The Time Space Diagram Revisited

Afian Anwar (Corresponding Author)
Computer Science and Artificial Intelligence Lab
Massachusetts Institute of Technology
32 Vassar Street
Cambridge, MA 02139
afian@csail.mit.edu

Wei Zeng
Future Cities Laboratory
Swiss Federal Institute of Technology Zurich
1 CREATE Way
06-01 CREATE Tower
Singapore 138602
zeng@arch.ethz.ch

Stefan Müller Arisona
Future Cities Laboratory
Swiss Federal Institute of Technology Zurich
1 CREATE Way
06-01 CREATE Tower
Singapore 138602
arisona@arch.ethz.ch

4068 words + 7 figures + 0 tables
November 15, 2013
ABSTRACT

Widely used in the design and analysis of transportation systems, time-space diagrams were developed in an era of data scarcity, when it was necessary to obtain data by means of driver logs, human observers or aerial photographs. In this paper we show how time-space diagrams still remain relevant today, in an era of data abundance. We present an application that efficiently encodes the trajectories of bus GPS data in a time-space cube and uses simple geometric methods to calculate and visualize the headways and separation of buses on a bus route. We discuss these methods in detail and explore how they can be used as the basis of a software package to monitor performance measures for a variety of applications.
INTRODUCTION

Time-space diagrams are commonly used to solve a wide variety of transportation problems. Typically, we use the variable $x$ to denote the distance traveled along a guideway from some arbitrary reference point, and another variable $t$ to denote the time elapsed from an arbitrary instant \(^{(1)}\). By studying how the position of a vehicle changes over time, we can better understand the performance characteristics of the transportation system under analysis.

The biggest challenge in using a time-space diagram is collecting sufficiently accurate data to build one \(^{(2)}\). Time-space diagrams came into prominence in the 1950s and 1960s when transportation engineering was in its infancy. Back then, data collection was tedious and expensive. Today, it is standard practice for transport operators to know where all their vehicles are at all times. For example, bus operators routinely use bus location data to inform passengers when the next bus is arriving while taxi operators use real time taxi GPS data in their booking systems.

All this data is location based, and contains at the bare minimum a position (latitude and longitude) and time stamp. In this paper, we propose using such data to build time-space diagrams for use in the analysis and visualization of transportation operations. The main contributions of this paper are:

- a way to efficiently store vehicle trajectories as line segments in 3D space for fast retrieval
- simple geometric methods inspired by the time-space diagram that apply intersection tests with these line segments to find vehicle headways and spacing,
- a visualization and analysis tool that can allow transit operators and regulators to better understand network performance and behavior,
- a brief discussion on how any transportation system with big data capability can use time space diagrams to monitor service levels, network capacity and delays

Related Work

Time-Space Diagrams

Time-space diagrams study how vehicles move in time, and are regularly used to determine the maximum throughput of airport runways \(^{(3)}\), devise optimal train schedules \(^{(4)}\) and coordinate traffic lights \(^{(5)}\).

We find it convenient to plot $x$ on the x-axis and $t$ on the y-axis as in Figure 1, which shows the trajectories of 3 vehicles, $a$, $b$ and $c$ on a straight road. By plotting the position of vehicles along a guideway over time to obtain the function $x(t)$, the displacement $x$ of the vehicle for every time $t$, we can easily deduce headways (when viewed from a fixed position) and separation (when viewed from a fixed point in time). The inverse of the slope of each trajectory gives you the vehicle’s speed, while its curvature, its acceleration. An increasing slope indicates that the vehicle is decelerating while a decreasing slope shows that the vehicle is accelerating. As noted in \(^{(6)}\), a vertical line through the diagram identifies the times at which successive vehicles pass a stationary observer, and a horizontal line identifies the vehicle positions at the given time. The times between consecutive vehicle observations at a fixed location are called headways ($h_{bc}$ in Figure 1), and the distance separation between consecutive vehicles at a given instant, spacings ($s_{bc}$ in Figure 1).
FIGURE 1: Time-space diagram of three vehicles a, b and c. A vertical line for a at position $x_1$ shows that the vehicle a has come to a complete stop. When trajectories intersect as c does with b at position $x_2$, it means that vehicle c has overtaken vehicle b. The horizontal distance $s_{bc}$ between trajectories b and c at time $t_1$ tells us the spacing while the vertical distance $h_{bc}$ between b and c at position $x_3$, the headway.

1 **Time-Space Cubes**

2 Thus far our time-space diagram and methods of analysis are limited to two dimensions, time $t$ and distance $x$. By adding a third dimension $y$ to represent the distance travelled in the $y$ direction, we get a time-space cube. Time-space cubes were first developed by Hägerstrand to model the time-geography of individual movement, and have been used primarily by urban planners and to understand the social interactions of people and place. In its typical form, the cube has on its base a representation of geography e.g. a city map, while the cube’s height represents time (z-axis) (7).

3 As in the time-space diagram, trajectories are represented by joining together $x$, $y$ and $t$ points belonging to the same vehicle so as time progresses, the trajectory moves upwards in time.

4 Each vertical “slice” of the time-space cube along the vehicle’s path shows its trajectory in 2D, thereby illustrating that the time-space cube is a complete summary of the progress of our vehicle as captured by the time-space diagram (Figure 2).

5 Our work extends on prior art by first using line segments in a time-space cube to represent the observed trajectories of buses in Singapore and next, making vertical and horizontal cuts on the cube’s constituent time-space diagrams to efficiently find vehicle headway and spacing. Unlike previous work (8) which used headway and spacing data collected at selected geographic locations, our approach allows us to calculate performance metrics at any arbitrary point on the bus route and see how they vary over time.

6 **Outline**

7 We motivate our approach and describe the data we use for this study in Sections 3 and 4. Section 5 formulates the problem, defines notation and suggests a way to efficiently store and query our data using a spatial index. In Section 6, we show how our methods, packaged in an app, could be used to solve the real world problem of bus bunching by allowing users to compute and visualize bus headways and spacing easily and intuitively. Finally in Section 7, we consider how our work could
FIGURE 2: A bus enters the time-space cube at time $T_1$ and leaves at time $T_2$. The time it spends in the cube is given by $T_2 - T_1$. The trajectory of the bus through time-space is represented by the solid grey line while its actual path can be found by projecting its time-space path onto the 2D base of the cube.

be used in any transportation system with big data capability to monitor service levels, network capacity and delays.

3 MOTIVATION
Transport regulators are often held accountable for the level of service and reliability of the transport operators under their purview. In Singapore, the Land Transport Authority (LTA) plans routes and establishes minimum service standards for bus lines operated by the Singapore Bus Service (SBS) and Singapore Mass Rapid Transit (SMRT). The LTA is responsible for both ensuring that these minimum service standards are met and investigating complaints from the general public. This has traditionally required the use of observers to survey bus passenger loads, frequency and reliability, an expensive and time consuming task. In this paper, we show how we can use Big Data to replace these surveys with a visual analysis tool that uses time-space diagrams to capture the performance metrics for an entire bus network with greater accuracy, lower cost and at scale.

DATA
The bus dataset (2 GB) we used was made available through a recent initiative by the LTA to monitor bus ridership. It contains three months (June, July and August 2013) of bus GPS and passenger load records for each of Singapore’s 400 bus lines collected automatically by LTA’s
Each record contains the time-stamp, bus route ID, bus ID, bus stop ID of the current bus stop, direction (inbound or outbound), maximum capacity and number of passengers on each bus. Records are logged each time the bus arrives at a bus stop, thereby allowing us to track the exact position of each active duty bus over the course of a day. Crucially, most passengers in Singapore use smart cards (9) when getting on and off buses, thereby allowing us to know approximately how many people are on board. Since each record contains the unique ID of the bus that it belongs to and the bus stop ID of the bus stop the bus is at, we can cross reference this with an index of the GPS locations of each bus stop to reproduce the bus’s trajectory.

FIGURE 3: Bus headways and separation visualized. Consider two buses, \(a\) and \(b\), with bus \(b\) ahead of bus \(a\) on a straight road segment. The vertical line at a given abscissa identifies vehicles at a given time (represented by a horizontal plane). The spacing between two buses \(s_{ab}(t') = X_b - X_a\) at a particular time \(t'\) is found by taking the difference in abscissa of the intersection points of the bus trajectories and the horizontal plane. Similarly, the horizontal line at a specified ordinate identifies the time at which a vehicle passes a location (represented by a vertical plane). The headway \(h_{ab}(x') = T_a - T_b\) at a particular location \(x'\) is found by taking the difference in ordinate of the intersection points of the bus trajectories and the vertical plane.

PROBLEM FORMULATION

In this section we formulate the problem, define notation and propose an efficient method for storing and querying our bus data. Consider the task of monitoring service levels by finding mean bus headway and spacing at any location on the bus route over some time interval. We define a trajectory to consist of points \(p_1, p_2, \ldots, p_n\) where each point \(p\) is a 3-tuple consisting of a longitude \(x\), latitude \(y\), and a timestamp \(t\) as was observed by the automatic vehicle location system at location \((x, y)\) at time \(t\). Since each trajectory consists of a path in time-space, it is natural for us to represent...
the trajectory as a series of ordered line segments \( l_1, l_2, \ldots, l_{n-1} \) in a time-space cube. Each line segment \( l_i \) connects successive points \( p_i \) and \( p_{i+1} \) where \( p_i.t \leq p_{i+1}.t \forall i \in \{1, 2, \ldots, n-1\} \). For a large number of trajectories, iterating through every line segment in each trajectory to perform an intersection test is not practical. Since our goal is to query these trajectories quickly, a better approach is needed.

### Spatial Index

Our solution is to store our line segments in an R-tree, a data structure similar to a balanced binary search tree that is commonly used to index geometric objects such as points, lines and polygons (10, 11, 12, 13). The basic idea of the R-tree data structure is to group nearby trajectories and represent them with the minimum bounding rectangle that completely contains these trajectories in the next higher level of the tree. Queries on R-trees are fast because we can use these bounding rectangles to decide whether or not to search inside the subtree. In this way, most of the nodes in the tree are never read during a search.

For each line segment \( l_i \), we build an axis aligned bounding box (the smallest box within which contains all the points on the line) that has as its start and end vertices, points \( p_i \) and \( p_{i+1} \). We then add line segments to the R-tree one at a time so that the bounding box of each parent node in the R-tree completely contains the bounding boxes of its children. To keep the tree balanced, nodes are always added to the subtree that requires the least enlargement of its bounding box. This guarantees that we can search for line segments in \( O(\log(n)) \) time (14). It takes on average 250 ms to add a single bus line (both inbound and outbound for one day) to the spatial index. This is fast enough for our purposes since our app, described in Section 6, only loads new data from different days or bus lines when requested by the user.

Other methods to index line segments are possible e.g. by incoming and outgoing bus stops, but this is more computationally expensive because it requires us to keep a reference to each trajectory that enters and leaves every bus stop as well find the specific pair of bus stops that line segment “cuts”.

### Headways and Spacing

Recall that the time between successive buses gives you the headway and the distance, the spacing. Consider two successive trajectories \( a \) and \( b \). On a time-space diagram with \( x \) on the x-axis and \( t \) on the y-axis, the headway at \( x = x' \), \( h_{ab}(x') \), can be found by measuring the distance between the intersection of \( a \) and \( b \) with the vertical line at some fixed \( x' \). Similarly, the spacing at \( t = t' \), \( s_{ab}(t') \), can be found by measuring the distance between the intersection of \( a \) and \( b \) with the horizontal line at some fixed \( t' \).

Extending this analysis to the time-space cube, the intersection of these trajectories with the \( x - y \) plane at certain time \( t = t' \) tells us the location of the bus at \( t' \). If we superimpose this plane on a road segment, these intersections give us the position of the buses at \( t' \), which is analogous to taking an aerial photograph of that road segment. The horizontal distance between each intersection point gives us the separation between each bus. Similarly, the vertical distance between the intersections of these trajectories with the \( y - t \) plane at a fixed point \( x = x' \) perpendicular to the road gives us the headway between buses. This is analogous to having an observer standing on the side of the road at \( x' \) record the time between successive buses.

Based on the analysis above, we now have a geometric interpretation for an algorithm to calculate headways and spacing. The mean headway is simply the average vertical separation of
the intersection between a set of trajectories and a vertical rectangle while the mean spacing is then the average horizontal separation of the intersection between a set of trajectories and a horizontal rectangle (Figure 3). To find these intersections efficiently, we first use the spatial index to check which line segments have bounding boxes that intersect the rectangle. Next, we run a simple line and polygon intersection test to find the exact point of intersection, if any. The spatial index search is $O(\log(n))$ while the intersection test is $O(1)$, so the algorithm runs in $O(\log(n))$ time. The average execution time for the entire operation is 10 ms, which is fast enough to run in real time i.e. with negligible wait time or lag.

FIGURE 4 : Network Visualization and Analysis App Interface: Time-Space Cube View (A), Map Selection View (B), Visualization Options View (C)

NETWORK VISUALIZATION AND ANALYSIS APP
To explore the potential of using the time-space diagram with big data to better understand transportation network performance and behavior, we developed a visualization and analysis app (Figure 4) that automatically parses and represents bus data as trajectories in a time-space cube. This app, intended for use by transit operators and regulators, allows users to manage and visualize performance metrics such as headways and spacing effectively in an interactive and easily understandable way. One possible application of our software would be to diagnose bus bunching, which is when two or more buses that were scheduled to be evenly spaced on the same route, run together at the same time. This is characterized by irregular headways and spacing, and often results in longer waiting times for riders, overcrowding in some buses, and an overall decrease in service level and capacity (15).

Concern about bus bunching is far from academic. Transport regulators such as the LTA take it very seriously and have introduced incentives and penalties to encourage bus operators to improve reliability and reduce overcrowding. Under a new framework (16) announced earlier this year, bus operators in Singapore are required to reduce bus bunching by asking bus drivers to speed up or slow down or even add buses midstream.
Bus Bunching

Bus bunching arises naturally due to the stochastic nature of passenger arrivals at bus stops. The reason is that if on arriving at a bus stop a bus finds more than the average number of passengers waiting, this bus will likely be delayed (17). To make matters worse, the delayed bus starts picking up passengers that should have been picked up by the bus behind it so eventually the following bus catches up and both buses end up traveling as a single unit (18).

There has been significant research interest in designing control mechanisms to reduce bus bunching (19, 20, 21) but before such control systems can be implemented, the bus operator needs to be able to quantify the severity of the problem. This is typically done with a traditional time-space diagram by plotting the bus’s position along the route on the y-axis and time on the x-axis. But this approach abstracts the bunching problem away from geography. Our app, described in the next section, improves on this by allowing the user to easily identify and visualize bus bunching events temporally and spatially at any arbitrary point along a bus route and zoom in on problem areas.

Interface Design and User Interaction

We chose to prototype our app in java primarily because this allowed us to build on previous work (22) in visualizing large spatiotemporal datasets.

Two panels are displayed on the “Main Screen”. On the left is the “Time-Space Cube View” (Figure 4A) which shows bus trajectories from a single bus line, Bus 51, plotted in 3D space. Bus 51 was chosen because of its importance and popularity. It is a cross-island bus line that starts in a residential neighborhood in the west, enters the city center and ends in the eastern part of the country. The base of the cube is a city map of Singapore with the bus route highlighted in bright pink. Trajectories are colored according to how full each bus is. Green for empty and red for full. As in Section 2.1.2, the x-axis, y-axis and z-axis correspond to longitude, latitude and time.

On the right is the “Map Selection View” (Figure 4B) which allows users to drop markers on a zoomable map to define the plane that will be used in the intersection tests mentioned in 5.2. The “Visualization Options View” (Figure 4C) lets users choose via drop down menu the day, route number, route direction they are interested in (inbound or outbound), the type of statistics they want to view (headway or spacing) and the time period (00:00 hrs - 24:00 hrs).

Headways

To find the average headway at a specific point on the route for a specified time interval, the user drops markers (Figure 5B) on the map to draw a line segment across the road before selecting the start time and the duration of the time interval. This reinforces the notion that the user is making a “cut” through time across the road.

Pressing the run button generates a vertical “time-slice” and the app then uses the methods described in 5.2 to retrieve a list of intersecting trajectories and their intersection points. It then calculates both the average headway and standard deviation and displays a chart (Figure 7A) showing how buses (green for empty, red for full) passing by the line segment are separated in time. This is equivalent to recording the times at which busses pass by a stationary observer. Each bus is represented by a circle and overlapping circles indicate bus bunching. Simultaneously, the time-space cube view zooms in on the time-slice and highlights intersecting trajectories with a dull glow (Figure 5A), thereby allowing the user to visualize how trajectories intersect the vertical slice in time-space. As with the bus’s trajectory, the color of the glow is proportional to the number
of passengers on board.

2 Spacing

To find the average spacing of vehicles along a road segment, the user drops two markers (an X and O) at the start and end points (Figure 6B). This automatically generates a rectangle bounding box defining the horizontal time-slice that is used to find intersecting trajectories. The user then selects the start time and presses the run button to retrieve a list of intersecting trajectories and their intersection points. As before, the app calculates the average spacing and standard deviation and displays the results in a chart (Figure 7B). This is equivalent to taking an aerial photograph (represented by the light orange area in Figure 6B) of the road segment at a point in time. Again, each bus is represented by a circle and overlapping circles indicate bus bunching.

The user can adjust a time slider to move the horizontal time-slice up or down (forwards and backwards in time) to simulate the effect of watching a time-lapse video of buses moving along the road segment. As before, the time-space cube view zooms in on the time-slice and highlights intersecting trajectories with a dull glow (Figure 6A).
FIGURE 7: The app calculates the average headway (A), spacing (B) and their standard deviations and plots the bus’s relative positions in time (A) and space (B). Overlapping circles indicate bus bunching. The bus’s passenger load is encoded in the color of each circle.

DISCUSSION
We’ve shown how the classic time-space diagram remains an important tool in transportation system analysis by indexing real world bus trajectories as line segments in a time-space cube and using intersection tests between these trajectories and rectangle time-slices to efficiently calculate headways and spacing. Although this work was motivated by the problem of deriving performance metrics for a bus network, the methods presented in this paper are general; hence, they can be applied in contexts beyond public transit.

For example, since the realtime GPS positions of all aircraft in North America and Europe are logged and publicly available (23), minimum connection times for a pair of flights can be found by drawing a vertical time-slice at an airport runway and measuring the headway of the incoming and outgoing pair. In another very different example, the queue length of taxis waiting at a taxi stand at time $t'$ can be found by counting number of taxi trajectories that intersect a horizontal polygon time-slice bounding the taxi stand at $z$-position $t = t'$. 

CONCLUSION AND FUTURE WORK
In this paper, we demonstrated how simple geometric techniques inspired by the time-space diagram can be used to efficiently compute and visualize performance metrics for a bus fleet. We’ve developed an app that indexes observed bus trajectories as line segments in a time-space cube, and uses an intuitive user interface to allow users to analyze and visualize headways and spacing at any arbitrary point along a bus route.

The methods and algorithms described in this paper are general, and can be applied to any transportation system that collects data with a time-space component. We believe they can be used as the basis for a visualization and analysis tool that will enable transit operators and regulators to better understand network performance and behavior and monitor service levels, network capacity and delays.

The individual parts of this tool are well known in their respective fields: Time-space diagrams in transportation, time-space cubes in human geography, R-trees and intersection geometry
in computer graphics. This work combines them together in a novel way to analyze the real world problem of bus bunching.

Ultimately, we hope this tool can be used to help transit operators and regulators make better decisions when implementing control strategies or modifying bus frequency. To this end, future versions will allow users to highlight common bunching spots and see how different control strategies or scenarios (e.g. increasing bus frequency) affect service levels. We also plan to conduct a field trial in collaboration with a local transit agency to understand their needs and improve our app design and usability.

ACKNOWLEDGMENTS
Support for this research has been provided by the Future Urban Mobility project of the Singapore-MIT Alliance for Research and Technology (SMART), SMART Innovation Center Explorer Grant No. 015824-119 and Innovation Grant No. ING13057-ICT. We are grateful for this support. The authors would also like to thank Amedeo Odoni for his generosity, encouragement and advice.

REFERENCES


