**Proceedings of the 11th International IEEE
Conference on Intelligent Transportation Systems
Beijing, China, October 12-15, 2008**

Obtaining Dense Road Speed Estimates from Sparse GPS Measurements

Andrew Phan and Frank Ferrie

***Abstract*— A major challenge for traffic management systems is the inference of traffic flow in regions of the network for which there is little data. In this paper, GPS-based vehicle locator data from a fleet of 40-60 roving ambulances are used to estimate traffic congestion along a network of 20,000 streets in the city of Ottawa, Canada. Essentially, the road network is represented as a directed graph and a belief propagation algorithm is used to interpolate measurements from the fleet. The system incorporates a number of novel features. It makes no distinctions between freeways and surface streets, incorporates both historical and live sensor data, handles user inputs such as road closures and manual speed overrides, and is computationally efficient - providing updates every 5 to 6 minutes on commodity hardware. Experimental results are presented which address the key issue of validating the performance and reliability of the system.**

I. Introduction

Maintaining the state of a complex road network given limited sensor input is a key challenge to the design of modern traffic control systems. In this paper we address the specific problem of inferring traffic density along each of 20,000 streets in the greater city of Ottawa, Canada, from GPS-based vehicle locator data supplied by a fleet of 40-60 roving vehicles. The context for this work is a system for managing the redeployment of a fleet of ambulances operated by the Ottawa Paramedic Service. The goal of the project, RISER[1], is to develop a system capable of dynamically positioning the fleet so as to guarantee response time while respecting operating constraints such as load balancing, crew scheduling, and minimizing movement. As can be seen in Fig. 1, the system relies on a traffic server module to provide an estimate of the traffic congestion along each of the approximately 20,000 streets in the city road network. These data are used to estimate time of arrival (ETA) for any path through the network, which in turn is used for determining optimal redeployment of the fleet by the scheduling engine.

The focus of this paper is the traffic server, which uses a combination of sparse measurements, historical traffic data, and a GIS database to determine the likely traffic flow state at any location in the network. The key idea of the paper is that given prior information in the form of date/time indexed historical data, one can reliably interpolate a set of sparse, real-time measurements, and obtain reasonable estimates over the extent of the network. This is achieved by means of a belief propagation algorithm which operates on a representation of the traffic network as a directed graph and serves to integrate information from different sources. Here we consider only vehicle locator data, but the algorithm is sufficiently general to incorporate any available source of information provided that i) it is geo-referenced to a particular location in the network, and ii) that it can be expressed in terms of relative congestion (described shortly). The particular belief propagation algorithm we use is Relaxation Labeling (RL), which has been widely used in the computer vision field [1], [2], [3].

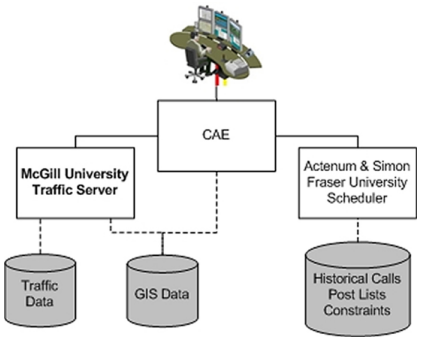

Fig. 1. Proposed Ambulance Dispatch System

The organization of the paper is as follows. Section II formally defines the problem. Section III reviews the current state of the art. Section IV describes the solution we propose. Section V details the experimental methodology and the validation results. The paper ends with a discussion in Section VI and a suggestion for future work in Section VII.

II. Problem Definition

In this section, we formally define the problems that are addressed by the traffic server. The first problem is that of interpolating sparse data since the city of Ottawa has approximately 20,000 streets while the number of available sensors is orders of magnitude lower. Second is the problem of data fusion since there exist many different types of sensors such as traffic cameras, inductive loops, radars and GPS units that each provide information about the state of the road network. Third is the question of how to benefit from the historical data accumulated over time and how to combine it with the live sensor data. Fourth, the user should be able to modify the state of the road network by closing a road or manually inputting a road speed without rendering the model useless. An example of a road closure would be

The authors are with the Center for Intelligent Machines, Department of Electrical and Computer Engineering, McGill University, Canada aphan2@cim.mcgill.ca and ferrie@cim.mcgill.ca

[1]Rapid Intelligent Scheduling for Emergency Responders, a joint project led by CAE Inc. under the Precarn Inc. CORE program, with the participation of Actenum Inc., McGill University, Simon Fraser University, and the Ottawa Paramedic Service.

during a special event such as a festival or parade while a speed override would be used in the case of an accident. Fifth, the model should scale well to handle an entire city including both freeways and surface streets. Sixth, the model must function in real-time for the ambulance dispatchers. Finally, validating the model is challenging due the size of the road network and the quantity of the data.

III. Related Work

In the literature, traffic models are generally classified as either microscopic or macroscopic. A microscopic model tracks the movement of individual vehicles and generally includes a model for the driver, the vehicle and how the vehicle interacts with its surroundings. A macroscopic model, on the other hand, tends to treat congestion as a fluid and represents the dynamics of traffic flow as a group rather than as individual vehicles. We describe below several systems that we believe are representative of the current state of the art in traffic modelling.

FreeSim [4], [5] is a framework that offers both macroscopic and microscopic traffic modelling of freeways. It requires that a large number of vehicles in the road network have a GPS device that transmits their current speed and location to a central server. The central server then uses this information to update the travel speed of every road segment in the road network database. At any time during the simulation, a user may request the quickest route to get from one location to another and the system responds by using one of the six routing algorithms implemented to compute quickest travel times. Only live data are used or necessary because it is assumed that a large enough number of vehicles are transmitting and updating the central server regularly. FreeSim has been tested on Los Angeles' freeways using user-generated data and live data from the California Department of Transportation. Their main limitations are that they do not interpolate sparse sensor data, do not incorporate historical data and do not consider surface streets.

Corsim (2007) [6], [7] is a microscopic traffic simulation sponsored by the United States Federal Highway Administration and consists primarily of two traffic models: Netsim for modelling surface streets and Fresim for modelling freeways. It has undergone extensive validation, can model control devices (e.g traffic signals, ramp metering etc.), vehicles and driver behaviour, and includes the ability to accommodate complicated road geometries. However, as is generally the case with microscopic models, there is a limit on both the maximum number of sensors and the maximum size of the road network. Furthermore, it is unclear from the literature if they handle the interpolation problem or if they use any historical data.

A variant of the classical cellular automaton model [8] (2001) is presented in [9] for modelling freeways. It is a microscopic traffic model that divides the road into cells such that each cell may contain no more than one vehicle. Vehicles are modelled according to rules and lane changes are a function of the speed of the vehicle ahead and on the number of empty cells ahead and to the side. To validate their system, they tested it using three years of inductive loop data on the North Rhine-Westphalia freeway network for which they simulated 14,000,000 cells on 2,500 kilometres of freeway. The system uses both historical and live inductive loop counts. When the historical data indicates insufficient vehicle counts on a given segment, it fills up the available (empty) cells in the vicinity of the inductive loop with vehicles until the number of counts matches the number of counts in the historical data or until there are no available cells left. Their model can extrapolate estimates into areas with little or no sensors and considers 4 classes of traffic patterns: Monday-Thursday (except holidays and days before holidays), Friday and days before holidays, Saturday except holidays, and Sunday and holidays. Their main drawback is their microscopic model that limits their scope to only freeways.

In [10], the authors present Adaptive Routing (2000) that integrates historical and live congestion information into a routing system for both surface streets and freeways. Each road segment has an estimated travel speed depending on the time of day. Each day is discretized into 15 minute intervals such that estimated travel speed for a given time interval is a geometrically weighted average of the historical and live speed data. Given the large number of road segments compared to the low number of sensors, when no new data are available the system generates synthetic speed data to prevent the average speed from decreasing monotonically. They explore many different routing algorithms and have tested their system using the Trafficmaster [11] service which provides road speed data of the United Kingdom. The system we propose takes their system to the next level by interpolating the sparse sensor data rather than generating synthetic data.

There are other systems not covered in the preceding literature review but worth mentioning. If interested, the reader is invited to refer to the following list of related works: Renaissance (2006) [12], Type-2 Fuzzy Logic Approach [13], Dicaf [14], Vatsim [15], Mitsim [16], Watsim [17], Transims [18], Paramics [19] and Vissim [20].

IV. Proposed Solution

Unlike the previous traffic systems, we do not implement any routing algorithm. The reason being simply that it was not a requirement of the traffic server module. Also, in the literature, microscopic models are mostly used because they are better able to handle complicated road geometries and features such as traffic lights or on-ramp metering [6]. However, a macroscopic traffic model has the advantage when it comes to large scale designs thanks to the reduced number of necessary variables [21]. Because we hope to estimate congestions for an entire city, we use a macroscopic model to keep the large scale implementation as simple as possible. Indeed, we propose a system that can handle the entire city of Ottawa including both freeways and surface streets, is not limited by the number of sensors and is updated every 5 to 6 minutes which is acceptable for the needs of the ambulance dispatchers.

A. Module 1: Computing Sparse Local Estimates

The first module has the task of determining the most likely travel speed for each road segment based on the available sensor and user override data. To accomplish this task, we construct, for each road segment, a speed profile that is essentially a trend in ambulance speeds over time.

The sensor data that we use is automatic vehicle locator (AVL) GPS data, available for many of Ottawa's ambulances. A GPS device is installed on the ambulance and transmits its position in regular intervals to a central server that records the data. Data from February 1, 2007 to December 20, 2007 were used for the experiments presented in this paper. The dataset comprises approximately 18 million records, with each record containing vehicle ID, date, time and GPS latitude and longitude. Since the AVL data only gives the positions of ambulances every 10 to 12 seconds on average, we use 3 consecutive positions in time to reliably estimate the ambulance's direction and register it to a specific road segment. Explicitly, if we know where the ambulance was at time t_{t-1} and where it will be at time t_{t+1}, then we can determine on which road segment the ambulance is travelling at time t_t. We estimate the ambulance travel speed using a straight line approximation by computing the distance travelled and dividing this value by the time interval. For time intervals on the order of 10 to 12 seconds, the straight line approximation provides a satisfactory baseline though there is room for improvement by using a more sophisticated road following algorithm. Any invalid speeds due to GPS tracking error are filtered in this process such that, from the original 18 million raw AVL samples, we obtain approximately 13 million useful ambulance speed samples that we refer to as the *processed* AVL data.

Currently, we split the processed AVL data into two classes, weekdays and weekends, so that the model can be trained to give appropriate speed estimates according to the day of the week and the time of day. In the future more sophisticated data clustering could be used to improve the estimates (e.g. splitting the data according to weather conditions and holidays).

Next we construct two speed profiles (one for each data class). Two different methods for constructing the trend in ambulance speeds were investigated. The first uses linear regression to estimate the coefficients of a polynomial fit to the sample data,

$$\text{Speed}(t) = w_0 t^0 + w_1 t^1 + w_2 t^2 + ... + w_p t^p. \qquad (1)$$

The second method, bin averaging, does not attempt to fit a curve to the data. Rather, we divide a day into 15 minute bins and compute the mean for each bin. Fig. 2 shows the weekday speed profiles of 4 different road segments with abundant AVL samples using both methods.

Then we address the problem of merging different sensors by converting arbitrary sensor measurements to a relative congestion term $\in [0, 1]$ using an equation that may be unique for each sensor. For the AVL data, we use

$$\text{Relative Congestion}(t) = 1 - \frac{\text{Speed}(t)}{\text{Speed Limit}}, \qquad (2)$$

but for other sensors such as inductive loops the Relative Congestion(t) might be obtained directly from counts rather than from vehicle speeds.

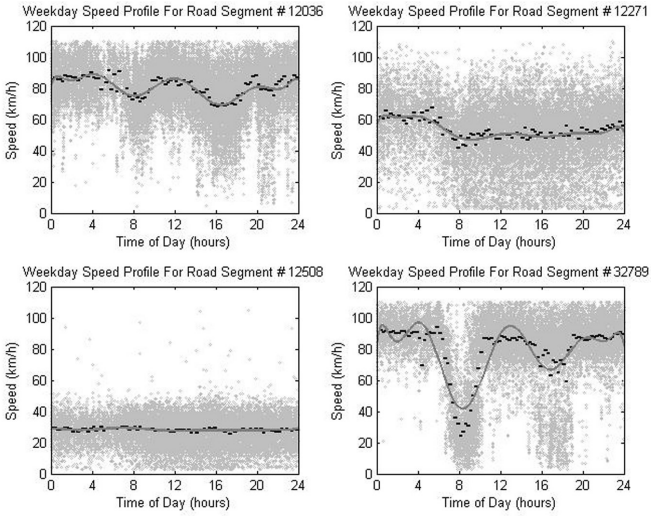

Fig. 2. Weekday Speed Profiles for 4 Different Road Segments. Light grey points show the processed AVL speed samples. The black dashed lines is the fit due to bin averaging while the dark grey continuous line is the polynomial fit.

Unfortunately, the relative congestions that serve as local estimates are sparse and noisy. Since ambulances drive over some roads more often than others, some road segments may have tens of thousands of AVL samples while others have none. Furthermore, as can be seen in Fig. 2, there is a great deal of variance in the AVL data. Both sparse and noisy data contribute to the likelihood that relative congestions of neighbouring road segments contradict each other.

B. Module 2: Obtaining Dense Global Estimates

It is up to the second module to resolve local inconsistencies and propagate the sparse local estimates to neighbouring road segments such that we obtain dense global estimates that are locally consistent. To accomplish this interpolation task, we implement a Relaxation Labelling algorithm (RL) [22] that uses the road network topology (number of lanes, speeds limits and linkages) to determine how likely each road segment is congested based on how likely its neighbouring road segments are congested.

We chose the RL algorithm for interpolation because it is relatively simple to implement and offers a baseline to which we can compare eventual more complex strategies. Furthermore, it allows us to split a complex computation into many simple and parallel computations, uses context to compensate for noise and ambiguity, and is deterministic.

We begin by representing the road network as a directed graph structure. Each node in the graph corresponds to a single road segment with a unique direction and nodes are

linked to each other via edges. As such, a two-way street gives rise to two road segments whose directions are 180° apart while a one-way street corresponds to a single road segment. Each node i has a set of m labels where each label corresponds to a range of relative congestion. For example, if m is 5, then labels $\{\lambda^1, \lambda^2, \lambda^3, \lambda^4, \lambda^5\}$ correspond to relative congestions $\{[0\text{-}0.2], (0.2\text{-}0.4], (0.4\text{-}0.6], (0.6\text{-}0.8], (0.8\text{-}1.0]\}$ respectively. In turn, each label has a weight, $p_i(\lambda^x)$, that defines the likelihood of the label. Label weights are bound between 0 and 1 and a node's label weights must add up to 1 i.e. $\sum_{x=1}^{m} p_i(\lambda^x) = 1$. Detailed next are the 8 main steps of the RL algorithm:

RL 1: Initialize the iteration counter $k = 0$ and the label weights $\bar{p}^{\,k}$ according to the current date and time, t_{curr}. If a given road segment at t_{curr} has only historical data, then we initialize the label weights using a normal distribution $N(\mu, \sigma^2)$, where the mean μ is the relative congestion (computed in Module 1) at t_{curr} and the variance σ^2 of the distribution is a function of both the variance and the number of historical AVL samples. If there are many samples and the AVL variance is low, then the label weights will have a more peaked distribution. On the other hand, if there are few samples or the AVL variance is high, then the label weights will have a flatter distribution. In the extreme case that the road segment has no historical data, then we initialize the label weights using a uniform distribution. If live sensor data are available or a user has manually entered a speed override, then we initialize the label weights using a highly peaked distribution.

RL 2: Compute the support of each label, $\bar{q}^{\,k}$. The support for a given label of a given node is dependent on the label weights of neighbouring nodes and on the road network topology. If each node has m labels such that node i with label λ has neighbour j with labels λ', then the support for λ can be computed as follows:

$$q_i(\lambda) = \sum_{\lambda'=1}^{m} r_{ij}(\lambda, \lambda') p_j(\lambda'), \tag{3}$$

where $r_{ij}(\lambda, \lambda')$ represents the compatibility between label λ at node i and label λ' at node j. If the two labels greatly support each other, then $r_{ij}(\lambda, \lambda')$ should be large and positive. If the labels greatly inhibit each other, then $r_{ij}(\lambda, \lambda')$ should be large and negative. Otherwise, if there is no relation between the two labels, then $r_{ij}(\lambda, \lambda')$ should be 0. In practice, node i can have multiple neighbours $j^1, j^2, \ldots, j^n$ with labels $\lambda^1, \lambda^2, \ldots, \lambda^n$, respectively, where n is typically 1-6. Then the equation for support becomes:

$$\begin{aligned} q_i(\lambda) = \sum_{\lambda^1=1}^{m} \sum_{\lambda^2=1}^{m} \cdots \sum_{\lambda^n=1}^{m} & r_{ij^1 j^2 \cdots j^n}(\lambda, \lambda^1, \lambda^2, \cdots, \lambda^n) \\ & \cdot p_{j^1}(\lambda^1) p_{j^2}(\lambda^2) \cdots p_{j^n}(\lambda^n). \end{aligned} \tag{4}$$

Currently, the label compatibilities are static and do not change according to the date or time. Only when the road network topology is modified as in the case of a road or lane closure is it necessary to re-compute them. The label compatibilities are dependent on the road network topology and on various traffic flow assumptions at intersections. As a first-order approximation, the label compatibilities being static provide a satisfactory baseline though eventually they should be dynamic and also be dependent on intersection data such as turn ratios.

RL 3: Compute the update direction of each label, $\bar{u}^{\,k}$. The algorithm to obtain $\bar{u}$ is described in [22] Appendix A, was used exactly as is and, as a result, is not detailed here due to lack of space. In essence, it is shown in [22] that $\bar{u}$ is the projection of the support vector $\bar{q}$ onto $T_{\bar{p}}$, the set of all tangent vectors at $\bar{p}$.

RL 4: If $\bar{u}^{\,k} = \bar{0}$ such that the projection computed previously is zero then go to RL 8.

RL 5: Otherwise, update the label weights $\bar{p}^{\,k+1} = \bar{p}^{\,k} + h\bar{u}^{\,k}$, where $0 < h \leq \alpha_k$ and α_k is a small valued maximum step size that may decrease as k increases to speed convergence. Each iteration moves the label weights $\bar{p}$ incrementally in the projection direction.

RL 6: If $k = k_{\max}$ such that the iteration counter has reached the maximum iteration count then go to RL 8.

RL 7: Otherwise, increment the iteration counter $k = k+1$ and go back to RL 2.

RL 8: Obtain the final labelling by assigning the label with the largest weight to each node and go back to RL 1 to resume the entire process for the new current date and time, $t_{\text{curr}'}$. When each road segment is labelled with a most likely relative congestion interval, we convert the centre of the interval to a potential ambulance speed using

$$\text{Speed} = \text{Speed Limit} \cdot (1 - \text{Relative Congestion}). \tag{5}$$

The entire RL algorithm starts over again in order to keep the road network up-to-date with the latest speeds.

V. Experimental Methodology and Results

To validate our system, we employ a common technique known as K-fold cross-validation. First, we split the historical data randomly into k evenly sized parts called folds. Then we choose one of the folds to act as the test set and train the model on the remaining k-1 folds. We repeat this process k times using each fold only once as the test set and always training on the remaining folds. Due to the time it takes to update the model and the large number of data samples in the test set, we ran the traffic server and recorded the results over a continuous span of 24 hours for each data class k times. If $k = 5$, then it takes 10 days $(= 5 \text{ folds} \times 2 \text{ data classes})$ to validate the system regardless of the size of the road network or the amount of historical data.

Using 5 folds, 9 RL iterations and 7 RL labels, we obtain the results shown in Table I and II. Because each label in the RL algorithm corresponds to a range of speeds, we compute the model error in both kilometres per hour and labels.

In the hopes of reducing our error, we tried relaxing the update time interval constraint from 5-6 minutes to 10-11 minutes in order to increase the number of RL labels from 7 to 9 thus reducing the coarseness of our estimates. Using 5 folds, 9 RL iterations and 9 RL labels, we obtain the results shown in Table III and IV.

TABLE I

K-FOLD CROSS-VALIDATION SIMULATION RESULTS FOR WEEKDAYS (TOP) AND WEEKENDS (BOTTOM) USING POLYNOMIAL REGRESSION, 9 RL ITERATIONS, 7 RL LABELS. MAE: MEAN AVERAGE ERROR; RMSE: ROOT MEAN SQUARED ERROR; STD: STANDARD DEVIATION.

Fold k	1	2	3	4	5	Mean
MAE (km/h)	12.75	12.75	12.76	12.75	12.78	12.76
RMSE (km/h)	17.06	17.05	17.07	17.05	17.09	17.06
STD (km/h)	11.34	11.33	11.34	11.32	11.34	11.33
MAE (labels)	1.26	1.26	1.27	1.27	1.27	1.27
RMSE (labels)	1.82	1.82	1.82	1.82	1.82	1.82
STD (labels)	1.31	1.31	1.31	1.31	1.31	1.31

Fold k	1	2	3	4	5	Mean
MAE (km/h)	13.12	13.13	13.11	13.12	13.14	13.12
RMSE (km/h)	17.96	17.97	17.97	17.96	17.98	17.97
STD (km/h)	12.26	12.28	12.28	12.26	12.27	12.27
MAE (labels)	1.29	1.29	1.29	1.29	1.29	1.29
RMSE (labels)	1.93	1.93	1.93	1.93	1.93	1.93
STD (labels)	1.43	1.43	1.43	1.43	1.43	1.43

TABLE II

K-FOLD CROSS-VALIDATION SIMULATION RESULTS FOR WEEKDAYS (TOP) AND WEEKENDS (BOTTOM) USING BIN AVERAGING, 9 RL ITERATIONS, 7 RL LABELS. MAE: MEAN AVERAGE ERROR; RMSE: ROOT MEAN SQUARED ERROR; STD: STANDARD DEVIATION.

Fold k	1	2	3	4	5	Mean
MAE (km/h)	12.18	12.17	12.18	12.18	12.19	12.18
RMSE (km/h)	16.25	16.24	16.26	16.24	16.26	16.25
STD (km/h)	10.76	10.75	10.76	10.75	10.76	10.76
MAE (labels)	1.22	1.22	1.22	1.22	1.22	1.22
RMSE (labels)	1.76	1.76	1.76	1.76	1.76	1.76
STD (labels)	1.27	1.27	1.27	1.27	1.27	1.27

Fold k	1	2	3	4	5	Mean
MAE (km/h)	12.24	12.25	12.22	12.23	12.27	12.24
RMSE (km/h)	16.69	16.69	16.66	16.67	16.71	16.68
STD (km/h)	11.35	11.35	11.32	11.33	11.35	11.34
MAE (labels)	1.23	1.23	1.22	1.22	1.23	1.23
RMSE (labels)	1.84	1.83	1.83	1.83	1.84	1.83
STD (labels)	1.37	1.37	1.36	1.36	1.37	1.37

TABLE III

K-FOLD CROSS-VALIDATION SIMULATION RESULTS FOR WEEKDAYS (TOP) AND WEEKENDS (BOTTOM) USING POLYNOMIAL REGRESSION, 9 RL ITERATIONS, 9 RL LABELS. MAE: MEAN AVERAGE ERROR; RMSE: ROOT MEAN SQUARED ERROR; STD: STANDARD DEVIATION.

Fold k	1	2	3	4	5	Mean
MAE (km/h)	12.63	12.62	12.64	12.62	12.63	12.63
RMSE (km/h)	17.00	16.99	17.01	16.99	17.00	17.00
STD (km/h)	11.37	11.37	11.39	11.37	11.38	11.38
MAE (labels)	1.58	1.58	1.59	1.58	1.59	1.58
RMSE (labels)	2.30	2.30	2.30	2.30	2.30	2.30
STD (labels)	1.66	1.66	1.67	1.66	1.66	1.66

Fold k	1	2	3	4	5	Mean
MAE (km/h)	13.03	13.04	13.02	13.02	13.05	13.03
RMSE (km/h)	17.92	17.94	17.91	17.90	17.94	17.92
STD (km/h)	12.30	12.31	12.30	12.29	12.31	12.30
MAE (labels)	1.63	1.63	1.63	1.62	1.63	1.63
RMSE (labels)	2.44	2.44	2.44	2.44	2.44	2.44
STD (labels)	1.82	1.82	1.82	1.82	1.82	1.82

TABLE IV

K-FOLD CROSS-VALIDATION SIMULATION RESULTS FOR WEEKDAYS (TOP) AND WEEKENDS (BOTTOM) USING BIN AVERAGING, 9 RL ITERATIONS, 9 RL LABELS. MAE: MEAN AVERAGE ERROR; RMSE: ROOT MEAN SQUARED ERROR; STD: STANDARD DEVIATION.

Fold k	1	2	3	4	5	Mean
MAE (km/h)	12.21	12.21	12.09	12.09	12.09	12.14
RMSE (km/h)	16.41	16.40	16.28	16.27	16.27	16.33
STD (km/h)	10.97	10.95	10.90	10.89	10.89	10.92
MAE (labels)	1.54	1.54	1.53	1.53	1.53	1.53
RMSE (labels)	2.24	2.24	2.22	2.22	2.22	2.23
STD (labels)	1.63	1.62	1.61	1.61	1.61	1.62

Fold k	1	2	3	4	5	Mean
MAE (km/h)	12.40	12.40	12.29	12.30	12.32	12.34
RMSE (km/h)	16.99	17.01	16.88	16.88	16.90	16.93
STD (km/h)	11.61	11.63	11.57	11.56	11.57	11.59
MAE (labels)	1.56	1.56	1.54	1.54	1.55	1.55
RMSE (labels)	2.34	2.35	2.32	2.33	2.33	2.33
STD (labels)	1.75	1.75	1.74	1.74	1.74	1.74

VI. DISCUSSION

A. Validation

The mean absolute errors (MAEs) we obtained of 12-13 km/h may seem large but, at the same time, they are unsurprising for several reasons. First, we observed that the AVL data has a great deal of variance, as can be seen in Fig. 2. Second, our road network database of Ottawa has errors in terms of road linkages (certain streets are supposed to connect but do not), lacks critical information such as labelling of one-way streets, and has incomplete information such that certain streets have no speed limits while others have ambiguous or no number of lanes information. Our partners are actively working to get a more complete and correct road network.

For computing local estimates, we notice that bin averaging consistently leads to better results compared to polynomial regression. The reason is not evident for roads with a great number of AVL samples as is the case in Fig. 2 but, for roads with too few AVL samples, polynomial regression results in a poor fit. On the other hand, increasing the number of RL labels did not produce a significant improvement in the MAE or the root mean squared error (RSME). One possible explanation is the large variance in the AVL data. Overall, however, given the sparse nature of the measurements and the coarse quantization of speed ranges, the results obtained were much better than expected - more than sufficient to estimate travel times between any two nodes in the network. Perhaps more importantly, they can serve as a baseline from which to assess enhancements to the method and new data sources.

B. Data Sources

The original goal was to use image sequences from the existing traffic camera network, but this proved unfeasible for many reasons. Most notably is the lack of a uniform infrastructure for routing data to a central server. Also, lack of access to camera parameters such as pose and zoom makes the design of robust estimation procedures problematic.

The inductive loop count data was more promising because

we had access to almost 200 loops spanning 5 years. Unfortunately, we had an issue with the registration of the sensor data for although we were given the nearest intersection and the road segment direction, we did not know in which lane or where along the road segment the loops were located. Because different lanes do not necessarily receive the same number of vehicle counts despite having approximately the same travel speed, there is a high probability that different loops of the same road segment contradict one another, especially if one of the loops is located in a turning lane. Another issue is the lack of ground truth that we could have used to convert vehicle counts into travel speeds. Without the exact loop positions and the ground truth, we have no way of reliably converting loop counts into average travel speeds unless we implement a microscopic traffic model which, as we explained in Section IV, is an approach we want to avoid if possible.

In the end, we used readily available AVL data which turned out to be a more practical solution. Unlike the inductive loops which are statically located and potentially contradictory, we do not need to know the exact location of the ambulance on the road segment because we assume that the ambulance is travelling on whatever lane is available and that the average speed of the ambulance for a given lane is the same for each lane. This technology can also be expanded to other fleets such as municipal vehicles or taxis to produce more accurate results. Unfortunately, due to privacy concerns, we cannot make the data we used available to the research community for standardized testing.

VII. Future Work

In this section, we propose several possible improvements. The first is to consider other data classes such as weather conditions, holidays and special events or to employ more sophisticated data clustering techniques. However, doing so would only be useful if there is sufficient historical data from which to construct reliable speed profiles. The second is to improve the straight line approximation used to estimate ambulance speeds and to distinguish speeds where the ambulance has its lights or sirens on compared to when they are off. As we are only interested in the redeployment problem, we are not interested in the ambulance data with lights or sirens on because ambulances generally drive with their lights and sirens off during redeployment. Third is to study the relationship between the sparseness of the data and the accuracy of our estimates in order to answer the question: how sparse is too sparse? Fourth is the need to accept other data sources such as traffic cameras or inductive loops since each provide useful information about the state of the road network. Finally, label compatibilities should be dynamic rather than static and be computed from intersection data such as turn ratios.

VIII. Conclusion

To summarize, we presented a system that applies to both freeways and surface streets, uses both historical and live APL data, handles user modifications such as road closures and speed overrides, outputs an estimate every 5-6 minutes, and interpolates the sparse sensor data by using the Relaxation Labelling algorithm. Furthermore, we validated the system using K-folds cross-validation and obtained a mean absolute error of 12-13 km/h (1-2 labels), a root mean squared error of 17-18 km/h (2-3 labels) and a standard deviation of 11-12 km/h (1-2 labels). Currently, the traffic server is being prototyped in Ottawa, Canada. CAE Presagis aims to have it serve other departments such as Police or Fire as well as other cities across the world. Such a system is useful anywhere there are dispatchers that must effectively manage a limited set of resources.

References

[1] A. Rosenfeld, R.A. Hummel, and S.W. Zucker. Scene labeling by relaxation operations. *IEEE Transactions on Systems, Man and Cybernetics*, 6(6):420–433, 1976.

[2] W.J. Christmas, J. Kittler, and M. Petrou. Structural Matching in Computer Vision Using Probabilistic Relaxation. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, page 749764, 1995.

[3] P.W.H. Kwan, K. Kameyama, and K. Toraichi. On a relaxation-labeling algorithm for real-time contour-based image similarity retrieval. *Image and Vision Computing*, 21(3):285–294, 2003.

[4] J. Miller and E. Horowitz. Freesim - a free real-time freeway traffic simulator. *IEEE Intelligent Transportation Systems Conference*, pages 18–23, 2007.

[5] Freesim http://www.freewaysimulator.com/.

[6] LE Owen, Y. Zhang, L. Rao, and G. McHale. Traffic flow simulation using Corsim. *Simulation Conference Proceedings. Winter*, 2, 2000.

[7] Corsim http://ops.fhwa.dot.gov/trafficanalysistools/corsim.htm.

[8] K. Nagel and M. Schreckenberg. A cellular automaton model for freeway traffic. *Journal de Physique I France*, 2(2):2221–2229, 1992.

[9] R. Chrobok, J. Wahle, and M. Schreckenberg. Traffic forecast using simulations of large scale networks. *Intelligent Transportation Systems, 2001. Proceedings. 2001 IEEE*, pages 434–439, 2001.

[10] J. Fawcett and P. Robinson. Adaptive routing for road traffic. *Computer Graphics and Applications, IEEE*, 20(3):46–53, 2000.

[11] Trafficmaster http://www.trafficmaster.co.uk/.

[12] Y. Wang, M. Papageorgiou, and A. Messmer. Renaissance: a real-time freeway network traffic surveillance tool. *Intelligent Transportation Systems Conference, 2006. ITSC'06. IEEE*, pages 839–844, 2006.

[13] L. Li, W.H. Lin, and H. Liu. Type-2 fuzzy logic approach for short-term traffic forecasting. *Intelligent Transport Systems, IEEE Proceedings*, 153(1):33–40, 2006.

[14] N. Utamaphethai and S. Ghosh. Dicaf: A distributed architecture for intelligent transportation. *Computer*, 31(3):78–84, 1998.

[15] J. Lei, K. Redmill, and U. Ozguner. Vatsim: a simulator for vehicles and traffic. *Intelligent Transportation Systems, 2001. Proceedings. 2001 IEEE*, pages 686–691, 2001.

[16] QI Yang and H.N. Koutsopoulos. A microscopic traffic simulator for evaluation of dynamic traffic management systems. *Transportation Research Part C*, 4(3):113–129, 1996.

[17] E. Lieberman et al. Watsim: wide area traffic simulation model for freeways and surface streets. *75th TB Annual Meeting, Washington, DC*, 1996.

[18] L. Smith, R. Beckman, and K. Baggerly. Transims: transportation analysis and simulation system. Technical report, LA-UR–95-1641, Los Alamos National Lab., NM (United States), 1995.

[19] G. Cameron, B.J.N. Wylie, and D. McArthur. Paramics: moving vehicles on the connection machine. *Proceedings of the 1994 ACM/IEEE conference on Supercomputing*, pages 291–300, 1994.

[20] M. Fellendorf. Vissim: a microscopic simulation tool to evaluate actuated signal control including bus priority. *the 64th ITE annual meeting, session*, 32, 1994.

[21] A. Barisone, D. Giglio, R. Minciardi, and R. Poggi. A macroscopic traffic model for real-time optimization of signalized urban areas. *Decision and Control, 2002, Proceedings of the 41st IEEE Conference on*, pages 900–903, 2002.

[22] R. Hummel and S. Zucker. On the foundations of relaxation labelling. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 5(3):267–287, 1983.