Real-time urban traffic information extraction from GPS tracking of a bus fleet

Evangelos Denaxas*, Savvas Mpollas*, Dimitrios Vitsios*, Christoforos Zolotas*,
Dimitris G. Bleris†, Georgios M. Spanos‡ and Nikos P. Pitsianis*§
*Department of Electrical and Computer Engineering, Aristotle University, Thessaloniki, Greece.
†Link Technologies, Thessaloniki, Greece.
‡O. A. S. Th., Thessaloniki, Greece.
§Department of Computer Science, Duke University, Durham NC, USA.

***Abstract*—We present a novel system named Speed-O to estimate in real time the average speed of traffic on every section of urban roads in the city of Thessaloniki, Greece. Speed-O processes the telemetry data reported by the fleet of more than 600 public transportation buses. The system is solving for the mean speed at different temporal and spatial scales much finer than the raw telemetry data. Speed-O is comprised of a large and sparse model of the road system, the bus routes, and the fleet dynamics together with a fast solver that renders the model solution. The solver is multi-layer and iterative. It exploits the physical properties of the problem to operate efficiently in real-time on a multicore processor. We demonstrate the Speed-O system using the Google Maps API as an interactive map that refreshes every minute with the average speed displayed as color-coded road sections.**

I. Introduction

Traffic congestion is one of the most common problems in large cities and has grave consequences to health, financial and overall quality of life of all urban residents. Any system that measures and reports the speed of traffic flow accurately, offers great direct and indirect pay offs. Transportation is one of the great challenges of any aspiring smart city and a critical component of urban design and sustainability.

In this paper we present a working system named Speed-O that utilizes the information from GPS sensors installed in the public transportation fleet to estimate the average speed of traffic. Novel processing algorithms are used to estimate the average speed of traffic in a hierarchical representation of the city road network in much finer detail than other existing state-of-the-art systems. The output of Speed-O is an accurate real-time traffic model that allows the study of city traffic patterns at various times and conditions. It offers the opportunity for new services for the commuting public, commercial service providers and emergency vehicles.

Conventional traffic monitoring systems are based on inductive road bronchus systems or traffic cameras in several locations in the roads. These systems are accurate, as they make direct measurements, but they are also expensive to acquire, install, and maintain. Traffic cameras, in addition, have very high communication bandwidth and processing requirements.

Alternatively, the average speed of traffic can be determined by using data from dedicated GPS devices or cellular phones of participating drivers [1]–[5]. The idea to use the location data of commercial fleets to estimate road traffic conditions is not new, transit buses and taxis have been used as probes to traffic detection [6], [7]. It has been found that speed is a better traffic parameter than travel time to be directly used in modeling bus-car relationships using the average speed of vehicles to determine the traffic in urban areas.

The paper is organized as follows: Section II describes the telemetry data acquisition. Section III describes the model formation and Section IV presents two methods for solving it. Section V presents preliminary results. We conclude with a discussion in Section VI.

II. Telemetry data

The position acquisition and collection system of the bus fleet was initially created and developed by *Link Technologies*, a private company specializing in telemetry applications and *O.A.S.Th.* the public transportation company of the city of Thessaloniki and its extended urban area. O.A.S.Th operates more than 600 buses in 76 routes that carry over 150 million passengers annually.

The main functionality for the telemetry has been to provide on-board automated announcements of the approaching bus stop in multiple languages and also to display the bus route identification and arrival time estimation at an online web site and smart bus stop information screens. The equipment consists of an embedded system that contains a GPS module to assess its position and instantaneous speed and a GSM module to transmit it and other identification to a central data base via the cell phone network. Collected data record the time stamp, instant velocity, longitude and latitude coordinates every 20 seconds. In addition, this information is used to evaluate the performance of the bus fleet overall and in the design of new routes. The Speed-O system utilizes this pre-existing and paid-for infrastructure to inexpensively recover the speed of road traffic of a metropolitan area, via the algorithmic processing of real time position data.

III. Traffic model

In this section we describe the traffic model data structure, and the synthesis of the GPS positions into a linear system of equations.

The roads of the extended urban area of Thessaloniki are represented by the geographic information system (GIS) in terms of road network segments called shapefiles. A shapefile is a standarized open format data vector that describes a geographic feature by its geometry, for instance, point coordinates of a road. Attributes provide additional information such as the road name, and whether it is a local street or a highway. The length of shapefiles vary from a few tens of meters in city streets, to several hundreds of meters in highways. Speed-O uses the GIS shapefile information to form its main data structure, a directed graph of the road network used by the buses. Additional data structures are linked lists of these shapefile identifiers that represent the bus routes. The traffic model data structure is the above graph annotated with the average speed of traffic for each edge, for a given time interval.

A. *Route linked lists*

A preprocessing phase expresses each bus route as a linked list of road shapefiles. This is necessary to make the projection of the GPS coordinates to the route to be accurate and efficient. Oftentimes there is a big error in position when the GPS satellites are obscured by very tall buildings. The mapping of the GPS positions denoting a bus route to a sequence of shapefiles also known as map matching is a non-trivial endeavor [8]. As successive position reportings are 20 seconds apart, several shapefiles need to be identified to connect the two points. Moreover, the positions do not necessarily fall over the correct road segment. The projection of the reported location to the nearest shapefile section might be erroneous and lead to the wrong assignment.

We solve this problem using a classic machine learning technique. We take historic data for each route and identify the $k = 8$ nearest neighboring shapefile line sections to each location point. Each shapefile with line sections within the k nearest gets a vote. The shapefiles with the lowest number of votes are eliminated if they do not affect the connectivity of the route. A final pass eliminates the “brush” or “comb” effects where side roads and intersections to the route were included and correctly selects the correct way for two-way streets.

The linked list is required to be recomputed whenever a bus route is changed or a new route is introduced, or when a new map is used.

B. *Speed equations*

The average speed on the urban road network is extracted by solving a linear system of equations that associates the successive GPS position data for each vehicle with the distances traveled on the road network.

This is done in the following way. A subset of the bus positions within a specified time window are retrieved periodically from the central data base. The time interval may vary from as often as 20 seconds up to several minutes in order to estimate urban traffic conditions over different time periods, with varying accuracy and real time performance. The basic information acquired this way consists of bus id, route id, bus position as it is calculated by its GPS system, its instant speed, and a time stamp.

Successive positions of the same vehicle reveal the time it took to traverse a road section. The traveled distance can be estimated accurately by projecting the GPS locations onto the vehicle route. The average speed estimate is the ratio of the distance over the time to traverse the road section.

A linear equation is formed from a pair of successive positions and the corresponding time stamps. The left-hand-side of the equation is the summation of the partial times to traverse each road segment shapefile. The partial times are expressed as the ratios of the shapefile lengths over the corresponding unknown average speeds. The right-hand-side is the time difference between the time stamps. Only the shapefiles of segments that connect the two points in the specific route have non-zero coefficients. Partially traversed shapefiles are represented with the fraction of their length.

Thus, the linear system $Ax = b$ is composed of equations due to pairs of successive GPS positions for each vehicle that reported within a given time period Δt. Each row of the system matrix A marks the segments that a vehicle has passed through, expressed as a fraction of their total length. The vector of the right-hand-sides b has the corresponding elapsed time between the two GPS reports. The vector of the unknowns x is the lengths of the shapefiles divided by the corresponding average speed at that shapefile.

The shapefile is the finest common unit between the routes. If two routes use the same section of a street, then the corresponding shapefiles are in their linked lists. Thus information from more than one vehicle that happen to use the same road in the same time interval can be combined to provide more accurate and or finer results.

IV. Efficient information extraction

We describe in this section two approaches for the fast solution of the dynamics model introduced in the previous section. The size of the model increases with that of the fleet, the time interval and the resolution and complexity of the city road map. We utilize however the sparcity of the system and advanced computing techniques for reaching a solution.

When the synthesis of the linear system results to a square invertible or over-determined matrix, a non-negative least squares solver is used for its solution [9], [10]. However, when the linear system is under-determined, we need to select a possible solution among the infinite number of solutions that satisfy the linear equations. We employ the following two different methods, inspired from signal processing.

A. *Least gradient*

The property of varying the least in the gradient, allows us to solve under-determined linear systems $Ax = b$ by considering them as an optimization problem with linear constraints:

$$\begin{aligned} x_{\mathrm{LG}} = \quad & \mathrm{argmin} \|\nabla x\|_2 \\ s.t. \quad & \\ & A x = b \end{aligned} \tag{1}$$

where ∇ denotes a special discrete gradient operator. The ∇ implements the central difference of the components of the unknown vector x that correspond to components that are adjacent in the road network. In matrix expression, ∇ is an $n \times n$ matrix with rowsums equal to 0; the i^{th} row associates road segment variable x_i with all other road segments that it is adjacent.

The solution to the least gradient (LG) problem (1), is accomplished in two steps. First, we obtain a particular solution x_p to the linear equation $Ax = b$. Then, the general solution is given by $x = x_p - N\,c$, where N spans the null space of A, and c is an arbitrary coefficient vector. The problem (1) is reduced to a linear least squares problem without constraints,

$$x_{\rm LG} = \arg\min_{c} \|\nabla(N\,c - x_p)\|_2.$$

The solution to the LG problem (1) can be expressed as follows

$$x_{LG} = x_p - N(N^{\rm T}\nabla^{\rm T}\nabla N)^{-1}(\nabla N)^{\rm T}\nabla x_p, \qquad (2)$$

assuming that the ∇N is of full column rank.

The estimation model (1) is essentially the same to the model used in [11].

Unfortunately, the LG method does not guarantee non-negative solutions. Truncating negative solutions to zero may result to a sub-optimal solution. On the other hand, a non-negative least squares solver has a prohibitive run time for the problem sizes we are concerned with Speed-O.

Our only alternative has been the use of an iterative algorithm that efficiently solves non-negative under-constrained linear systems presented next.

B. Nested adaptive refinement

The Nested Adaptive Refinement Estimation (NeAREst) [12], is a very efficient multi-level iterative solver for situations where locality of spatial smoothness and sparsity are not easily regulated across different resolution scales, or where minimizing total variation in gradients, as in the previous subsection, may sacrifice accuracy or fail.

In NeAREst, it is assumed that the unknowns can be represented by a nested structure of models starting at a coarse quantization level to the finer and target quantization level. The main ideas behind this solver are the following: (i) At the coarsest level it is easy to find a solution with a few simple maximum likelihood iterations starting from any initial guess. (ii) The solution of a coarser level can be used as the initial guess for the next finer level. (iii) The convergence of maximum likelihood can be dramatically accelerated.

There are multiple urban traffic models, with different quantization levels and weights corresponding to a nested hierarchy of coarser to finer road segment lengths that conform with the bus routes. By conformity of bus routes we mean that at all hierarchy levels, a road segment never leaves from or arrives to an interior point of another road segment. Therefore all connections of road segments are between starting and ending points, no mater the coarseness resolution (scale level).

The system model at any scale level s is of the same form

$$b = {\rm A}_s x_s.$$

Once a solution is approximated for scale s, it is used as an approximation for the initial solution for scale $s + 1$.

In reconstruction models, the measurement equation at each scale is replaced by a variational formulation to address properties or features of the solutions (such as non-negativity), as well as in the measurements (such as the presence of noise). In hierarchical quantization, estimates of the unknown quantity at different levels are consistent in the sense that the change between the solutions at two neighboring levels is within predictable bounds. In other words, this hierarchical consistency condition describes and restores the physical nature that is lost in a fixed discretized measurement equation expressed at a single quantization level. Thus, it effectively narrows down the permissible solution domain from the null space of the measurement equation at the target level, which is linear and most likely high dimensional.

The main component of NeAREst is the Richardson-Lucy (RL) iteration [13], [14]. Starting with a positive initial estimate x_0, the RL-iteration at step k is given by

$$\begin{array}{rcl} b_k &=& A\,x_k \\ m_k &=& (1/c)\cdot A^T(b/b_k) \\ x_{k+1} &=& x_k \cdot m_k \end{array} \qquad (3)$$

for $k = 0, 1, 2, 3, \cdots$, where the element-wise product is denoted by $u{\cdot}v$, vector c denotes the column sums of matrix A, and the vector division operations are performed element-wise. It has been shown that the Richardson-Lucy iteration converges when the operator matrix A is of full row rank [12], because at each iteration the Kullback-Leibler (KL) divergence of the quantity

$$D(b, b_k) \overset{\rm def}{=} b^T \log{(b/b_k)} = b^T(\log{(b)} - \log{(Ax_k)})$$

is reduced, i.e. $D(b, b_{k+1}) < D(b, b_k)$. In essence the NeAREst method iteratively solves the KL divergence optimization problem

$$\min_{x>0} D(b, Ax)$$

for any $A \geq 0$ and $b \geq 0$. Moreover, the RL iteration converges to the maximum likelihood solution [15].

A relaxation step between every $d \geq 2$ RL-iteration steps, is performed using a quasi-Newton optimization; for instance, taking a step along the steepest descent direction with a Broyden line search. The descent direction is:

$$F(y) \overset{\rm def}{=} -\nabla g(y) = x(y) \cdot (A^T(b/(Ax(y))) - e).$$

with function $x(y) = \exp{(y)}$, function

$$g(y) \overset{\rm def}{=} b^T(\log(b) - \log(Ax(y))) - e^T(b - Ax(y))$$

and e denoting the vector of ones of appropriate dimension. The relaxation step is calculated as

$$x_{k+1} = \exp(\Delta y_{k+1}) \cdot x_k$$

with

$$\Delta y_{k+1} \quad = \quad F(y_k) - \frac{\Delta y_k^T F_k}{\Delta y_k^T \Delta F_k}(\Delta y_k + \Delta F_k), \qquad (4)$$

where $\Delta F(y_k) = F(y_k) - F(y_{k-1})$ is available from two previous RL steps.

The NeAREst method is more tolerant to the presence of noise and it does not require the calculation or estimation of the null space of A that can be computationally expensive.

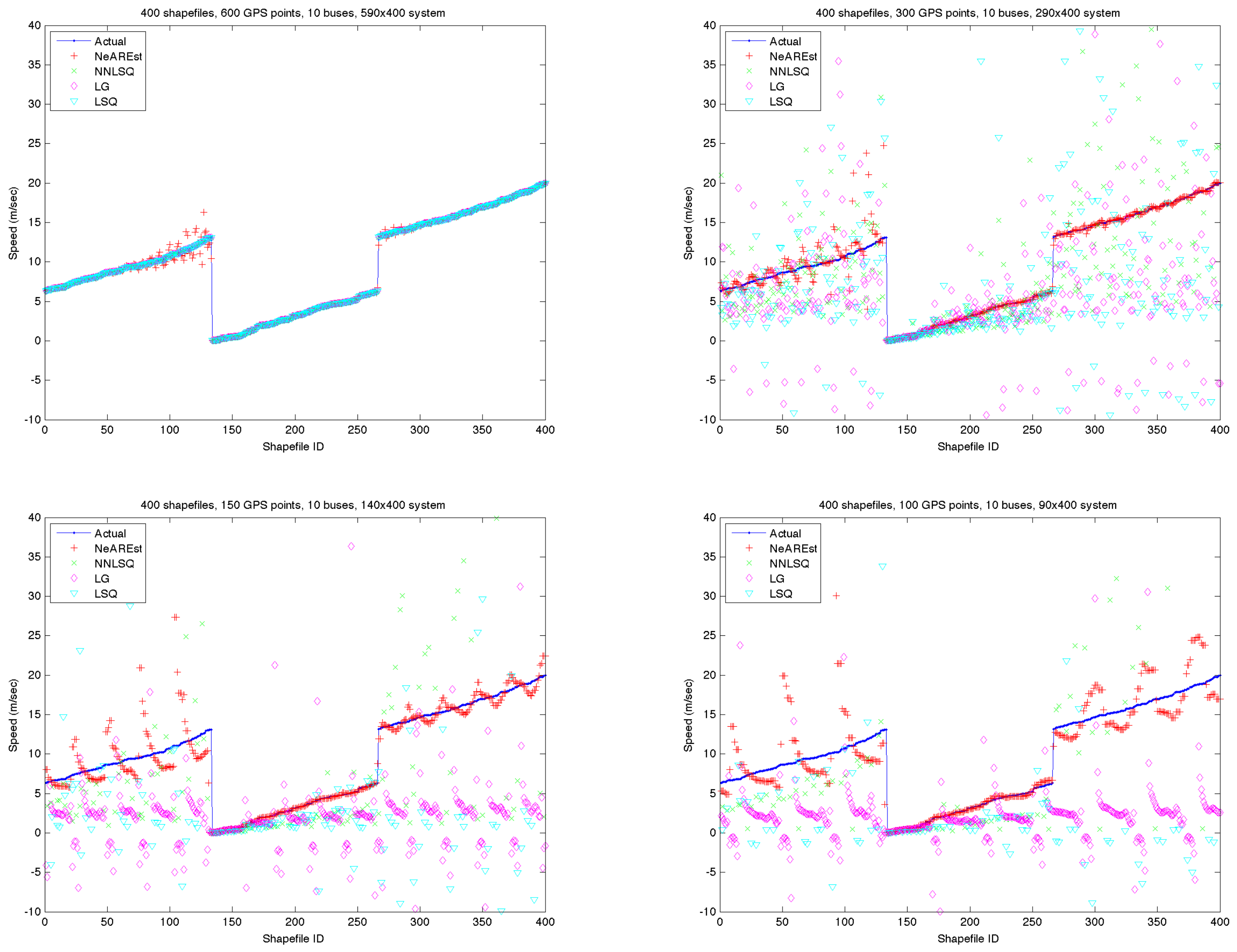

Fig. 1. Numerical accuracy tests in average speed with synthetic data of the following methods: least squares (LSQ in cyan triangles), non-negative least squares (NNLSQ in green x), least gradient (LG in magenta diamonds), NeAREst (in red cross) and true answer in connected blue dots. The range of the y axis range has been constrained manually to maintain consistency among the plots; out of range results are omitted.

V. Results

We have designed a synthetic test data set with known answers for quality assurance validation. The test data set is constructed as follows. Shapefiles are defined as one dimensional line segments that form a single linked list with random lengths. Shapefiles are assigned random average speeds. In order to provide some structure, the speeds are sorted in ascending order and then the first third is assigned in the middle set of shapefiles to introduce regions of nearly smooth but random values with two sharp discontinuities. GPS points are randomly distributed along the shapefiles and the corresponding time stamps are calculated according to their position and average speeds. The above test data are then fed into the processing pipeline and the results are evaluated and displayed. The synthetic data span the range of the values we expect to encounter. We are in the process of designing more realistic test scenarios using MATSim, a framework to implement large-scale agent-based transport simulations [16].

Figure 1 shows the averge speed solution computed by different algorithms for tests with 10 buses and 400 shapefiles. The tests differ in the number of GPS points, and effectively the number of rows of the resulting linear system matrix. The number of unknowns remains constant. The plots are showing the average speed recovered using the following numerical methods: least squares (LSQ in cyan triangles), non-negative least squares (NNLSQ in green x), least gradient (LG in magenta diamonds), and NeAREst (in red cross). The true answer is also displayed for reference, in connected blue dots. The range of the y axis range has been constrained manually to maintain consistency among the plots, out of range results are omitted.

In the first test, (top left), 600 GPS points are introduced and the synthesized linear system is of size 590×400. Since the system is overdetermined and consistent, all methods converge to the true answer.

In the second test, (top right), 300 GPS points are introduced and the synthesized linear system is of size 290×400. Although the linear system is underdetermined, the NeAREst method estimates the correct answer accurately. The other methods roughly approximate the correct speeds.

In the third test, (bottom left), 150 GPS points are introduced and the synthesized linear system is of size 140×400. The linear system is severely underdetermined, as there are almost 3x unknowns than equations. The NeAREst method still estimates the correct answer extremely well. The other methods fail to approximate the correct speeds.

In the fourth test, (bottom right), 100 GPS points are introduced and the synthesized linear system is of size 90×400. This time the linear system is extremely underdetermined, with more than 4x unknowns than equations. The NeAREst method still converges to an albeit noisy approximation of the average speeds.

We have also experimented with the actual traffic data feed. Figure 2 shows average traffic speed results in four hierarchical resolutions as displayed using the Google maps API [17]. The time intervals Δt we have experimented with are: 1 minute, and 5 minutes. Bus positions reported within the time interval are fetched from the central data base. The linear system is formed, the average speed is determined and the results are exported as an html file using Google maps. The processing is completed before the expiration of the time interval. The total data transfer and processing time for the five minute interval is approximately 180 seconds and for the 1 minute about 40 seconds on a 24 core computer. The number of unknowns of the linear system varies in the order of 3 to 5 thousands. We have observed that depending on the traffic conditions, there can be up to twice as many unknowns as equations. The above results do not contain any post processing of the data for denoising or smoothing.

We are in the process of conducting field tests to validate the results of Speed-O against traditional traffic measurement systems. We are also working on models to establish the correlation between general traffic and bus traffic, especially for the road segments that contain a bus stop or where a traffic lane is reserved for the exclusive use of buses.

VI. Discussion

With this paper we showed that the Speed-O system with advanced processing algorithms and efficient implementations can recover average speed information in real time utilizing the existing telemetry infrastructure of a bus fleet. The least gradient and nested adaptive refinement algorithms, presented in this paper, appear to be highly competitive, efficient, cost-effective and scalable to handle the traffic network of any metropolis.

The availability of accurate and fine-detailed real-time city-wide traffic information data will open new opportunities for on-line route planning services based on computational learning and prediction.

Acknowledgment

This work has been supported by the Greek General Secretariat for Research and Technology.

References

[1] F. W. Cathey and D. J. Dailey, "Transit vehicles as traffic probe sensors," *Transportation Research Record: Journal of the Transportation Research Board*, vol. 1804, pp. 23–30, 2002.

[2] ——, "Estimating corridor travel time by using transit vehicles as probes," *Transportation Research Record: Journal of the Transportation Research Board*, vol. 1855, pp. 60–65, 2003.

[3] P. Chakroborty and S. Kikuchi, "Using bus travel time data to estimate travel times on urban corridors," *Transportation Research Record: Journal of the Transportation Research Board*, vol. 1870, pp. 18–25, 2004.

[4] R. L. Bertini and S. Tantiyanugulchai, "Transit buses as traffic probes: Use of geolocation data for empirical evaluation," *Transportation Research Record: Journal of the Transportation Research Board*, vol. 1870, pp. 35–45, 2004.

[5] B. Coifman and S. Kim, "Using transit vehicles to measure freeway traffic conditions," in *9th International Conference on Applications of Advanced Technology in Transportation*, Chicago, Ill, Aug. 2006.

[6] W. Pu, J. J. Lin, and L. Long, "Real-time estimation of urban street segment travel time using buses as speed probes," *Transportation Research Record: Journal of the Transportation Research Board*, vol. 2129, no. 1, pp. 81–89, 2009.

[7] L. D'Acierno, A. Cartenì, and B. Montella, "Estimation of urban traffic conditions using an Automatic Vehicle Location (AVL) System," *European Journal of Operational Research*, vol. 196, no. 2, pp. 719 – 736, 2009.

[8] C. Goh, J. Dauwels, N. Mitrovic, M. T. Asif, A. Oran, and P. Jaillet, "Online map-matching based on Hidden Markov model for real-time traffic sensing applications," in *IEEE Intelligent Transportation Systems Conference*, 2012.

[9] C. L. Lawson and R. J. Hanson, *Solving least squares problems*. Prentice-Hall, 1974.

[10] G. H. Golub and C. F. Van Loan, *Matrix Computations*. The Johns Hopkins University Press, 1989.

[11] M. Shankar, N. P. Pitsianis, and D. Brady, "Spatio-temporal sampling for video," in *Image Reconstruction from Incomplete Data V*, vol. 7076. SPIE, 2008, p. 707604.

[12] X. Sun and N. P. Pitsianis, "Solving non-negative linear inverse problems with the NeAREst method," in *Advanced Signal Processing Algorithms, Architectures, and Implementations XVIII*, vol. 7074. SPIE, 2008, p. 707402.

[13] W. H. Richardson, "Bayesian-based iterative method of image restoration," *Journal of the Optical Society of America*, vol. 62, no. 1, pp. 55–59, January 1972.

[14] L. B. Lucy, "An iterative technique for the rectification of observed distributions," *The Astronomical Journal*, vol. 79, no. 6, pp. 55–59, June 1974.

[15] L. A. Shepp and Y. Vardi, "Maximum likelihood reconstruction for emission tomography," *IEEE Transactions on Medical Imaging*, vol. 1, no. 2, pp. 113 –122, 1982.

[16] "`http://www.matsim.org`."

[17] "`https://developers.google.com/maps`."

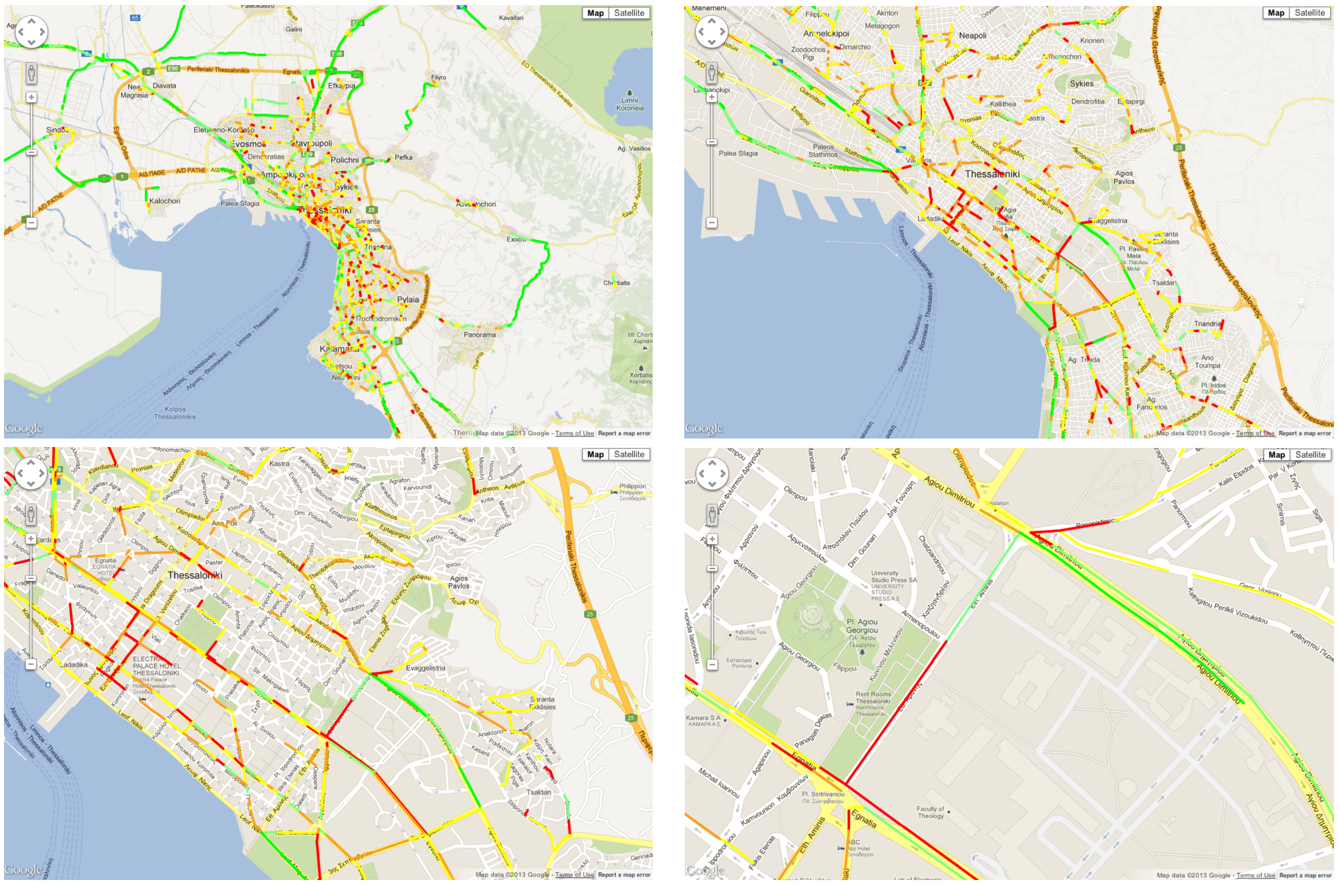
Fig. 2. Preliminary average traffic speed results with real data in four hierarchical resolutions as displayed using the Google maps API.