphys1760. URL http://dx.doi.org/10.1038/nphys1760.

SQLPOWER. Sql power architect 1.0.6, Jan 2011. URL http://code.google.com/p/power-architect/.

Arvind Thiagarajan, Lenin S. Ravindranath, Katrina LaCurts, Sivan Toledo, Jakob Eriksson, Samuel Madden, and Hari Balakrishnan. VTrack: Accurate, Energy-Aware Traffic Delay Estimation Using Mobile Phones. In *7th ACM Conference on Embedded Networked Sensor Systems (SenSys)*, Berkeley, CA, November 2009.

W. Toplak, H. Koller, M. Dragaschnig, D. Bauer, and J. Asamer. Novel road classifications for large scale traffic networks. In *Intelligent Transportation Systems (ITSC), 2010 13th International IEEE Conference on*, pages 1264 –1270, sept. 2010. doi: 10.1109/ITSC.2010.5625182.

Trapeze ITS. On-board systems, 2011. URL http://www.trapezeits.com/solutions-on-board-systems.php.

R. Trasarti, S. Rinzivillo, F. Pinelli, M. Nanni, A. Monreale, C. Renso, D. Pedreschi, and F. Giannotti. Exploring real mobility data with m-atlas. In *Proceedings of the 2010 European conference on Machine learning and knowledge discovery in databases: Part III*, ECML PKDD'10, pages 624–627, Berlin, Heidelberg, 2010. Springer-Verlag. ISBN 3-642-15938-9, 978-3-642-15938-1. URL http://dl.acm.org/citation.cfm?id=1889788.1889838.

TriMet. Making a difference: Bibiana mchugh, 2011. URL http://trimet.org/difference/bibi.htm.

Henk J Van Zuylen, Yusen Chen, and Fangfang Zheng. Using floating car data for traffic state estimation in signalized urban networks. In *International Workshop on Traffic Data Collection and its Standardisation*, Barcelona, 2008.

Webtech Wireless. Webtech wireless homepage, 2011. URL http://www.webtechwireless.com/.

Changshan Wu and Alan T Murray. Optimizing public transit quality and system access: the multiple-route, maximal covering/shortest-path problem. *Environment and Planning B Planning and Design*, 32(2):163–178, 2005. URL http://www.envplan.com/abstract.cgi?id=b31104.

Dongdong Wu, Tongyu Zhu, Weifeng Lv, and Xin Gao. A heuristic map-matching algorithm by using vector-based recognition. In *Proceedings of the International Multi-Conference on Computing in the Global Information Technology*, pages 18–, Washington, DC, USA, 2007. IEEE Computer Society. ISBN 0-7695-2798-1. doi: 10.1109/ICCGI.2007.3. URL http://dl.acm.org/citation.cfm?id=1259588.1259856.

Jae-seok Yang, Seung-pil Kang, and Kyung-soo Chon. The map matching algorithm of gps data with relatively long polling time intervals. *J. of the Eastern Asia Soc. for Trans. Stud.*, 6:2561–2573, 2005.

Nickolai Zeldovich. Google groups post: Inferring routes from real-time data, 2010. URL http://groups.google.com/group/massdotdevelopers/browse_thread/thread/cd5d26f5be498a98?hl=en.