Real-time learning and prediction of public transit bus arrival times

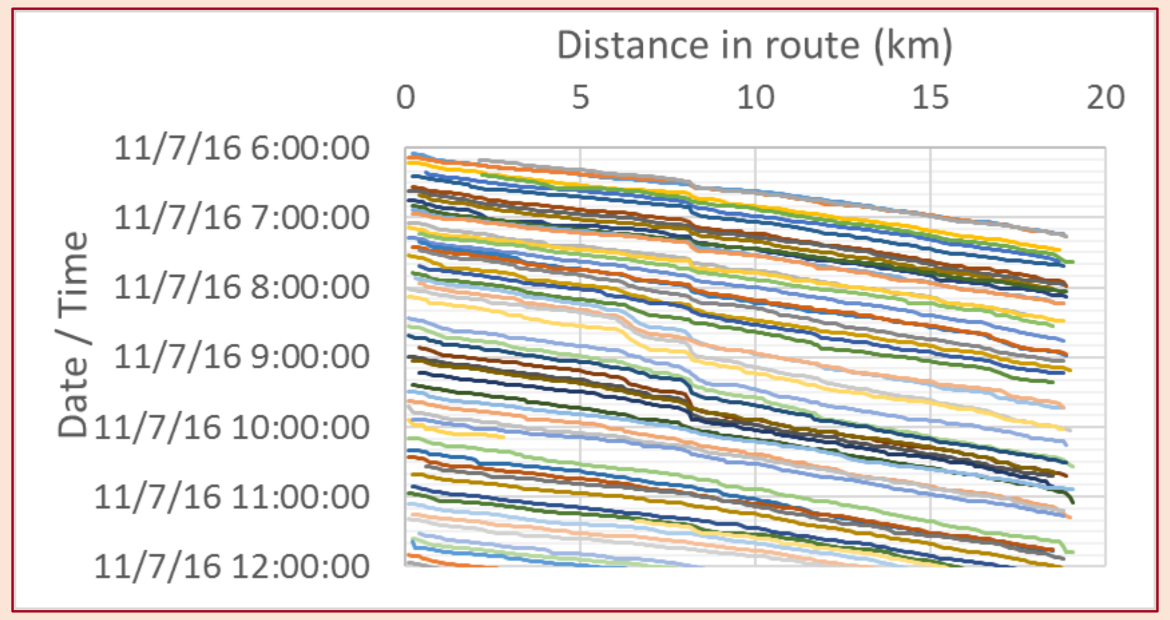
David Nissimoff (davidni@stanford.com) | Department of Electrical Engineering | Stanford University

Abstract

Millions of people depend on public transit to get to their destinations on time every day. This project leverages the real-time data available from the transit system in São Paulo, Brazil (API managed by SPTrans), and applies machine learning techniques to learn patterns and provide accurate predictions of bus arrival times in that city. The results obtained so far are very encouraging and demonstrate compelling prediction accuracy, especially considering the limitations of the API.

Overview

The figure below shows a typical morning of the "875C-10-0" service. Each line shows a vehicle as it performs the ~19 km trip. Notice the sheer number of transit vehicles and note the traffic jam at around the 8km mark, which clears at around 10:00am.



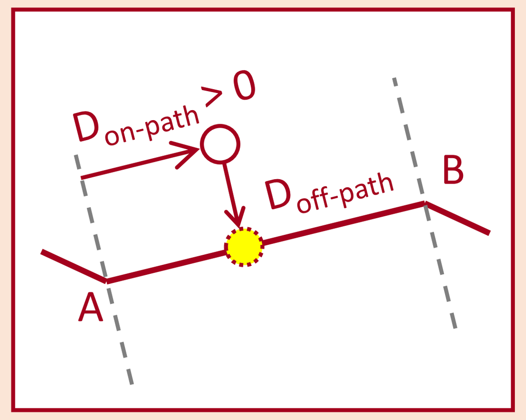
This chart was generated in Excel with data output from the present system. Transit vehicles perform the same routes multiple times each day and one vehicle is often just a few minutes behind another on the same transit line. Therefore, information from the vehicle ahead is indicative of conditions that will be encountered ahead. Also, traffic patterns have periodicity, so data from previous days / weeks are also useful.

We first break the trips down into segments and then use locally-weighted averaging[1] with appropriate transformations to the input data in order to model their periodicity. To predict the time a bus will arrive at a given place in its route, the last known position of the bus is simulated forward segment by segment and, as the simulation gets to new segments, we estimate the time to traverse that segment at the simulation timestamp. This enables us to deal with traffic conditions that change while a bus is *en route*.

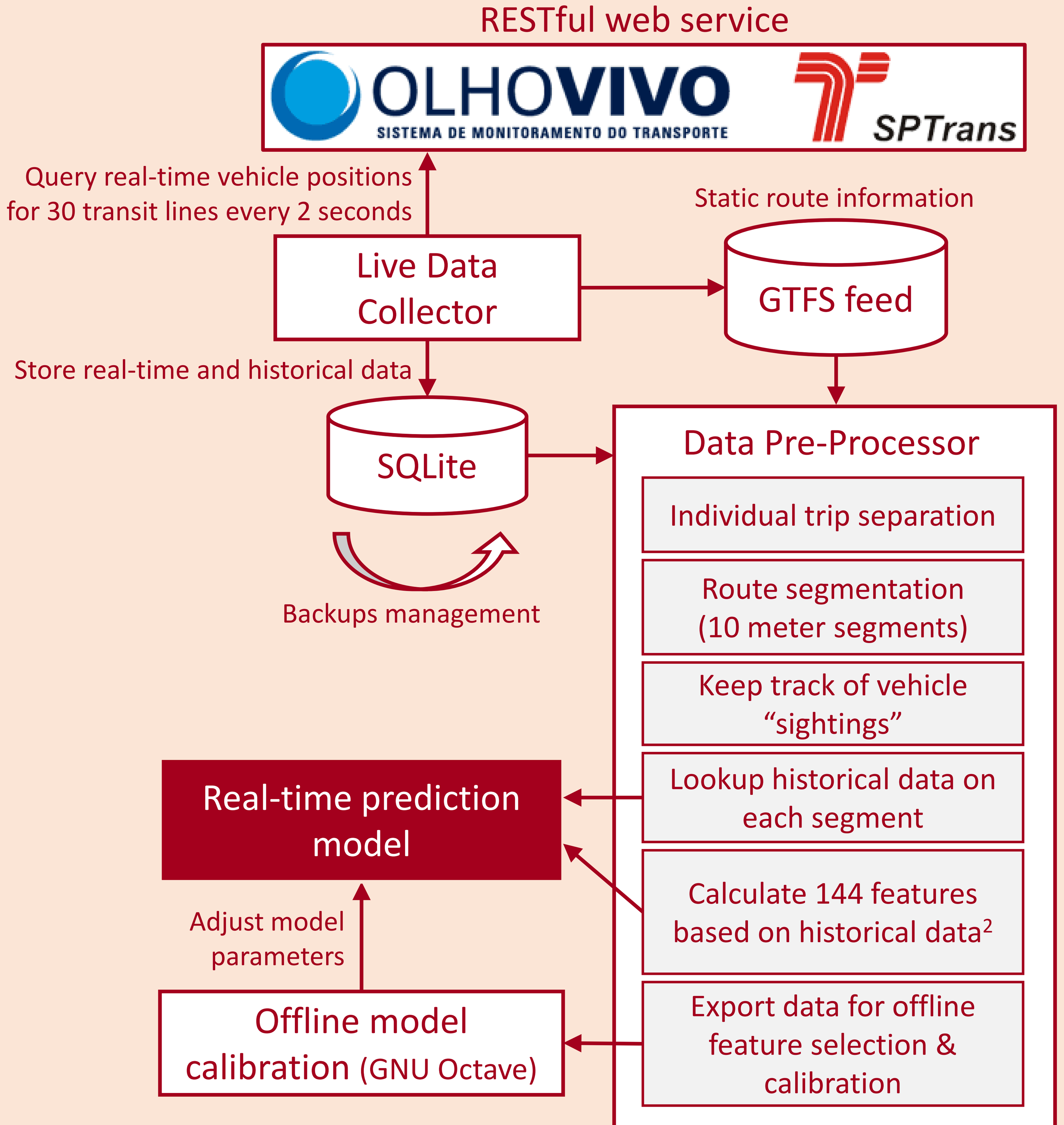
[1] due to its poor extrapolation ability, locally-weighted linear regression performed worse

Route segmentation

SPTrans provides a **GTFS feed** (General Transit Feed Specification) that gives the routes of all lines. We then project the real-time geo-coordinates from vehicles onto the known GTFS routes, obtaining a single number (meters along the route). The route is then split onto **equal-length segments** and a sample is registered each time a vehicle passes through a segment. Each sample includes the timestamp when the vehicle crossed the center of the segment and the travel time duration. Vehicle positions are linearly interpolated between "sightings" (a "sighting" refers to a new vehicle position returned from the API).



Data collection pipeline



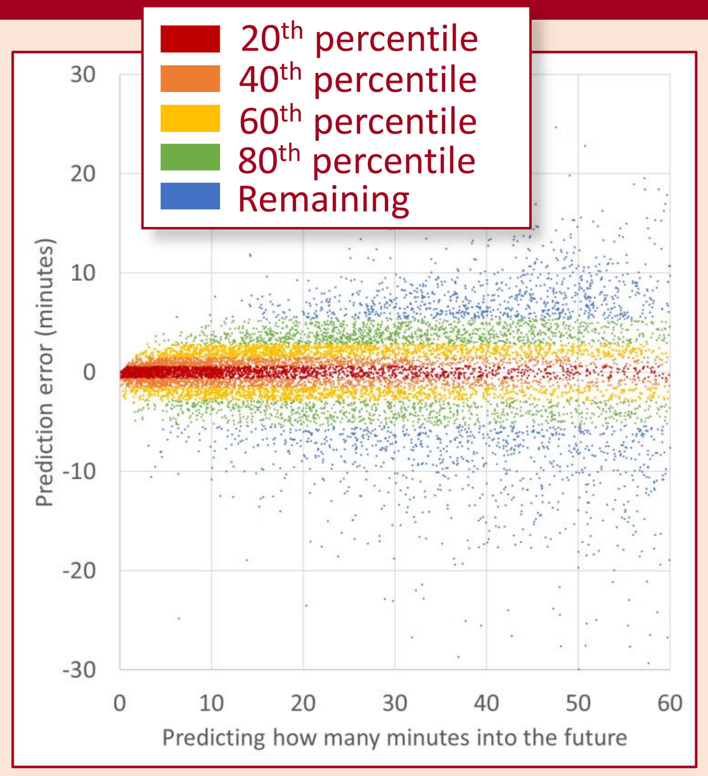
[2] see the box "Periodic locally-weighted averaging"

Periodic locally-weighted averaging

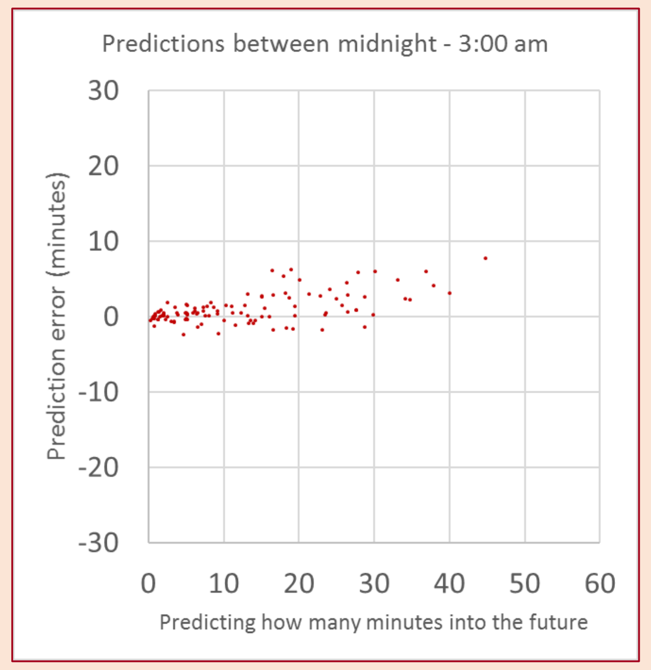
For each route segment individually, say we have samples $x^{(i)} \in \mathbb{R}$ and $y^{(i)} \in \mathbb{R}$ where $x^{(i)}$ is the timestamp when the i-th measurement for that segment was made (number of seconds from a reference date, for example) and $y^{(i)}$ is the measured time to traverse the segment. We first map $x^{(i)}$ onto a 3-D helix defining $v^{(i)} = \left(\sin\left(\frac{2\pi x^{(i)}}{T}\right), \cos\left(\frac{2\pi x^{(i)}}{T}\right), \frac{p x^{(i)}}{T}\right) \in \mathbb{R}^3$ where T is the model periodicity, and p is the helix pitch. To obtain an estimate at a query time t, we map t to $q \in \mathbb{R}^3$ with the helix transform and take the average of $y^{(i)}$'s using weights $w_i = \exp\left(\frac{-\sum_{j=1}^{3}\left(q_j - v_j^{(i)}\right)^2}{\tau^2}\right)$. Of course, the weights must be normalized. By varying T, τ and p we obtain different estimates for a segment. Call these $\hat{y}_{T,\tau,p}$ for each value of the parameters. For each query time t, we compute $\hat{y}_{T,\tau,p}$ with 9 values of τ, 8 values of p and 2 values of T, totaling 144 values. We do all of this as each segment is crossed by a vehicle, and we generate data for a second learning algorithm. Now we have features $c^{(i)} \in \mathbb{R}^{144}$ mapping to $y^{(i)}$ and linear regression was found to perform adequately. The best 3 features (measured by best test error with a train:test ratio of 70:30) were selected by brute-force, so only the 3 corresponding values of $\hat{y}_{T,\tau,p}$ are needed for predictions.

Results and future work

More than **13 million vehicle locations** have been collected so far (in just over 7 weeks). The chart to the right helps visualize the accuracy of predictions for the "875C-10-0" line with the proposed method. The horizontal axis represents the time horizon of the prediction, whereas the vertical axis shows the prediction error. Predictions were made between all "sightings" of each vehicle as it went on its way.



As the figure below illustrates, accuracy is better outside of peak hours, since traffic is a lot more predictable at those times (no traffic jams).



The main goals for the project have been achieved and it was demonstrated that it is indeed possible to predict bus arrival times based on past data.

Being able to predict where a bus will be up to one hour into the future is very powerful and opens up many interesting scenarios. Ideas for future work include migrating the implementation to a distributed cloud environment and building a user interface on top of this project to show useful and actionable data to a transit rider, as well as further improving the prediction models.