Abstract—This article gives an overview of the research background, the techniques and approaches of team AnnieWAY participating at the Grand Cooperative Driving Challenge 2011. It describes the composition of the team, the experimental vehicle, the algorithms and approaches developed for the GCDC, and the preliminary results.

I. TEAM COMPOSITION AND RESEARCH BACKGROUND

A. History of Team AnnieWAY

Team AnnieWAY has been founded as a spin-off of the collaborative research center “Cognitive Automobiles” in 2006 as a collaboration of the Karlsruhe Institute of Technology (KIT), the Technical University of Munich, and the University of the German Forces Munich. The overall goal of the research center was to investigate techniques for fully autonomous driving. This task included research in the areas of on-board sensors like cameras, lidar, and inertial sensors, signal processing, vision, sensor fusion, scene understanding, behavior generation, and control.

While the basic research was made in the institutes belonging to the collaborative research center, team AnnieWAY was created to integrate these components into a single hardware and software setup and to build an autonomous vehicle that is capable of autonomous driving. Participating at competitions for autonomous vehicles allowed team AnnieWAY to compare the performance of their approaches with other research groups on the same benchmarks.

The formation of team AnnieWAY was triggered by the DARPA Urban Challenge 2007. The objective of the Urban Challenge was autonomous driving in urban environments. The task was to navigate autonomously through a road network similar to an urban environment. The vehicles had to keep their lanes and follow normal traffic rules. Additionally, the vehicles had to execute special maneuvers like parking and three-point turning. Since the road network was provided as a set of polylines, navigation could be based mainly on the digital map and GPS/INS localization. On-board sensors were required to sense other traffic participants and to acquire knowledge on their maneuvers [10].

After the termination of the collaborative research center on cognitive automobiles in 2010 team AnnieWAY continues its work with a changed staff composition being hosted solely at Karlsruhe Institute of Technology. However, the goals of making vehicles moving autonomously remain the same. As a next step the team is participating at the Grand Cooperative Driving Challenge 2011 in Helmond with their experimental vehicle [12]. The preparation phase started in fall 2010 and became more intensive month by month.

B. Research Background

Team AnnieWAY is embedded into the research of the Institute of Measurement and Control at KIT which consists of research in visual scene perception and scene understanding, optical measurement techniques,
and digital signal processing with applications to advanced driver assistance systems, 3D surface reconstruction, and train localization. As example for those research interests we want to focus on some topics which are related to autonomous driving and into which members of team AnnieWAY are incorporated.

One prerequisite for video-based scene understanding are efficient algorithms for 3D reconstruction from stereo camera images. Only real-time capable approaches are appropriate for autonomous vehicles and driver assistance systems. However, stereo reconstruction must also be accurate and the resulting depth maps must be dense. Therefore, we are developing efficient stereo matching algorithms for high resolution camera images. We follow two lines of development. On the one hand, we are improving stereo matching using a combination of sparse stereo matching at unique feature points with a variational approach on small local patches to obtain dense depth maps [4, 5]. On the other hand, we are working on parallel implementations of stereo matching algorithms using multi-core CPUs and GPUs [16]. This allows us to obtain dense high resolution depth images with 25 frames per second or more which can be used for scene understanding or map-generation [11].

The image sequences of the on-board stereo cameras are used for scene understanding. While classical approaches for scene understanding for driver assistance systems are based on simple visual features like road markings our approach uses a set of non-standard image features like house facades and vehicle flow which are combined in statistical models using elaborated stochastic inference techniques [3]. This allows us to reconstruct complex urban environments while classical approaches are limited to relatively simple scenarios like highways with little variation in the scene.

Besides video cameras, lidar data interpretation plays an important role in our research activities since lidar data are very precise and allows for reliable obstacle detection and scene reconstruction. We focus on 3D lidar point clouds for which we developed algorithms for scene segmentation [14] and object tracking [13]. As well as with stereo cameras the lidar data can be used for map generation [15].

Although our research focus is on the perception of autonomous vehicles some techniques have also been developed for path and trajectory planning for autonomous vehicles and for vehicle control. This includes efficient collision checking [19] and trajectory generation based on fast lattice search [18].

The remainder of the paper is organized as follows. In section II. we describe our experimental vehicle, in section III. the general software and hardware architecture of our system. In the subsequent sections we discuss individual components of our system like the communication modules (sec. IV.), the environment representation (sec. V.), and the control strategy (sec. VI.). The final section wraps up our preliminary results.

II. EXPERIMENTAL VEHICLE

Our experimental vehicle, AnnieWAY (cf. fig. 1), is equipped with several modifications over the VW Passat base vehicle: Electronically controllable actuators for acceleration, brakes, transmission and steering have been added, each of which can be enabled individually. A CAN gateway allows sending requests to these actuators and receiving selected signals like wheel speeds and status information. It additionally implements a low-level safety disengagement of autonomous functions in case the driver needs to interfere.

Determining reasonable commands for the actuators requires cognition. Several complementary sensors are available for this task: A high definition laser scanner delivers several all around 3D point clouds per second ². Multiple cameras can be mounted in different configurations on a roof rack, e.g. to provide stereoscopic vision. A third source of environmental information is the vehicle’s stock radar, which can be used to supplement the communication-based information about other vehicles. Self localization of the ego-vehicle is realized by a combined inertial- and satellite-based navigation system ³, which can optionally be augmented by reference stations (differential GPS).

²Velodyne HDL64-E
³OXTS RT 3003
A total of three computers are installed in the vehicle’s trunk: A custom Intel Xeon-based server performs higher level tasks like managing information on other vehicles and determining a setpoint acceleration suitable for the current driving conditions. An underlying real-time database [17] is employed as middleware for inter-process communication. It serves the purpose of a virtual bus system and enables both synchronous and asynchronous communication, as well as recording and replaying of data streams. Lastly, being equipped with two six-core CPUs and a high-end GPU, this computer is able to handle many concurrent processes and computationally demanding image processing tasks. A second x86-based PC has recently been added for 802.11p-based communication. The connection to the prototype vehicle itself is made through a modular rapid prototyping system \(^4\), which can meet hard real-time requirements at critical tasks like actuator control, driver interference handling, fail-safe functionality and feedback trajectory stabilization. Especially the latter is important for the GCDC since it implements the low-level acceleration controller, see section VI.

AnnieWAY’s equipment is completed by a custom buffered power supply and relay box to supply and switch any of the aforementioned components.

### III. System Architecture

Figure 2 sketches out the components of the system and data flow in between them. Grey boxes symbolize hardware systems, while white boxes stand for software components. We will now briefly explain the purpose of the single components.

**24 Ghz Doppler radar** This radar belongs to the standard equipment of the VW Passat and is part of its standard ACC system. We use it to improve detection of the immediately leading vehicle, both in terms of precision and robustness. It is connected to the system via CAN bus.

**GPS/INS** The commercial GPS/INS-System is employed to provide precise position, velocity and acceleration of the host vehicle. The data is both transmitted via CALM and used as reference for control of the vehicle. The device is connected to the main computer via UDP/IP, and to the real time computer via CAN-bus.

\(^4\) dSPACE AutoBox
**AnnieWAY host computer** The main computer is responsible for most control- and data processing tasks. It runs a real time database, which serves the purpose of a virtual bus system and enables inter process communication.

**Real time computer** The real time computer is reserved for the time critical purpose of low level control. It is directly connected to the car actuators via CAN bus, and is interfaced to the main computer via UDP/IP.

**Car Actuators** AnnieWAY has a CAN-bus interfaces to operate steering, brakes and acceleration.

**CALM-gateway** A separate, mini-ITX-formfactor computer is installed in the car, which is connected to radio hardware. It runs the calm daemon and dispatches incoming and outgoing data packages.

**Sender and Receiver** These processes are the host computer sided interfaces to the calm daemon.

**Vehicle Manager** A software component which receives vehicle data broadcast from the other platoon members. This data is augmented with radar data. The main purpose of the Vehicle Manager is to abstract from the latency of the received data: Through extrapolation and filtering, it can provide an estimate of the platoon state at any given time.

**Map Matcher** The Map Matcher is responsible for assigning vehicles to lanes, and hence decides which vehicles can be in the same platoon as the host vehicle. Furthermore, it computes geodesic distances between vehicles, which serve as input to the controller modules.

**pose server** This process interfaces to the GPS/INS hardware, via UDP/IP.
radar server This process interfaces to the radar sensor, via CAN.

LL Controller This process receives a reference acceleration from the main computer, and stabilizes it based on readings from the GPS/INS. It talks directly to the actuators via CAN-bus.

HL Controller The high level controller determines an optimal acceleration of the host car based on the platoon state. This acceleration is passed downstream to the low level controller.

The main components of our system will be described in detail in the subsequent sections.

IV. INTER-VEHICLE COMMUNICATION

The inter vehicle communication relies on a hardware and a software layer which will be discussed in order.

A. Hardware & Drivers

The communication hardware is based upon the 802.11p standard, an amendment to the popular wireless LAN standard 802.11[7] that is widely used in consumer devices. Besides defining the transmission frequencies, gains and ranges, the standard also specifies the basic addressing of devices using the MAC layer that is also used for wired Ethernet networks. The 802.11p standard broadcasts in the ITS band of 5.85-5.925 GHz and was specially derived for car-to-car communication. A good overview over the standard is given by [9].

Team AnnieWAY uses two different cards: the main vehicle and the first testing car are equipped with a Mikrotik RH52 card, while the second testing car is equipped with a UNEX DCMA-86P2 card. The first two cars also use the same antennas, an ECP12-5800, while the second test car has been equipped with a DM2-5500S dome antenna. The setup of the first two cars allows for a stable communication up to roughly 800 m if an unimpeded line of sight between the cars is maintained. The ping times are between 1 and 50 ms depending on surroundings, weather and distance. The setup for the third car only allows for 250-300 m communication range under the same conditions. The ping times are comparable.

All wireless LAN cards used are similar to comparable 802.11a cards. Still, kernel drivers needed to be written in order to access all the required features. We based our implementation on current Atheros 5k drivers from the Linux kernel (ath5k) and on patches from older Atheros drivers provided by TNO.

B. Software

1. Protocols

The GCDC is not using the IP protocol for communicating, instead the ISO Communications Access for Land Mobiles (CALM) protocol[8] was chosen. It defaults on MAC multi- and/or broadcasting packages, and only offers a limited addressing scheme for individual sending and receiving. It also does not implement routing ideas, instead relying on important messages being passed on by higher level protocols. CALM is not natively supported by Linux, but can be implemented in user space using RAW sockets.

The CALM protocol is very complex and offers a rich feature set and therefore high implementation costs. For the GCDC, a small wrapper program called the calmd was provided by TNO that essentially translates from incoming broadcasted CALM messages to a TCP connection and vice-versa. We based our own calmd on this version, but significantly improved upon the feature set and stability. We also added 64 bit compatibility. The calmd is running on the ITX CALM gateway computer which communicates via TCP/IP over a wired connection with the AnnieWAY host computer.
2. Receiver & Sender

The receiver and sender program are running on the AnnieWAY host computer. The receiver is handling all packets that are passed over from the CALM gateway via TCP/IP, tries to unpack the GCDC payload inside the packages and writes the data into the real time database. If no GCDC payload is found, the packet is discarded. As the CALM protocol does not offer error correction or checksumming, the receiver also implements a number of heuristics that reduce the risk of corrupt packages reaching the database.

The sender is observing the database for changes, encodes corresponding GCDC packets, and sends them via the calmd on the relay box. As all sources on the host computer are trusted, this software is significantly less conservative in its error checking compared to the receiver.

3. Auxiliary Software

The combination of the new 802.11p standard, the new CALM protocol and the new cards with custom drivers proved to be unstable at first. As the communication is crucial for the GCDC, a lot of effort was put into making it as stable as possible. During the implementation, bug tracking and network hardening steps, a number of tools proved useful. Their benefit and design intentions are discussed in the next paragraphs.

**CALM Sender and Receiver** A pair of scripts were implemented that checked the number of lost or corrupted packages send via the CALM protocol over wireless LAN. This information is vital to estimate the probability of receiving wrong information in the GCDC. As the CALM protocol does no error detection or correction, data that was received partly scrambled is directly passed on. These scripts also proved useful to detect buffer over- and underruns in the kernel driver and the user land libraries.

**CALM Roundtrip Sender & Receiver** The roundtrip receiver is an echo server for the CALM protocol: it immediately rebroadcasts everything it receives. The roundtrip sender is sending packages with a fixed content and a defined delay between packages. It also listens for the echo replies and measures the roundtrip time for each package. Usually, networks are designed to value bandwidth over latency, e.g. by collecting many small send requests and combining them into one Ethernet frame. But for the GCDC low latency is more important than bandwidth. These scripts helped profiling and optimizing the roundtrip time. We achieved to reduce it by an order of magnitude.

**CALM Fuzzer** To maximize stability and security of network applications, all data received from the outside must be considered unsafe, potentially broken, or even maliciously crafted to exploit a vulnerability in the receiving system. To stress test our communication framework, a CALM network fuzzer was written. It floods the network with either completely random data or slightly mutated GCDC payload packages with a very high frequency. This helped testing and improving the stability of the network system tremendously. It revealed for example a number of critical bugs like crashes and infinite loops in the CALM software stack.

Note that further work is needed to make car-to-car communication reliable and secure for every day use: the CALM protocol offers no encryption or source verification and standard network attacks like man-in-the-middle are trivial to perform and potentially lethal when wrong data is relied upon by controller strategies. The current state is also far from prefect for the GCDC: a maliciously crafted broadcast package can put all calmd that were not modified to work with garbage input into an infinite loop or a crash. Sabotaging the GCDC is therefore not only possible, but trivial.
V. Environment Representation

The GCDC takes place on a normal highway with additional infrastructure, namely traffic lights and road side units. Hence, the environment can be split into a static and dynamic part. The static environment comprises the road with its lanes, the traffic lights, and the road side units. The dynamic environment corresponds to all vehicles.

A. Vehicles

All vehicles of the GCDC, including the GCDC leading vehicle, share the same dynamic properties. This allows to describe each vehicle at time $t$ by a simple state vector

$$\vec{x}_t = (l, w, \phi, \lambda, \psi, v, \dot{\psi}, a)^T$$

with $l$ and $w$ denoting the length and the width of the vehicle respectively, $\phi, \lambda$ the GPS position (latitude, longitude), $\psi$ the heading, $v$ the velocity in direction of heading, and $a$ the acceleration in direction of heading.$^5$

Frequently, the state of other vehicles is estimated by sensors which return noisy measurements of only some of the entries of the state vector. To allow smoothed estimation of the whole state vector nevertheless, Bayesian filtering [2] is commonly applied. In the GCDC, each vehicle broadcasts its own state vector to the full extent. With the assumption that broadcasted information is already filtered and always correct, we directly store this information without applying any filtering method.

As our control strategy (see section VI.) implements a model-predictive controller, we not only have to know about the current state of other vehicles but we also have to be able to predict their state into the future. We achieve this by employing a non-linear kinematic model that is based on the assumption of constant yaw-rate and acceleration. This corresponds to the movement on a circle. To achieve a smoother behavior of the controller in the velocity limits (0 and 80 km/h), prediction is cropped by these velocity-limits.

B. Maps and Matching

The GCDC competition involves platooning in scenarios with multiple adjacent lanes. In order to join a platoon of other vehicles, the correct assignment of vehicles to lanes is important. Further, the controller needs to be precisely informed about the distance to neighboring cars on the ego-vehicle’s lane. We handle this by recording a map of the environment a priori, containing static elements of the scene, e.g., lanes, traffic lights and road side units.

To this end, our algorithm subdivides a recorded GPS track into piecewise linear segments of length 0.5 meters and adds further lanes at a given offset, if required. The vertices of these trajectories are stored in a 2-dimensional kd-tree [1], which is an efficient search structure under Minkowski metrics. In our case, we compute exact nearest neighbors using the library ANN$^6$ in conjunction with the $l_2$-norm. The kd-tree structure reduces nearest neighbor search complexity from $O(n)$ for the naive algorithm to $O(\sqrt{n})$. Time for constructing the tree, $O(n \log n)$, can be neglected, since this needs to be done only once, namely when the map is loaded from disc. An example of a kd-tree space decomposition is illustrated in Fig. 3(a) for one of our testing grounds.

In order to compute distances to all other vehicles on the ego lane, we first assign all vehicles to their closest lane by retrieving the nearest neighbor GPS track vertex. The geodesic distance between two vehicles is readily obtained by summing the segment lengths falling in between those vertices, see Fig. 3(b). For efficiency, we precompute geodesic distances of all vertices with respect to the beginning of the respective lane. We also interpolate distances in between vertices using projections of the vehicle centers onto the line segments, i.e., ‘foot points’.

---

5The length and width of a vehicle could also be stored separately, as it will not change over time

6http://www.cs.umd.edu/~mount/ANN/
Figure 3: Our maps are stored in kd-trees, as shown in (a) for one of our testing grounds. This allows for fast map matching and evaluation of geodesic distances between vehicles on the same lane (b).

VI. CONTROL

For the GCDC, several distinct control related tasks can be identified:

- Low level control stabilizes a reference acceleration.
- A follow controller stabilizes the desired safety distance to a single leading car.
- A platooning strategy stabilizes a platoon of multiple cars.

After a quick recapitulation of the precise requirements which GCDC rules impose onto the control strategy, we will deal with each of these tasks in a separate section.

A. Problem Definition and Formalization

Let the platoon consist of $N$ vehicles (only vehicles which are in front of the host vehicle are considered relevant to the platoon). The state of the $i$th car in the platoon is described by the vector $x_i(t) = (x_i(t), \dot{x}_i(t))$, which contains its position and velocity. Here, position is a scalar quantity which describes the distance traveled on a reference path, cf. section B. The system model of a single car is assumed to be a simple double integrator, i.e. it has a single input $u_i(t)$, which is its acceleration, $\ddot{x}_i(t)$. Cars are ordered by their index, i.e. $i < j \Rightarrow x_i < x_j$. The host car has index $i = 0$. Let the complete state of the platoon be the tuple $X(t) = (x_0, x_1, \ldots, x_{N-1})$.

Since the arrangement of the GCDC implies a decentralized platooning strategy, we can only control the acceleration of the host car ($i = 0$). The task of the controller is now to determine the acceleration $u(t)$ for the host vehicle, such that the following conditions hold:

- keep safety distance: $x_0(t) < x_1(t) - r + t_h \dot{x}_1(t)$. Here, $t_h$ is a constant headway time and $r$ (reserve) is a constant distance.
• keep limits for acceleration \( a: -4.5 \frac{m}{s^2} < a < 2.0 \frac{m}{s^2} \)

• keep limits for velocity \( v: 0 < v < 80 \frac{km}{h} \)

### B. Low Level Controller

Under the assumption that a low level controller is in effect, we take it that the host car can be controlled by a single input, which is its acceleration. The low level controller consists of two feedforward controllers translating setpoint accelerations to virtual pedal actuations. An integral anti-windup feedback controller compensates for disturbances from wind, slope etc.

### C. Follow Controller

The follow controller will determine an optimal acceleration for the 0th vehicle in the platoon (the host vehicle), based on its current state \((x^0_0, v^0_0) = (x_0(t_0), \dot{x}_0(t_0))\) and the trajectory \(x_{\text{lead}}(t)\) of a single leading car. Indices placed to the upper right will designate discrete time indices in this section, \(0\) indicating current time, \(t_0\). We assume that \(x_{\text{lead}}(t)\) is given. In practice, we generate it under the assumption that the lead vehicle will drive at constant acceleration, except when velocity limits must be respected. Note that the current acceleration of the vehicles in the platoon is known, since it is part of the communication protocol. The control law which we will derive hence has a single output, acceleration \(a\), and receives the current position and velocity of the host car, and current position, velocity and acceleration of the leading car as inputs. For reasons which will become clear in the next section, the control law is furthermore parametrized with a specific headway time, \(t_h\) and safety reserve, \(\bar{r}\). We will designate it as the function \(k\):

\[
a = k(x^0_0, v^0_0, x^0_{\text{lead}}, \dot{x}^0_{\text{lead}}, \bar{r}, t_h).
\]

To determine the optimal acceleration, we minimize the following functional

\[
J[u(t)] = \int_{t_0}^{t_0 + T} \left( w_{\text{dist}} \Delta d(t)^2 + w_{\text{acc}} u(t)^2 + w_{\text{vel}} \Delta v(t)^2 \right) dt,
\]

where \(\Delta d(t) = x_{\text{lead}}(t) - \bar{r} + t_h \dot{x}_{\text{lead}}(t) - x_0(t)\) is the error of the safety distance, \(\Delta v(t) = \dot{x}_{\text{lead}}(t) - \dot{x}_0(t)\) is the velocity difference to the trailing car and \(u(t) = \ddot{x}_0(t)\) is the sought-after acceleration. The functional is evaluated up to the time horizon \(T\) (currently 10 seconds). The functional integrates a weighted sum of the square of these terms, using the weighting factors \(w_{\text{dist}}\), \(w_{\text{acc}}\) and \(w_{\text{vel}}\). The first, \(w_{\text{dist}}\)-weighted term asserts that the goal of the controller, i.e. reaching the required safety distance, is met. The second, \(w_{\text{acc}}\)-weighted one incorporates dampening, by penalizing excessive accelerations. The last, \(w_{\text{vel}}\)-weighted term can be tuned to avoid overshoot.

The functional (2) can be minimized in closed form by means of the \textit{Euler-Lagrange} equation, which leads to a system of \textit{Riccati} type equations. This, however, does not allow to account for the limits of both velocity and acceleration explicitly. We therefore discretize equation (2) by sampling \(x_0(t)\) at \(m\) equidistant time steps: \(x^j_0 = x_0(t_0 + j \Delta t), j \in\{0, \ldots m - 1\}\). Furthermore, we approximate derivatives \(\dot{x}_0\) and \(\ddot{x}_0\) at time index \(j\) by central finite differences:

\[
\dot{x}^j_0 \approx \Delta^2_x x^j_0 = \frac{x^{j+1}_0 - x^{j-1}_0}{2 \Delta t} \\
\ddot{x}^j_0 \approx \Delta^2_x x^j_0 = \frac{x^{j+1}_0 - 2x^j_0 + x^{j-1}_0}{\Delta t^2}.
\]

The functional (2) then becomes a finite sum

\[
J_d(x^0_0, x^1_0, \ldots, x^{m-1}_0) = \sum_{j=1}^{m-2} w_{\text{dist}} \Delta d_j^2 + w_{\text{acc}} u_j^2 + w_{\text{vel}} \Delta v_j^2
\]
with

\[ \Delta d_j = x_{\text{lead}}^j - \tilde{r} + \tilde{t}_h \dot{x}_{\text{lead}}^j - x_0^j \]
\[ u_j = \Delta c x_0^j \]
\[ \Delta v_j = \dot{x}_{\text{lead}}^j - \Delta c x_0^j. \]

Minimization of (3) can be treated as an ordinary extremum problem. Equation (3) is a positive definite quadratic form, and both velocity- and acceleration limits can be expressed as linear inequalities. Hence, the extremum problem is a quadratic program (QP), which can be solved for exactly, in a finite number of iterations, e.g. using Goldfarb and Idnani’s active set method [6]. The desired acceleration can now be reconstructed from the extremum point, again by using finite differencing:

\[ a = \Delta c x_0^j. \]

D. Platooning

Our basic strategy of building a controller which is capable of stabilizing a platoon can be described informally like this:

- For each vehicle in the platoon
  - Consider this vehicle as a single leading vehicle. Using the control law (1), calculate an acceleration based on the state of this vehicle, using a multiple of the safety distance required between adjacent vehicles (the safety distance is multiplied by the integer index of the leading vehicle).

- Out of all these accelerations, choose the smallest one (which is the most conservative one).

To assure stable behavior, a small loose or slack \( l \) is added to the safety distance, multiplied by \( i - 1 \), where \( i \) is the vehicle index. This assures that, in the steady state, a stable lock is established on the leader of the platoon, as can be seen in figure 4(a). This lock will only change if one vehicle deviates from its optimum position by an amount greater than \( l \), as has happened in figure 4(b). Without the slack, the lock would, in the presence of noise, change very quickly near the steady state, a behavior which could possibly induce oscillation. On the other hand, when sufficient slack is used, platoon stability follows directly from the stability of the follow controller. Note that figures 4(a) and 4(b), for the sake of clarity, convey the impression that accelerations are determined only based on the distance to vehicles. However, as has been shown in the preceding section, both accelerations and velocities of the vehicles are important as well. If, e.g., in figure 4(a) the vehicle with index \( i = 2 \) was braking very hard, while the others would move uniformly, the lock would be on the braking vehicle.

VII. Preliminary Results and Discussion

At the present stage of development all modules described in this paper are implemented and running. Test drives and component tests have been made to check the availability of communication over distances of up to 800 meters, to test the precision of the GPS/INS system, to test the follow controller, and to test the platooning strategy. Beside the main experimental vehicle VW Passat we can use two additional vehicles for our tests which are equipped with a GPS/INS system and the communication devices so that they be used as manually driven heading vehicles in a platoon while the VW Passat follows automatically driven as the third vehicle. The results of the tests are promising up to now.

However, some tests can only be made at the competition site including tests concerning the communication with the vehicles of other teams and the platooning strategy for platoons with more than three vehicles. For the time until then, we plan additional tests to increase the robustness of our system and to train our safety drivers in dangerous situations.
Figure 4: Platooning strategy
The participation of team AnnieWAY at the GCDC is one step on our way towards fully autonomous driving. Since our activities are driven by long term research goals we will integrate our experience from the GCDC into our work during the next years as we already did after the DARPA Urban Challenge. For the next years, we would like to propose an extension of the GCDC objectives to fully autonomous driving including steering control and more complex scenarios like vehicle interaction in urban environments and mixed platoons of autonomously and manually driven vehicles.

ACKNOWLEDGEMENT

Thanks to the Forschungszentrum Informatik (FZI)\(^\text{7}\) for providing their experimental vehicle for our tests.

REFERENCES


\(^{7}\)http://www.fzi.de


