6th International Conference on Applied Informatics
Eger, Hungary, January 27–31, 2004.

Intelligent urban traffic development support system—the simulation software and the database [*]

Attila Fazekas, Lajos Kollár, Zoltán Zörgő, András Hajdu, János Kormos, Krisztián Veréb

Institute of Informatics, University of Debrecen
e-mail: {fattila,kollarl,zorgoz,hajdua,kormos,sparrow}@inf.unideb.hu

Abstract

In this paper we present the architecture an urban traffic surveillance system based on smart sensors capable of license plate recognition. A data model that is suitable to store data coming from this system is also discussed. This data model needs to be connected to the spatial description of the underlying road network.

Due to the relatively high cost of the sensors, a software for simulating vehicle movements and detection by the sensors is needed, as well. We have developed two applications for this purpose.

Categories and Subject Descriptors: H.4 [Information Systems Applications]; J.7 [Computers in Other Systems]; K.4 [Computers and Society]; E.1 [Data Structures]

Key Words and Phrases: Traffic simulation, Data model

1 Introduction

An increasing problem in the modern world is the rapidly growing urban traffic. The highly populated cities are not prepared to ward off the effects of this growth, such as traffic jams, flow slowdown, not speaking about the environmental pollution and the psychological effects. In most cases the lack of space is the major issue. The public road development as part of the urban planning has to be performed in the most appropriate way.

In an urban environment vehicle counting alone may not be enough to acquire satisfying information about the behaviour of traffic participants, their route choosing habits. Knowing not only quantitative information, but concrete vehicle paths, we can determine more precise origin-destination profiles, patterns and trends in the traffic. Therefore we intend installing sensor devices capable of recognizing license plates. These sensors will be placed on previously selected lanes in a well delimited area. Knowing the precision

[*]This work was supported in part by the NKFP2 grant 2/2002, and by the OTKA grants T032361 and F043090.

of the devices and simulating the urban traffic in that specific area, we can propose the minimum number of sensors needed in the actual data acquisition, and the position of the sensors.

Our aim is to create an intelligent urban traffic development support system to monitor urban traffic, and to predict traffic behavior. To reach this goal a project has been started with a consortium (the members are the Adaptive Recognition Hungary Ltd. [1] and the Institute of Informatics, University of Debrecen) to manage the development and deployment. In the last years the work has been separated into two large parts, namely the development for software and database techniques and simulation software tools and a development for a statistical model. The second part is explained in [2].

Our goal is to collect information about the urban traffic which can be the starting point of a statistical analysis which provides useful data for the designers and maintainers of the public road network in order to use them for both tuning the legacy network and further development.

This paper is organized as follows. Our 4-layered system architecture is discussed in Section 2. Section 3 describes the data model used for storing data acquired by the sensors. In Section 4 we discuss the need for a simulation software and two implementations are proposed. Section 5 contains concluding remarks.

2 Architecture

The architecture of our system is a 4-layered one. Each layer is based on the layer below it and provides services for the above layer.

The bottom layer is the *physical road network* itself.

Spatial model stores the physical location of both the roads and sensors. It provides an abstraction of the road network. It is considered as a "digital picture" of the underlying road network. Sensors form a directed graph called sensor graph. These sensors are considered to be *smart*: they are not only counting the number of cars but gather more information about vehicular traffic, including license plate identification of the crossing vehicles and (optionally) the photo of the license plate.

Since statistical model needs aggregated data which is not supported by the spatial model, we need a new layer, the inputs of which are the data gathered by the sensors and provides the required aggregates.

Hence, collected information from each sensor are inserted into a *database*. However, sensor graph can be derived from the spatial model, the need for faster processing suggests to store the graph in a database, as well. Therefore the database, besides storing data produced by the sensors, contains data about the sensors, as well, which need to be connected to the sensor descriptions of the spatial model.

Data are collected in order to create a forecast, e.g., about the preferred paths of drivers at Easter, or the average speed of vehicles in the morning, etc. To do so, complex *statistical computings* are needed. These issues are discussed in detail in [2].

3 Data model

As we mentioned, data collected by the sensors are inserted into a database. Each sensor has a unique ID, knows the timestamp of the photo, the recognized license plate of the

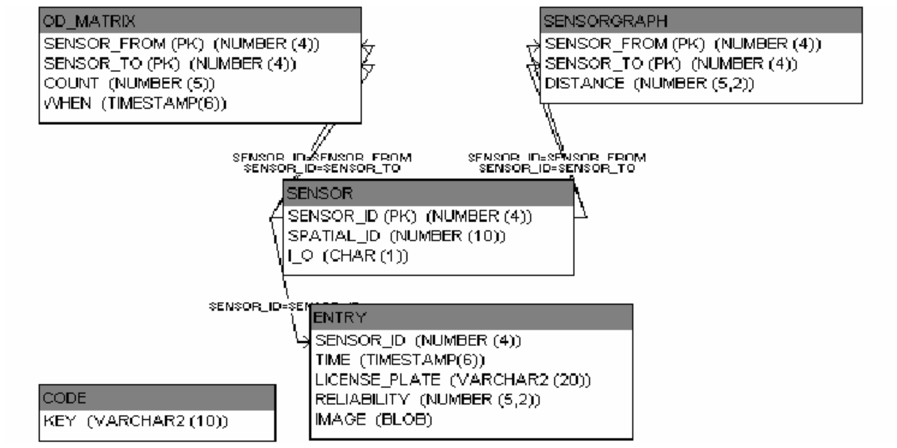


Figure 1: Data model used for storing sensor data

vehicle, and optionally, the reliability of the recognition and the photo itself (in JPG or BMP format). These data serve as a base for the statistical analysis. To store the gathered information we have developed the data model shown in Figure 1.

This model is based on the relational model for databases [3]. Therefore, all the entities of Figure 1 are mapped to a database table and SQL should be used for data manipulation.

Derived data from the spatial model should be stored, as well. The physical location of sensors is given by the spatial model. To achieve independence of the spatial model, sensors are assigned unique identifiers which can connect to the spatial model using a database table serving as an interface between the spatial and the data models. This table needs to map each sensor as a spatial model element to a generated ID. This solution makes the data model independent of the underlying spatial description.

`SENSOR` stores the information about sensors. We have three sensor types used when computing origin-destination (OD) matrices: input, output and pass-through.

As we mentioned before, sensors form a sensor graph. This can be derived from the spatial model but for faster processing we need to store it in the database, as well. `SENSORGRAPH` stores which sensors have direct connections and the distance of them. The latter can be used for computing average speed of vehicle flow, for example.

`ENTRY` stores data acquired by sensors. For each vehicle, the time of recognition is recorded as a timestamp which can be used for computing trends and seasonalities. The result of the recognition is a license plate. The reliability of the recognition and the image taken may also be provided by the sensors. The former allows the statistical correction of the erroneous data caused by the incidental incorrect recognition.

The table `OD_MATRIX` is used from time to time for aggregating information needed to compute OD matrices.

`CODE` is a singular table (its singularity is maintained by a database trigger) used for security reasons but its detailed discussion is outside the scope of this paper.

3.1 Functions, services

Logical data independence as a general principle in database design suggests to have the data model independent of the other parts of the system which results in avoiding data manipulation directly in the tables (i.e., the subsystem controlling sensors should not insert new entries using direct INSERT SQL statements, and the statistical model consuming the produced aggregates should be prohibited from direct SELECTs). An interface should be defined through which other parts of the system can access database services such as determining average speed between two sensors or computing the OD matrix.

The interface can be implemented using stored subprograms because the change of the database structure results in refactoring this code only and the other parts of the system are not affected.

Besides trivial services, such as adding, re-locating or deleting a sensor, querying sensor type or deleting obsolete data, this interface should provide all the information which are needed for the statistical analysis. Production of OD matrices, determining both the average speed of a vehicle and the number of appearing/disappearing vehicles, and appointing the flow rating of a given sensor are included in these services and they need to be extensible.

3.2 Security issues

Since personal data are stored (the technology allows the tracing of whole paths of given vehicles), it is very important, how data are protected against incompetent access. Although data are stored itemised (i.e., there are entries for every recognized license plate), no one should be allowed to access these detailed data. Only the number of vehicles are important so aggregated data need to be provided, i.e., one can get information about the number of vehicles on a given path but cannot access data of concrete ones.

We have created an interface on top of the data model which consists of several stored procedures and functions. For security reasons, only the interface is allowed to manipulate itemised data, i.e., no one but the interface is authorized to access database tables directly. Thus, data protection is played back to the security mechanism of the underlying DBMS.

Since sensors are intended to send data to the database over network, data need to be secured not only in the database but in the course of network communication, as well. Accordingly, license plates should be encrypted using cryptographical methods, e.g., DES3. Decryption should be done before inserting into the database and the proposed interface approach provides an easy way to do so.

3.3 Implementation

The proposed data model can be implemented in any RDBMS or ORDBMS, e.g., Oracle, MySQL, PostgreSQL, etc. For implementing interfaces, it is subservient to choose such a DBMS which supports stored subprograms. If they are not supported by the DBMS, interface can be implemented in a high-level programming language, e.g., Java or C++, which accesses the database via JDBC or ODBC.

For our implementation we have chosen Oracle9*i* as a back-end because of its high performance, wide support for aggregations (data warehouse operations) and its powerful

PL/SQL [4] language for creating stored subprograms.

4 Simulation software

For testing our system we needed many sensors to be deployed. As they are “smart”, e.g., equipped with a camera, their costs are relatively high. For this reason a simulation software has been developed which is able to simulate not only the movement of vehicles but license plate recognition, as well. Beside these, the program has to be able to generate common traffic events, and to follow some of the behavioural differences between drivers. This kind of software can be used not only for testing purposes but tuning the minimum number of sensors to be deployed to a given area.

There are some traffic simulation projects, but this last feature is not included in any of them. So we need to develop our own software for this task.

The results of the simulation will be used not only in planning the data acquisition, but in refining the requirements for the sensors, which are currently developed by our partners.

4.1 Specifying requirements for the simulation software

First of all we need to specify the desirable features for the software. Some of these requirements are absolutely necessary; some of them could help in a more accurate and more realistic urban traffic simulation.

4.1.1 Simulating road system and traffic

It is a basic feature to store a *model of the road system* in an easily expandable way. A too complicated storage may cause performance problems. As we do not intend to simulate whole cities, only smaller regions of a few square kilometres, we do not calculate with large storage needs. The internal format must contain some geographical positioning information about the road sections. The ability to identify the lanes separately is also a must. For a more precise driver behaviour simulation, the visibility and terrain conditions could also be represented. For an easy data exchange with the local Transportation Departments, the application should have some built-in import features for the most commonly used *map formats*. As we noticed no trends in this matter in Hungary, we haven't focused on none of the alternatives.

As streets are either *one-way* or *two-way* ones, we cannot ignore this in a simulation. But as we all know, there are some exceptions too—like vehicles with distinctive flare. It would be interesting to simulate a traffic situation in which police or fire engines are involved.

As urban traffic is happening in time, and it is far from being constant in time, many statistical descriptors need to be calculated for specific time intervals. Thus we need to simulate the *timeline* too, even if the virtual time flows quicker or slower as our one. Thus, a timestamp can be attached to the data acquired.

As in real life, traffic flow may be controlled with *traffic lights*. When implementing this feature we need to able to simulate concrete traffic lights or light system programs. We need to identify the lights controllers separately to be able to simulate the outages, or the overnight caution light program.

It is obvious that not all vehicles have the same *speed*, thus we shouldn't ignore this. This involves some very complicated problems, like overtaking, and this, on it's turn involves the need of representing the size and some other physical parameters of the vehicles. We have to mention, that in the current state we have ignored this attributes, and we have calculated with some average values.

As traffic in general is guided by *traffic signs*, urban traffic is highly influenced by traffic rules. These ones cannot be ignored. It is not absolutely necessary to actually store the traffic signs, but to incorporate their effect in the internal representation. Nevertheless, the first alternative is more obvious. As not all drivers respect these regulations, why should a simulated driver be different? The simulation of a realistic urban driver is a very challenging problem, involving artificial intelligence and psychological research. In the current stage we reduced this task onto specifying some stochastic parameters for the virtual car objects.

4.1.2 Simulating data acquisition

As mentioned above we need to simulate the activity of the proposed sensors and the data acquisition process. This way—with regard to, and using the chosen statistical methods—we can give some guidelines for placing the physical sensors, the number of sensors needed, and the expectable result reliability.

First of all it is necessary to specify the position of the virtual sensors. The manual placing of the simulated sensors is inevitable, but in some cases, beside the manual one, random placement could be helpful. Automated full-range coverage with sensors of a selected region is also useful, because the starting point for the optimisation may be such a fully covered state.

Besides their own optical range limitations, there can be situations in which part of the field of vision of a sensor is permanently covered by field objects. So, when specifying position—including the height over the road surface—and the viewing direction of the device, the actual range could be also specified.

To track the traffic flow in commonly used routes consisting on many consequent streets, virtual path definition could also be useful.

4.1.3 Exception event simulation

There are normal events occurring during a journey. Some of them are actual traffic events, like jam, or lane lockdown. These events will come trough as irregularities in the data flow. Another sort of exception is caused by hardware or software error. As such, not all exceptions are errors. Let's take for example the case when a car disappears. That means it is not seen by the sensors for a while. The most common event this represents is that the vehicle has ended its itinerary, and it stopped somewhere between two sensors. But of course this can be caused by misrecognition of the license plates too.

It can happen that a vehicle jumps over a sensor. This can be an error in the recognition, or the license plate may temporary covered by another car, so the car is not "seen" by sensors.

It is possible that a car stops, but after a while, it is back again. This situation should not affect the calculation of the average time for a street.

But some events are real errors occurring in the software of the sensor devices, the hardware or in the network.

So we can see that the simulation software should be able to generate such exception events, but handling such data by the statistical model is a very challenging problem.

4.1.4 Simulation output

The simulation software is expected to provide well formatted output data, which can be used as input for the statistical analysis on one hand, and can be interpreted by humans to follow the events standing as base for the data. Thus we expect more than one output files.

The main data file must have the same format and content logic as the files acquirable with the sensor system. The timestamp contains common hour–minute–second time information. When simulating traffic over many days, the date information must also be included. If the sensor devices are capable of generating the same identification data for a specific license plate, it is not absolutely necessary to store the actual license plate content. This could be another improvement in security. The reliability factor is calculated by the sensor itself for every recognition separately. This "self-rating" should also be simulated. It can happen to have a correct sensor recognition, but low level of sureness. To implement this feature, we need to know the recognition algorithm.

Another data file can contain information about the events generated by the simulation.

4.2 Implementations

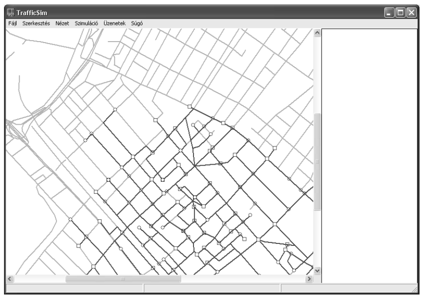
Because our simulation needs contain very special requirements, we found no appropriate software available for the public. That is why we began to implement different combinations of features. At this point we have two implementations. The first one is a GUI (Graphical User Interface) based Windows program, the second one is a console application.

4.2.1 TrafficSim

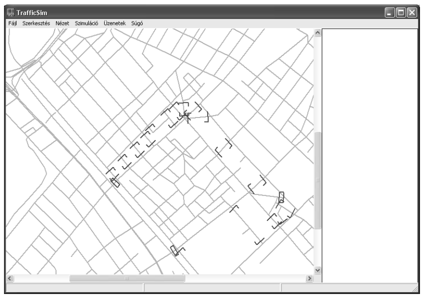
As mentioned above this software is a graphical application. It was developed in collaboration with our partners, the Adaptive Recognition Hungary [1] Ltd. This application supports MapInfo Interchange Data Form-style file import. Traffic flow can be predefined for a simulation but we can use random traffic too. This can be stored and reloaded to be investigated under other conditions. The program can transmit using TCP/IP communication recognition event data to a server application. The users can define virtual paths with adjustable flow weights. Sensors can be placed on different parts of a road; the height from the ground can be specified too. Users can Save and Load virtual paths and the position of the sensors.

In the initialization file we can specify parameters for the recognition error simulation: missed recognitions, bad character recognition, extra tailing characters, extra leading characters, extra characters in the middle and missing characters at the beginning, the end and in the middle of license plates.

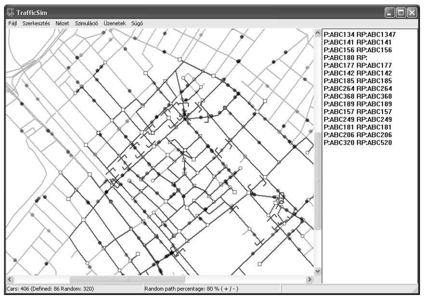
Screenshots of the application are shown in Figure 2. The spots in Figure 2(c) represent the virtual vehicles. During the simulation we can see the vehicles moving. A record to the "node" log file is added every time a vehicle reaches a road junction. The "sensor" log file contains the sensor identification, the actual license plate text and the simulated recognition. The “path” log’s entries consists of the nodes hit by the vehicles.



(a) Virtual paths loaded



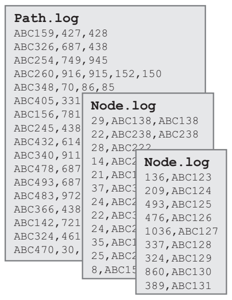
(b) Sensors loaded



```
P:ABC134 RP:ABC1347
P:ABC141 RP:ABC141
P:ABC156 RP:ABC156
P:ABC180 RP:
P:ABC177 RP:ABC177
P:ABC142 RP:ABC142
P:ABC185 RP:ABC185
P:ABC264 RP:ABC264
P:ABC368 RP:ABC368
P:ABC189 RP:ABC189
P:ABC157 RP:ABC157
P:ABC249 RP:ABC249
P:ABC181 RP:ABC181
P:ABC206 RP:ABC206
P:ABC320 RP:ABC520
```

```
Cars: 406 (Defined: 86 Random: 320)    Random path percentage: 80 % ( + / - )
```

(c) Simulation in progress



```
Path.log
ABC159,427,428
ABC326,687,438
ABC254,749,945
ABC260,916,915,152,150
ABC348,70,86,85
ABC405,331
ABC156,781
ABC245,438
ABC432,614
ABC340,911
ABC478,687
ABC493,687
ABC483,972
ABC366,438
ABC142,721
ABC324,461
ABC470,30,
```

```
Node.log
29,ABC138,ABC138
22,ABC238,ABC238
28,ABC222
14,ABC2
21,ABC1
37,ABC3
24,ABC2
22,ABC3
24,ABC2
35,ABC1
25,ABC2
8,ABC15
```

```
Node.log
136,ABC123
209,ABC124
493,ABC125
476,ABC126
1036,ABC127
337,ABC128
324,ABC129
860,ABC130
389,ABC131
```

(d) Fragments from output files

Figure 2: TrafficSim screenshots

4.2.2 CrossRoads

Features This software is a console application, currently with no interactive interface. Road-system and simulation parameters can be specified in an XML file with special DTD. The application can simulate traffic light programs, supports multiple lane for a road section. It is timeline based. The user can define nodes to be reached by a vehicle. Every vehicle (driver) has its own affinity to quicker or shorter tracks. This is a stochastic threshold values upon which the program chooses the direction in the intersections.

The simulation process The database of the program consists of the lists of the vehicles, intersection, road sections and sensors. The program picks every vehicle one by one and checks the state of them. A vehicle can be either moving towards a node or waiting at an intersection. The position (the lane on which it is, and the relative position from the lane end) of the vehicle is stored in the vehicle object.

If the vehicle is on its way on a lane, the program checks whether it’s reaching the end of the section. In this case chooses a new direction according to the predefined route or, if not present, the affinity of the driver. Otherwise checks if the vehicle has reached a sensor. In this case the virtual recognition is performed, and the result is logged.

If the intersection is controlled by traffic-lights, the vehicle is enrolled in the queue, according to its direction. During the wait-state the vehicle is idle. If there is no traffic light or finds clear signal, then the vehicle is passed to the desired road. If the vehicle reaches an output node, then it is removed from the active vehicle list.

The simulation ends if there are no more vehicles on the roads, or it is interrupted.

The input file format As mentioned before, this implementation uses as input an XML (Extensible Markup Language) file with special DTD (Document Type Definition). This way the introduction of new features is very simple. The definitions contain human-readable information besides the relational and other technical ones.

5 Conclusions

The development of easily expandable and configurable traffic simulation software is a must for our project in order to simulate real traffic events, vehicle movements and sensor detection. The database gets the simulation output transparently as it would come from real sensors. This allows the switching to real sensors without affecting both the data model and the statistical model using the services of the data model.

If the traffic simulation software is developed according to well-defined requirements, such a software could be used in other projects too. Our implementations do not meet all desirable requirements, but they were suitable for the basic statistical models. Our future aim is to create a more realistic, and more accurate software for traffic simulation.

References

[1] Adaptive Recognition Hungary Ltd., `http://www.arhungary.hu/`

[2] András Hajdu, János Kormos, Krisztián Veréb, Attila Fazekas, Lajos Kollár, Zoltán Zörgő, *Intelligent urban traffic development support system—statistical approach*, 6^{th} ICAI, to appear.

[3] Ramez Elmasri, Shamkant B. Navathe, *Fundamentals of Database Systems, Fourth Edition*, Addison-Wesley, 2003.

[4] Steven Feuerstein, *Oracle PL/SQL Programming, Third Edition*, O'Reilly, 2002.

Postal addresses

András Hajdu, Attila Fazekas, Lajos Kollár, János Kormos, Krisztián Veréb, Zoltán Zörgő
Institute of Informatics
University of Debrecen
P.O. Box 12, H–4010 Debrecen
Hungary

6th International Conference on Applied Informatics
Eger, Hungary, January 27–31, 2004.

Intelligent urban traffic development support system—statistical approach [*]

András Hajdu, János Kormos, Krisztián Veréb, Attila Fazekas, Lajos Kollár, Zoltán Zörgő

Institute of Informatics, University of Debrecen
e-mail: {hajdua,kormos,sparrow,fattila,kollarl,zorgoz}@inf.unideb.hu

Abstract

In this paper we present a statistical model that is suitable to evaluate data coming from an urban traffic surveillance system. The detection of the traffic flow is based on a sensor system using machine vision, which is able to determine the license plate characters of the vehicles. By extending former approaches of using origin destination matrices, we define a topological layer as a theoretical background for the suitable statistical model. To reduce the charges of the system we also study the effect of dropping different kind of sensors, as well.

1 Introduction

Our aim is to create an intelligent traffic development support system to monitor urban traffic, and to predict traffic behavior. To reach this goal a project has been started by a consortium (with the participation of the AR-Hungary Co. Budapest and the Institute of Informatics, University of Debrecen) to manage the development and deployment. Our work has been focused on two large parts, namely on composing a statistical model and on developing collateral software and database techniques and simulation software tools. In this paper we present our results according to the first part, while those about the second one are shown in [4].

Our statistical model for evaluating the observed data is primarily based on the topology of origin-destination (shortly OD) matrices. This way we can determine the traffic flow within different urban areas and also can investigate the flow between them. As basic statistical measures we calculate: flow rate (vehicle/h), density (vehicle/km/lane) and mean speed (km/h). Observing the dependencies between these parameters we can derive descriptive statistics. Namely, we can consider speed as the function of flow, speed as the function of density and flow as the function of density. Determining local/global minima/maxima of these functions we can describe the behaviour of free-flow speed, jam-density, capacity-flow and critical density.

[*]This work was supported in part by the NKFP2 grant 2/2002, and by the OTKA grants T032361 and F043090.

2 Monitoring the traffic, deploying sensors

Detecting vehicles and monitoring the traffic are based on the usage of sensors. Many types of them were developed for collecting quantitative information (the passes of vehicles). Such sensors are the inductive loops built in the road or infra-red gates located at the roadside. In some cases (for example using sensor-pairs) more information (e.g. speed, vehicle length) can be collected. The usage of machie vision (camera sensors) leads to a more complex model. The advantage of this approach is the possibility to collect additional information (color, size, shape) or even unique identifiers (e.g. license plates).

The appropriate number and placing of sensors is a challenging problem in traffic surveillance. If we want to collect only flow rate and load data, then the location of the sensor for a particular road segment is not that important. The main question in this case is the number of the sensors. If we monitor more lanes of a road, we have to use more sensors. If the observed area is a crossroad or a region, we have to use sensors for every entry and exit points, as well. (If some sensors are missing, then it will be impossible to detect all of the traffic jams in the observed area.) In case of a larger observed area the stopping and starting vehicles also may yield problems.

The descriptive statistics we considered (for describing traffic behavior) can be classified into three groups. The first one contains information about the lanes or lane segments which can be the flow rate, mean speed, etc. If the maximum flow rate is known, then the ratio of the current and maximum flow rates give the load of the lane. If the load tends to one, then the possibility of a traffic jam increases at the given measuring point. The changes of these values according to the time are also important statistics. The second group contains statistics for crossroads. Using them we can describe e.g. the preferred directions of a crossroad. After a large number of observations we can calculate the probability by which a path (output) will be chosen if a vehicle enters a crossroad. (Clearly, the sum of the probabilities for the output paths equals one). The changes of these probabilities against the time (the changes of the preferred paths) are also important descriptiors of traffic behaviour. The third group contains statistics for regions. We can build up OD matrices for a particular region, based on the observations performed at all of its entry and exit points. The precise theoretical model will be given later.

There are many factors affecting the statistics which are classified into two groups in our model. The first group contains the unobservable events caused by the stopping and starting vehicles during the observation time which obviously increase the error of our statistical results. The second group contains irregular traffic situations, like traffic jams and possible accidents. These two groups must be unambiguously separated from one another.

Before using our system under real traffic conditions we performed simulation to test our methods. In the following we list what traffic behavior was modeled by a simulation software for getting necessary data. When the flow rates and the load is examined, it is indispensable to simulate the speed of the vehicles, and to assign a constant speed to all of the vehicles in the system is not sufficient. (The higher speed gives larger volume and flow rate, but the load will not increase, while slow vehicles give lower flow rate, but may cause traffic jams). In the simulation of speed we also considered the effect of traffic signs: speed limit or STOP signs and traffic lights, as well. For the prediction of the preferred paths we also have to handle the one-way roads. (Especially, if the given road is an entry or exit observation point.)

3 Origin-Destination (OD) matrices

To perform observations, we consider such areas, where sensors are deployed at their entry and exist points. The literature refers to these areas as Origin-Destination (shortly OD) matrices (see e.g. [6]) in such systems that are based on mainly quantitative data, when the sensors are able to detect only the passes of the vehicles. Accordingly, the (i, j) element of the OD matrix is defined as the number of passes between the entry point i and exit point j. OD matrices can be also considered dynamically, when the elements of the matrix change as a function of time. Many sensors applied in existing systems (e.g. inductive loops built under the surface of the road) do not allow unique vehicle detection. Thus, numerous theoretical approaches (see e.g. [1, 3, 2]) were developed to estimate the traffic flow based on the number of entries and exists. These techniques naturally can be improved by using an intelligent sensor system that provide unique vehicle detection, as well. As we consider license plate recognition in our system, the unique detection is guaranteed here.

As the above mentioned OD matrix estimations are not reliable under general conditions, in the existing systems the network topology is usually restricted to the simple motorway model [6], shown in Figure 1, where some license plate reader sensors are also deployed beside the traditional inductive loops. For a more general network model only few results are known, see [5].

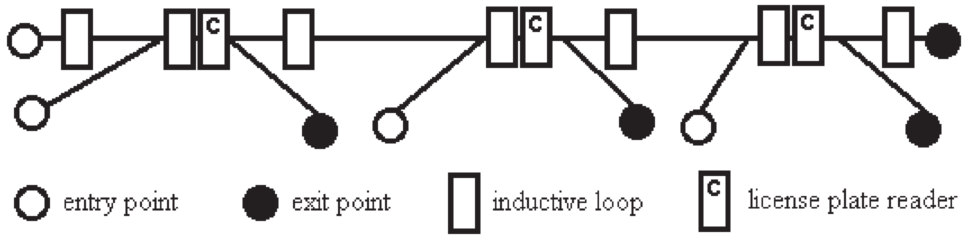


Figure 1: Motorway topology for classic Origin-Destination matrices.

As our sensor system is an intelligent one with respect to unique vehicle detection, we can extend the OD matrix model to more general (urban) traffic conditions without ruining the efficiency of the estimation of the traffic flow. We introduce a relation on the set of the road map points using the sensor system in the following way. Two points of the physical road map are in relation if either of the points can be reached from the other one without captured by a sensor. It is obvious that this relation is

- reflexive (a point can be reached from itself without captured by a sensor),
- symmetric (if we are not captured during traveling to one of the points to the other, we also will not be captured if we go back in the same way),

- transitive (if we can reach the second point from the first and also the third from the second without being captured, we can reach the third point from the first by taking the concatenation of the previous two paths),

and so is an equivalence relation. Thus, using this relation we can divide the road map into disjoint classes, and we call these classes complete OD matrices. Intuitively, we divide the road map into disjoint areas in such a way that a vehicle always will be captured by a sensor if it leaves or enters an area, where the sensors reside on the "boundary" of the area. Since these boundaries separate the OD matrices, the main advantage of this method is that we can monitor traffic flow as entries and exists into and from OD matrices, respectively. We can perform the following observations based on this approach:

- monitoring the traffic of a closed urban zone (main directions of the through traffic, rush hours, most polluted parts, etc.). The observation is based on the boundary sensors, but we can deploy some additional inner sensors to have better insight for the inner traffic,
- monitoring motorway-like urban routes, see Figure 1 (most popular entry and exist points of the route).

Using the above described OD matrix model, we can make observations not only for individual OD matrices, but also for their relations. Namely, OD matrices can surround others, or can reside at large distances. In case of "inclusion" (note that OD matrices are disjoint sets so this is only an intuitive terminology), when an OD matrix "contains" more OD matrices, we can monitor the following behaviour of traffic flow based on the distribution of the traffic of the surrounding zone with respect to the inner zones:

- locations of traffic jams inside the surrounding zone,
- most popular parking locations according to the number of stopping vehicles,
- pollution changes during the observation period according to the degree of loading of the inner zones,
- suggesting less loaded routes to pass through the surrounding zone.

When OD matrices reside at large distances we can perform the following observations:

- finding most popular (loaded) routes through zones, which help to determine the reasons of traffic jams in given zones,
- suggesting less loaded routes to reach a destination zone through intermediate zones.

At the definition of complete OD matrices we assume that every entry and exit points of the desired zones are observed by sensors. However, to decrease the charges of the system the number of the sensors should be minimized without loosing too much traffic data to keep statistical analysis reliable. On the other hand, if we decrease the number of sensors, the above introduced theory of OD matrices will ruin theoretically. To avoid this problem we handle the decrement of the number of sensors by introducing the concept

s_0	s_1	s_2	s_3	s_0	s_1	s_2	s_3
86	203	312	1491	0.011628	0.019704	0.016026	0.024816
101	198	280	1566	0.019802	0.010101	0.021429	0.021073
93	200	320	1521	0.010753	0.030000	0.015625	0.019066
100	195	289	1556	0.020000	0.015385	0.024221	0.021208
93	193	312	1483	0.021505	0.020725	0.009615	0.020904
⋮	⋮	⋮	⋮	⋮	⋮	⋮	⋮

Table 1: Valid flow rates and the difference of the valid and empirical ones for sensors $s_0, \ldots, s_3$.

of a random p-OD matrix with respect to a complete OD matrix, where $0 \leq p \leq 1$. Namely, in the p-OD matrix we can observe the $(p * 100)\%$ of the data of the complete OD-matrix, where the p-OD matrix contains the same road map points as the complete one, but having less sensors on its boundary. As a natural consequence, the number of starting and stopping vehicles will increase in a p-OD matrix causing larger error in traffic estimations. However, by leaving less important sensors (those ones which serve relatively small number of detections) the increment of error can be kept low, and the estimation remains reliable. In practice, this process can be realized by creating the estimations according to the observations of the complete OD matrix first. Then, after decreasing the number of sensors, we recalculate the estimations according to the poorer observation data, and the parameter p can be determined based on the comparison of resulted estimations for these two cases.

4 Experimental results

Generally, the technique for monitoring an area with fix number of sensors is based on the usage of OD matrices. In our tests, we deployed 50 sensors with the TrafficSim software (see [4]) to observe a valid area, which is a region of Budapest XIV (more precisely, a rectangle bounded by the roads Thököly, Lumumba, Mogyoródi, Mexikói between the boulevards Hungária and Nagy Lajos). The output of the simulation software is a set of files containing the ID of sensors, the original license plates and the observed license plate triplets, so the database layer can compute the measured data. The observation time was about 11 hours. Each sensor detected about 40 000–50 000 passes. The observed area had 20 exit sensors, 18 entry sensors and 12 inner sensors. The sensors was set up at the distance 10-15 meters from the crossroads. More technical details can be found in [4]. The following traffic parameters were calculated:

Valid flow rates for each sensor (vehicle/hour). The output table contains 50 columns with respect to the sensors $s_0, \ldots, s_{49}$, and 11 rows (for each observed hour). According to the i-th sensor and j-th hour the entries of the table are denoted by $v(s_i)_j$ (shortly $v_{i,j}$). For a part of such a table see Table 1.

Empirical flow rates. Similarly to valid flow rates, these values are also contained in a table, whose entries are denoted by $m(s_i)_j$ (shortly $m_{i,j}$).

Differences of the valid and the empirical flow rates. The difference values ($|m_{i,j} - v_{i,j}|$) are organized into a table as well. The error ($|m_{i,j} - v_{i,j}|/v_{i,j}$) is about

2%, which comes from the detection error of the sensors. The empirical mean value of the differences is 2.091%, shown in the Table 2.

The computing of flow rates does not use the license plate, only the quantitative data (vehicle passes). However the prediction of the preferred paths are based on the recognition of the plates. (A valid pass can be counted if the license plate has appeared also at an entry and an exit sensor in the given ordering).

For the 11 observing hours the system creates the particular OD matrices. So we have 20 exit and 18 entry sensors, thus the number of theoretical paths between them is 360. (Actually, there are more paths, but for the measurement we can consider only 360 different paths). For every hour we have three OD matrices data. The first is the valid OD matrix data, the second is the observed one, and the third is the difference of the first two. At this point not only sensor detection error appeared, but also the error of the stopped and started vehicles (there are input detections without output detections and vice versa). The error is between 10% and 50% (the mean value is approximately 27%).

The 12 inner sensors is used for defining paths through the observed region. From the observations we can state, that the transits follow the shortcuts in general. (The inner sensors with large flow rates are in between input and output sensors with large flow rates, too). So in the first step we can drop the inner sensors to reduce the charge of the whole system.

From the valid and empirical OD matrices data and from the error between them, we can derive that the mean value of the difference of the total entries and exits is about 2%, and this value is about 26% for transits. See also Table 2, where **vp** is for the valid passes, **op** is for the observed passes, **d** is for their difference and **%** is for the error in percentage.

entry points				transits			
vp	**op**	**d**	**%**	**vp**	**op**	**d**	**%**
21387	20952	435	2.0339	13895	10154	3741	26.9234
21389	20925	464	2.1693	13938	10295	3643	26.1372
21370	20930	440	2.0590	14088	10345	3743	26.5687
21444	21038	406	1.8933	14067	10252	3815	27.1202
21380	20923	457	2.1375	13961	10314	3647	26.1228
⋮	⋮	⋮	⋮	⋮	⋮	⋮	⋮

Table 2: Valid (vp) and observed passes (op), their difference (d) (in percentage, as well (%)).

5 Prediction opportunity

From the location of the sensors and the measured data, we can determine the most loaded roads (namely the roads Thököly and Mexikói). The changes of the flow rates can be seen in the sequence diagram.

The preferred way can also be determined from the measured data. If the flow rates have suddenly changed in some measuring point, from the change of the preferred paths

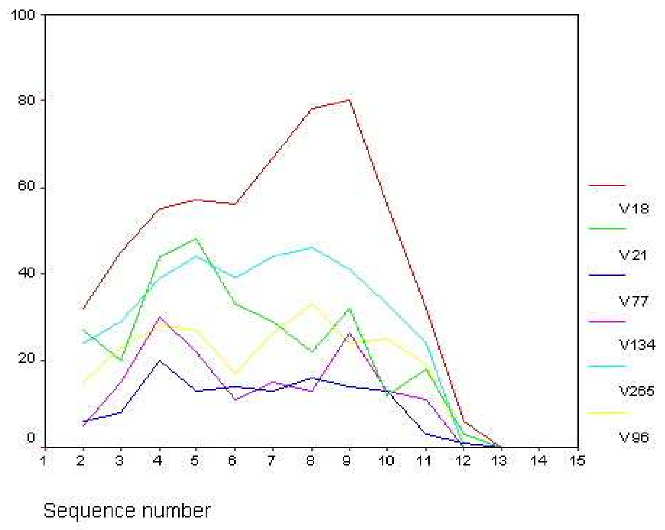


Figure 2: Sequence diagram for the changes of flow rates (v_i's) of different sensors.

we can localize the area of a possible accident. From longer observation periods we can derive trends according to the behavior of the traffic in the observed area. At this point a question arises, namely, what about the missing sensors?

6 Missing sensors

Let us see what happens if some entry sensors are missing. In our tests, three sensors with small flow rates were dropped. The average flow rates of the missing sensors were 417, 253 and 311 (per hour), respectively. The total number of observed vehicles was 10791 which is about 4.5% of the total entries. Our results are shown in Table 3, where **oe** is for the observed entering vehicles and **e%** is for the new detection error in percentage, **ot** is for the observed transit vehicles and **t%** is for the new transit error.

This case the error groves to 6.0% and the error of transits to 30.18%. We also dropped three sensors with large flow rates. The rates were 3795, 2407 and 1911, respectively, which yields 89268 unobserved vehicles (37%). The results can be seen in Table 3. The detection error in this case is 39.9%, while the transit error is 57%.

We also investigated the effects of dropping exit sensors. As it is expected, the results are similar to the case of the entry sensors. First, three sensors were dropped with small (1%) flow rates: 94, 21 and 60 vehicles per hour, respectively. The detection error is 3%, and the mean value of the transit errors is approximately 27%. Table 4 contains our results, when sensors were dropped with large flow rates (3623, 1754 and 1410 vehicles per hour, respectively, so we cannot detect 37% of the exiting vehicles), where **ox** is for

small flow rates				large flow rates			
oe	**e%**	**ot**	**t%**	**oe**	**e%**	**ot**	**t%**
20006	6.4572	9680	30.3347	13032	39.0658	5906	57.4955
19983	6.5735	9808	29.6312	12809	40.1141	5818	58.2580
19958	6.6074	9849	30.0894	12762	40.2808	5889	58.1985
20015	6.6639	9741	30.7528	12943	39.6428	5898	58.0721
19905	6.8990	9795	29.8403	12812	40.0748	5879	57.8898
⋮	⋮	⋮	⋮	⋮	⋮	⋮	⋮

Table 3: Transits (ot) and their error (t%) in case of missing entry sensors having small and large flow rates.

the observed exits, **x%** for its error, **ot** is for the observed transits and **t%** is for its error. Now, the detection error is about 39%, while the transit error is 58%.

ox	**x%**	**ot**	**t%**
10901	39.7702	5727	58.7837
10956	39.3859	5898	57.6840
11000	39.5737	5892	58.1772
11102	39.0603	5870	58.2711
11082	39.1366	5875	57.9185
⋮	⋮	⋮	⋮

Table 4: Transits (ot) and errors (t%) in case of missing exit sensors with large flow rates.

When both types of sensors are missing, tests were classified into two groups. First some sensors were dropped with small flow rates. In this case the detection error is 4% for the inputs, 1% for the outputs and the total transit error is 31%. In case of sensors with large flow rates, the input detection error is 39%, the output detection error is 37% and the total transit error is 86%.

7 Summary

In this section we sumerize our experiments about the error term which is a key problem in traffic surveillance. It is obvious that with missing sensors we cannot explote the possibilities of the OD matrices, only the stand-alone sensor detection results (e.g. the flow rates and the load of lanes). Table 5 summarizes the errors for our tests, where **me** is for the missed vehicles in percentage caused by dropping entry sensors, **mx** is for the missed vehicles in percentage caused by dropping exit sensors, **ee** is for the entry error, **xe** is for the exit error, while **te** is for the transit error.

We can derive that the sensors with large flow rates are very important. Dropping sensors with large flow rates will deteriorate the efficiency of the measuring in a large extent. The only possibilty to avoid such faults is the usage of duplicate sensors.

me	**mx**	**ee**	**xe**	**te**
0	0	2	2	26
4	0	6	2	30
0	1	2	3	27
4	1	6	3	31
37	0	39	2	57
0	37	2	39	58
37	37	39	39	86

Table 5: Missed vehicles in percentage (me, mx) and the total errors (ee, xe, te) in case of missing sensors.

In our simulations, the difference of the valid and observed passes casued by the sensor detection error was fixed for 2%, and could not be changed dynamically. In future tests we aim to study such cases, where the detection error can be changed. These investigations should be separated into two subclasses. Namely, when all of the sensors have the same (but variable) detection error, and when the sensor detection errors are independent variables.

References

[1] B. Coifman (1998) Vehicle Reidentification and Travel Time Measurement in Real-Time on Freeways Using the Existing Loop Detector Infrastructure, Transportation Research Board 77th Annual Meeting, January 11-15, Washington, DC.

[2] M. Cremer and H. Keller (1987) A new class of dynamic methods for the identification of origin-destination flows, Transportation Research 21B, 117-132.

[3] M. Cremer and H. Keller (1981) Dynamic identification of flows from traffic counts at complex intersections, Proc. 8th In. Symposium on Transportation and Traffic Theory, University of Toronto Press, Toronto, Canada.

[4] Attila Fazekas, Lajos Kollár, Zoltán Zörgő, András Hajdu, János Kormos, Krisztián Veréb, Intelligent urban traffic development support system—the simulation software and the database, 6^{th} ICAI, to appear.

[5] H.D. Sherali and T. Park (2001) Estimation of dynamic origin-destination trip tables for a general network, Transportation Research 35B, pp.217-235.

[6] N.J. Van Der Zijpp (1997) Dynamic OD Matrix Estimation from Traffic Counts and Automated Vehicle Identification Data, Transportation Research Record 1607.

Postal addresses

Institute of Informatics
University of Debrecen
H-4010 Debrecen, P.O.Box 12.
Hungary