Ad-hoc adaptive cruise control algorithm

BALÁZS MEZNY, PÉTER LABORCZI, GÉZA GORDOS

Institute for Applied Telecommunication Technologies (IKTI), Bay Zoltán Foundation for Applied Research

{mezny, laborczi, gordos}@ikti.hu

Keywords: wireless ad hoc networks, adaptive cruise control, tempomat, algorithm, simulation

Nowadays, nearly all car manufacturers can build a cruise control system (tempomat) in their cars if the customer demands. Usually, these systems can maintain a certain speed, set by the driver of the vehicle. The tempomat system can be extended with a distance measurement sensor in some top-end luxury cars, to measure the distance of objects in front of the car (this can be another car or some kind of obstacle). By using these systems, a certain following distance can be set, and the tempomat tries to maintain it by using small amounts of acceleration or breaking according to the data provided by the sensor. These adaptive cruise control systems are expensive, and the detection range and field of view of the sensors are limited. In this paper, we present an adaptive cruise control system, which sets the speed of the vehicle according to messages distributed over an ad-hoc wireless network. The wireless communication eliminates the problems caused by bad visibility or being out of line of sight. The distributed messages contain the exact position, speed and direction of the sender vehicle. The described algorithm determines also the vehicle to be followed, when the driver turns on the tempomat.

1. Introduction

Nowadays, car manufacturers strive to get more customers with more and more built-in comfort services. These services help the driver, so the driver needs to exert less effort to drive. Occasionally, it could be hard to choose or maintain the appropriate speed. For example, a long trip on the motorway, when one has to maintain constant speed over several hours, can be exhausting. A cruise control system can assist the driver in this task. These kinds of ever improving cruise control systems are on the market for several decades from nearly every car manufacturer.

However, these cruise control systems are not able to adapt to the fluctuations of the traffic, they are only able to maintain a set speed, until the driver turns off the system. If the state of the traffic changes, for example an accident causes congestion on a motorway and cars have to decelerate then the regular cruise control systems cannot offer assistance to the driver.

Adaptive cruise controls (ACC) systems started to spread recently. The ACC systems use some kind of sensors to monitor the road segment in front of the car, and warn the driver in case of an obstacle on the road. Due to the integration of the built-in sensors and actuators, it is possible to intervene into the brake system of the car, and slow it, so the driver has time to react to the situation. The drawback of the ACC systems is that they can function properly only under nearly optimal conditions. ACC systems can not provide sufficient information for the driver in case of bad weather, or in a road curve where the front sensor is not able to monitor a long enough segment of the road.

In this paper we provide a solution to the described problems, by exploiting the possibility that wireless communication can be established among the vehicles [1], and accurate information can be provided for the speed regulation. The advantage of the wireless communication is that direct line of sight is not needed, so it remains operable in curves, furthermore, the range of the wireless communication is around 300 meters, nearly the double of the range of the radars used in ACC systems [2].

Another possible application could be in urban traffic. On an urban road it is not common that a certain velocity can be maintained for an extended period of time. Under these conditions the traditional cruise control systems can not be used, because it can not adapt to a changing environment. With the system presented in this paper, it is possible to set the vehicle speed according to the preceeding vehicle. When the preceeding vehicle decelerates then the following vehicle will slow down, and if it accelerates then the following car will adapt its speed to that as well. The driver using our system does not have to deal with the actual accelerations or decelerations; the embedded computer handles the calculations.

2. The algorithm

The cruise control algorithm chooses a target to follow from the vehicles in front of the car. The aim of the algorithm is to regulate the velocity of the vehicle, so the following distance between the target and the follower vehicle is in accordance with the current speed of the vehicles, maintaining a safe following distance. A minimal speed threshold of 20 kph is built into the algorithm. If the velocity of the vehicle is lower than this threshold then the speed regulation is turned off. This threshold was set due to the fact that at so low speeds the safe following distance is commensurable to the error of the GPS position.

The algorithm uses the data received through radio communication and the GPS information gathered by its own GPS receiver to calculate the current speed to be set.

Stored values for position information are as follows:

Lon:	longitudinal coordinate in radians up to 8 decimal places (accuracy to approximately 6 centimeters).
Lat:	latitudinal coordinate in radians up to 8 decimal places (accuracy to approximately 6 centimeters).
Vel:	length of the velocity vector in kilometer per hours.
Hdg:	direction of the velocity vector up to 2 decimal places.
Svs:	nr. of tracked space vehicles (GPS satellites) by the GPS receiver.
Tof:	timestamp of the GPS information.

The following information is also included in a message used by the algorithm:

Original sender ID:
The identification number of the originating unit of the message.

Sender ID:
The identification number of the unit, which retransmitted the message most recently.

TTL (Time To Live):
This sets how many times a message can be retransmitted. Our cruise control algorithm currently sets this value to 1, so the messages are propagated to a distance of one hop.

The algorithm calculates the distance between the vehicles by using the position information contained in the messages. A Kalman filter [3] was used to refine this distance calculation.

The *sender ID* and the *original sender ID* has to be checked in the received packet, if either one is equal to the ID of the *own ID*, the message is dropped.

The next step is to compare the heading contained in the packet to the vehicles own heading. If the difference is lower than a set threshold, the algorithm figures the two cars are heading the same way. This threshold is variable, currently set to 20 degrees.

If the headings match, it has to be decided whether the sender of the message is ahead of the own vehicle. This is a similar calculation like where the headings were compared. The own position vector is subtracted from the position vector in the message, and the difference vector is compared to the heading vector. If the difference between these two vectors is less than 90 degrees, the sender of the message is in front of us.

If these conditions are not met, then the sender of the packet is either behind us or is headed to another direction, and it cannot be followed. In this case the algorithm drops the packet.

If the sender has the same heading as our vehicle and is in front of it, it can be followed. In this case the state machine described in the following section processes the message further.

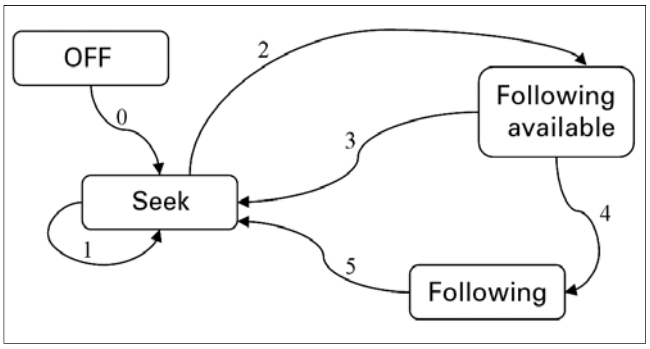
2.1. The state machine

The adaptive cruise control algorithm makes decisions according to the following stored variables:

– ID of the target (followed) vehicle.
– Distance to the target vehicle.
– Distance to the target vehicle in the moment when speed synchronization begins.
– Velocity of the target vehicle.
– Velocity of the target vehicle the moment when speed synchronization begins.
– Timestamp of the most recent message received from the target.

The algorithm works according to a state machine as shown in *Fig. 1*.

Fig. 1. State machine



2.1.1. Seek phase

When turned on, the algorithm is in this phase and waits for messages from other vehicles. It is decided here whether the actual following is possible, if the constraints described in the previous section are satisfied.

First the distance to the sender of the message is calculated by the Haversine formula [4], which returns the shortest distance between two points on a sphere. This calculation is done according to the GPS positions contained in the message and in the own receiver.

If the algorithm already has a target, it has to be decided whether the target sent the message or another car which is closer than the target.

If the message was sent by the target then the timestamp of the message and the distance is stored, and a counter is increased by one. This counter shows how many messages have been received from the same target.

If the message was not from the current target, but the calculated distance is shorter than the target's distance, then the originator of the message becomes the new target. The distance and the timestamp is stored in this case as well, and the counter is set to zero (transition 1 on Fig. 1).

If the algorithm didn't have a target, the calculated distance, the ID of the sender of the message and the timestamp of the message are stored. A counter is started from zero. This shows the consecutive messages received from the same target.

There are some thresholds in the algorithm, and exceeding these thresholds resets the state machine to the seek phase. This happens if the target vehicle's speed is lower than 20 km per hour or if no message is received from the target for five seconds. The counter is also set to zero. The stored target is unchanged, because it is still the closest vehicle in front of the own vehicle.

The following available phase is reached, if three consecutive messages have been received from the same target, while it remains the closest vehicle in front (transition 2 on Fig. 1). This threshold of three messages is a needed delay to ensure, that the chosen target is in a stable enough position to be followed. For example on a motorway when the following is engaged and someone overtakes both our vehicle and our target. When the overtaking vehicle is between us, the algorithm would decide, that it is the new target, because it is closer than the current target. At this time the algorithm would start to increase our speed to match the new target's speed. This would cause our vehicle to crash into the one in front of it, which was the real target vehicle.

2.1.2. Following available phase

In this phase there is a vehicle in front of our vehicle in a stable position and it's possible to follow it. The distance between the two vehicles is refreshed by every message received by the algorithm.

Transition 3 on Fig. 1 happens, if the target is changed. This could happen for example when someone overtakes us, or the current target overtakes another vehicle. The same thresholds are applied as in the seek phase. If the speed of the target is lower than 20 km per hour or 5 seconds passed without receiving a message from the target resets the state machine. This time the driver is able to reset the algorithm by pressing the corresponding button on the control panel.

2.1.3. Following phase

The algorithm enters this phase if the driver presses the engage button on the control panel (transition 4 on Fig. 1).

The algorithm resets in this phase as well, if the previously mentioned conditions are met, or when the driver turns off the following (transition 5 on Fig. 1).

Upon entering this phase the speed and distance of the target is stored. These values will be used in the calculation of the actual desired distance:

$$d_d = \frac{v}{v_0} \cdot (d_0 - l) + l, \qquad (1)$$

with the following notations:

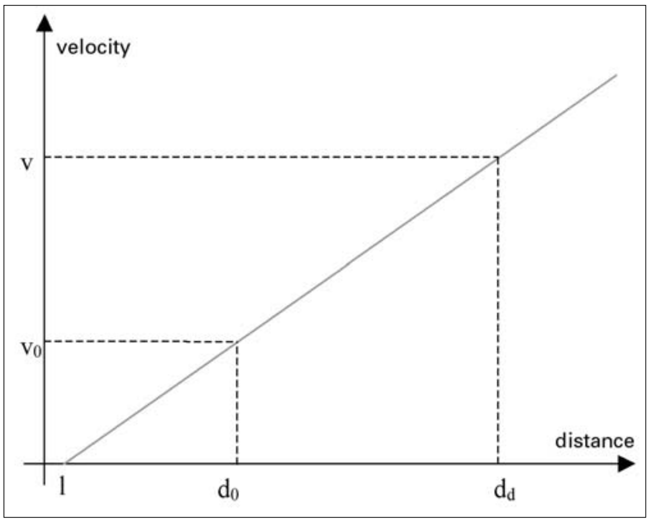
d_d: desired distance
v: current velocity of the target
v_0: velocity of the target when the following was engaged
d_0: distance of the target when the following was engaged
l: changeable parameter, the minimal distance to be kept between the GPS receivers

The distance values are the result of the Kalman filter, because the unfiltered distance values vary greatly since the position measurements do not happen in the same time. The GPS receiver used in our hardware has an update frequency of 1 Hertz, so under certain circumstances, for example at a speed of 50 km per hour, the difference between the GPS position and the actual position can be 13.8 meters.

The extension of the distance by the parameter *l* is needed, because the distance calculation gets the distance between the GPS receivers, not between the front and rear bumpers of the vehicles. This parameter can be set for various types of vehicles. It is currently set to 4 meters for passenger cars, but it could be set to for example 10 meters for trucks.

When the following starts, a line is defined according to the velocity and distance values at that time. When the speed of the target vehicle changes, the corresponding following distance can be calculated along this line *(Fig. 2)*.

Fig. 2. Following distance calculations



The following speed has to be set to reach the desired distance:

$$v_d = \frac{d}{d_d} \cdot v, \qquad (2)$$

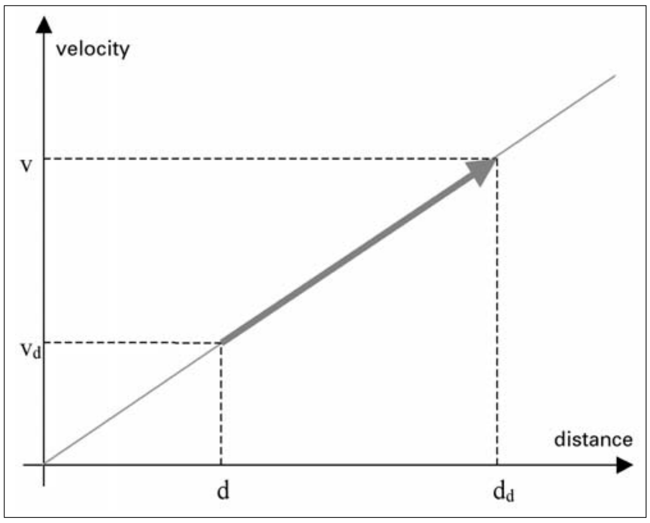
with following notations:

v_d: desired velocity to reach the desired distance,
d: current distance to the target,
d_d: desired distance,
v: current velocity of the target.

A hysteresis was built into the algorithm in order to avoid continuous intervention to the engine or brake systems. If the difference of the current distance and the desired distance is below 5 percent of the desired distance (d_d), the desired velocity (v_d) is set to the speed of the target.

The transition on the velocity-distance graph can be seen on *Fig. 3.* If the speed of the target changes, the corresponding following distance can be acquired according to (1). Using these values knowing the current speed, the velocity can be calculated to converge to the desired distance.

Fig. 3. Speed regulation



For example, as shown in Fig. 3, the target maintains a stable speed, and the following distance is set to a certain value (*d*) calculated by the algorithm. Both of the cars are traveling at the same velocity (v_d). If the speed of the target vehicle changes to *v*, a new desired distance (d_d) has to be reached. The distance between the vehicles will increase, because the target is moving faster than the other car. The algorithm will increase the velocity of the following car, as the distance converges to the desired distance, according to (2).

The acceleration has to be computed to adjust the engine control or the brake system properly. The calculation of the desired acceleration is as follows:

$$a_d = \frac{v_d - v_s}{T}, \tag{3}$$

with the following notations:
a_d: desired acceleration,
v_d: desired velocity,
v_s: current velocity of the own car,
T: free parameter to control the reaction time of the algorithm.

The maximal acceleration is 5 m/s^2 and the maximal deceleration is 9 m/s^2. These are the acceleration values of a typical passenger car.

To limit the number of repetitions, the acceleration is only computed if either of the following conditions is met:
- The distance of the target differs from the desired distance by more than 1 percent.
- The velocity of the target differs from the desired velocity by more than 5 percent.
- The difference between the speed of the target and the desired speed is greater than the difference between the desired speed and the own speed.

As a conclusion, the aim of the speed regulation is to reach the appropriate following distance. The engine is controlled according to the difference between the current speed and the desired speed (3).

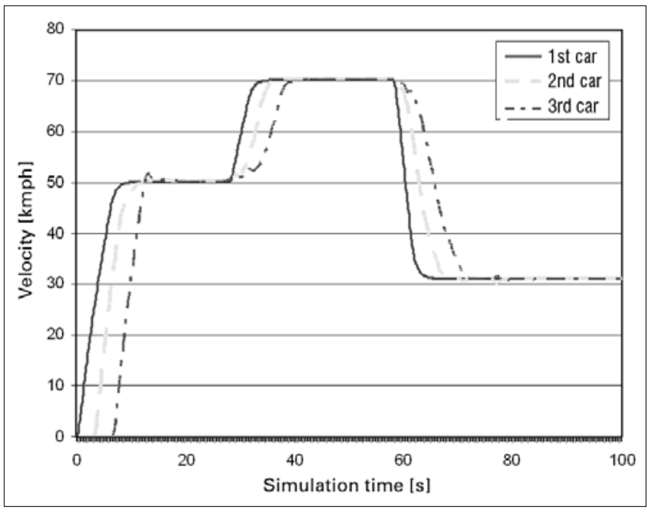
3. Simulation results

The testing of the algorithm was done on a simulator, described in [5] and [7]. At a later stage it was tested in real environment, embedded in vehicles.

Three cars were examined following each other in the simulator. The velocity of the first car was set to certain values. It started at 50 km per hour, at the third of the simulation time it was increased to 70 km per hour, and at two thirds of the simulation time it was decreased to 30 km per hour. The velocities of the following two cars were regulated by the algorithm. The messages containing the position information were broadcasted every tenth of a second to allow precise adjustments.

In the simulations a following distance of two seconds was set between the vehicles, because this is the minimal safe following distance. The results of this simulation can be seen in *Fig. 4.*

Fig. 4. Velocities of the vehicles



The vehicles accelerate to 50 km per hour at the start of the simulation, where the ACC algorithm is engaged.

It can be seen that the second and third cars reach their target's velocity with a little oscillation. It is also noticeable, that the acceleration of the following cars differs from their target's acceleration. This is due to the mechanism of the algorithm as it adapts the following distance to the changed velocity. In case of acceleration the following car drops behind a bit, and in case of deceleration it catches up on its target.

The distance between the cars was registered in the course of the simulation. The desired velocity calculated by the algorithm is based on this value. The various distance values between the first and second car can be seen on *Fig. 5.*

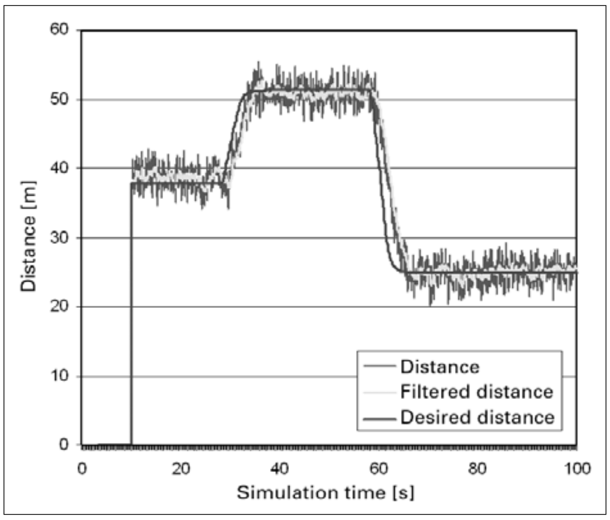


Fig. 5.
Distance values between the 1st and 2nd vehicles

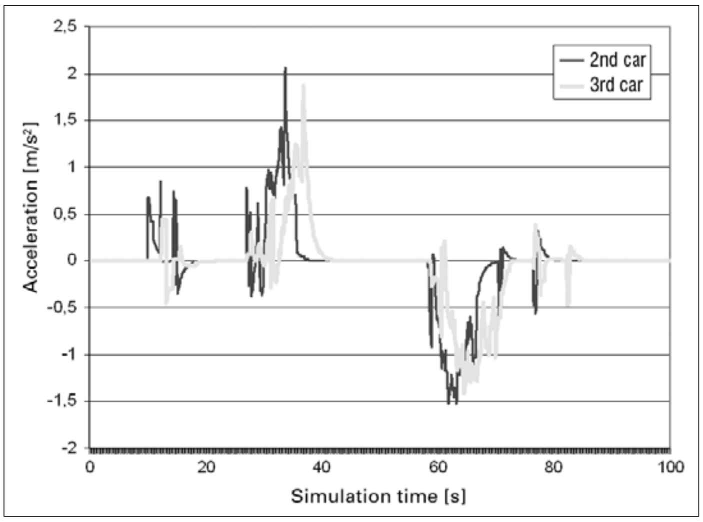


Fig. 6. Acceleration values

The algorithm was turned on at the 10th second of the simulation; the registration of the distance values was started at that point. The unprocessed distance value is shown by the narrow dark line. Though the position information is updated at a frequency of 10 Hz, this distance value is noisy due to the time difference of the position measurements and GPS error. Kalman filtering was used to eliminate most of the noise. The filtered distance values are shown by the bright line. The dark line shows the desired distance, which is calculated by the algorithm according to the actual speed of the target and the values stored at the start of the following phase. The algorithm tries to minimize the difference between the bright and the dark line by adjusting the velocity of the following vehicle. It can be seen, that after the transient caused by the acceleration or deceleration of the target, the distance between the cars settles at the desired distance.

The second and the third vehicle's acceleration values can be seen on *Fig. 6*.

The acceleration and deceleration periods can be examined on the graph, and it can be seen, that the threshold was never exceeded, where the algorithm would limit the acceleration value. The speed regulation algorithm would not cause uncomfortable acceleration or deceleration, as the maximal acceleration was around 2 m/s^2 and the maximal deceleration was 1.5 m/s^2. The noticeable acceleration spikes would be limited by the inertia of a real vehicle.

4. Field tests

The algorithm was tested under realistic circumstances. It was downloaded into two control units described in [8]. The control units were integrated into a passenger car and a truck, and the control unit in the truck was connected to the CAN bus, so the algorithm could operate the brakes and intervene to the engine control.

It can be noticed, that the speed regulation is not as precise as in the simulations for two reasons. The first is that the Kalman filtering was not implemented at the time of the test, the second is that the used GPS receivers had an update frequency of 1 Hertz, which is the tenth of the assumed frequency in the simulations.

The data of one of the test runs can be seen in *Fig. 7.*

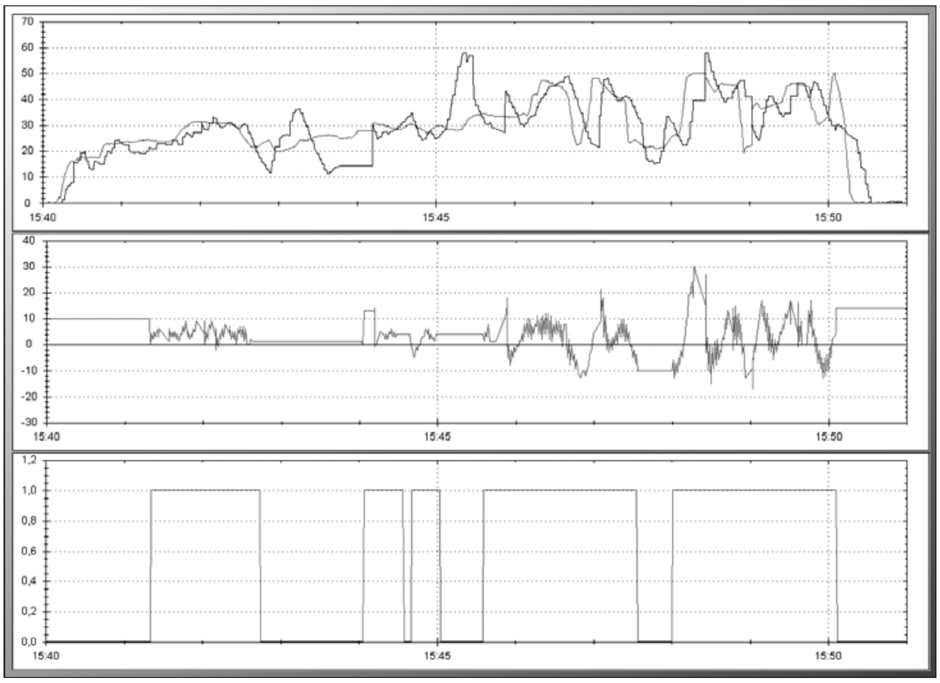


Fig. 7.
Field test results

The time on the horizontal axis is in minute units. On the top graph the velocities of the two cars can be examined. The darker line shows the following truck, the brighter line shows its target. The center graph shows the control value handed on to the CAN bus controller by the algorithm. This value is proportional to the acceleration, the exact coefficients for acceleration and deceleration had to be adjusted to the vehicle.

The bottom graph shows the status of the state machine. This value was 1 when the algorithm was in following phase, and the value was 0 otherwise. When the algorithm was engaged the truck adjusted its speed to the speed of its target with a little delay. This delay was caused by the infrequent position update messages.

The noise in the control value is due to the error of the distance calculation, as described in the previous section. This noise could be eliminated by using the same Kalman filtering as in the simulations.

This measurement noise and the one second delay between position update messages caused the speed regulation to not be as precise as in the simulations.

5. Conclusions

We presented an algorithm that is able (1) to match the velocity of the vehicle to a followed vehicle by utilizing wireless communication, (2) to adapt to the changes in the velocity of the followed vehicle while (3) maintaining a safe following distance.

The proposed cruise control system can be used if the vehicles have the necessary devices for communication, position determination and calculations. Therefore, it could be introduced in truck convoys of transportation companies where only the driver of the first vehicle needs to pay increased attention; the following drivers could let the algorithm to follow the preceding vehicle.

Authors

BALÁZS MEZNY received his M. Sc. Degree in 2008 from the Budapest University of Technology and Economics. He started his Ph. D. studies in 2008 with a scholarship from the Bay Zoltán Foundation for Applied Research, Institute for Applied Telecommunication Technologies (IKTI). He is working at IKTI with a scholarship since 2007. His research interests include intelligent transportation systems and optimization.

DR. PÉTER LABORCZI is currently employed as a senior research fellow at IKTI, the Institute for Applied Telecommunication Technologies. He received his M.Sc. degree in 1999 and his Ph.D. degree in 2002 both in Computer Science from the Budapest University of Technology and Economics. In 2002 he was invited to Arsenal Research, Vienna, Business Unit Transport Telematics, where he was a Marie Curie Fellow until 2004 in the framework of an EU research project. Dr. Laborczi is the co-author of more than 30 papers in refereed journals and conference proceedings. His research interests include intelligent transportation and infocommunication systems, graph theory, algorithms; network design, optimization and analysis.

DR. GÉZA GORDOS received his M.Sc, Ph.D and Dr. Habil. in Telecommunications from the Budapest University of Technology and Economics (BME) in 1960, 1966 and 1994, resp. He holds D.Sc. from the Hungarian Academy of Sciences. He is full professor with the BME serving as head of two sections in the Department of Telecommunications and Media Informatics since 1976. His previous employments include 3 years at various universities abroad (UK, USA) and 7 years in industry, among others as Chairman of the Board of the Hungarian Telecommunications Co. (1992-93). His research activities have been focusing on the technology and management of telecommunication systems and services as well as on speech processing. In 2004 he accepted the invitation from the Hungarian equivalent of NSF to establish and lead the Institute for Applied Telecommunication Technologies (IKTI).

References

[1] Raymond Freymann, "Connectivity and Safety", 5th European ITS Congress, Hannover, Germany, June 2005.

[2] A. Török, P. Laborczi, G. Gerháth, "Constrained Dissemination of Traffic Information in Vehicular Ad Hoc Networks" accepted for Presentation at the IEEE 68th Vehicular Technology Conference (VTC2008-Fall), Calgary, Canada, 21-24 September 2008.

[3] G. Welch, G. Bishop, "An Introduction to the Kalman Filter", Technical Report, UMI Order Number: TR95-041. University of North Carolina at Chapel Hill, 1995.

[4] W. Gellert, S. Gottwald, M. Hellwich, H. Kästner and H. Küstner, "The VNR Concise Encyclopedia of Mathematics", 2nd edition, Chapter 12, Van Nostrand Reinhold, New York, NY, 1989.

[5] Gordos Géza, Gerháth Gábor, Kardos Sándor, Laborczi Péter, Mezny Balázs, Vajda Lóránt, "Improving Intelligent Transportation System's performance with the help of MANETs", Híradástechnika, Vol. LXI., No.12, 2006, pp.29–34. (in Hungarian).

[6] P. Laborczi, A. Török, L. Vajda, S. Kardos, G. Gordos, "Vehicle-to-Vehicle Traffic Information System with Cooperative Route Guidance", in Proc. of the 13th World Congress on Intelligent Transport Systems, London, UK, 8-12 October 2006. CD-ROM, Paper no. 2237.

[7] Csák Bence, "Ambient intelligence on public roads", Híradástechnika, Vol. LXI., No.12, 2006, pp.35–39. (in Hungarian).