Team AnnieWAY's Autonomous System for the 2007 DARPA Urban Challenge

Sören Kammel, Julius Ziegler, and Benjamin Pitzer
Institute for Measurement and Control
University of Karlsruhe
76131 Karlsruhe, Germany
e-mail: kammel@mrt.uka.de

Moritz Werling
Institute for Applied Computer Science/Automation
University of Karlsruhe
76128 Karlsruhe, Germany

Tobias Gindele, Daniel Jagzent, and Joachim Schröder
Industrial Applications of Informatics and Microsystems
University of Karlsruhe
76131 Karlsruhe, Germany

Michael Thuy
Institute for Distributed Measurement Systems
Matthias Goebl
Institute for Real-Time Computer Systems
Technical University of Munich
80290 Munich, Germany

Felix von Hundelshausen
Department of Aerospace Engineering
University of the Federal Armed Forces
85577 Neubiberg, Germany

Journal of Field Robotics 25(9), 615–639 (2008)
Published online in Wiley InterScience (www.interscience.wiley.com). • DOI: 10.1002/rob.20252

Oliver Pink
Institute for Measurement and Control
Christian Frese
Industrial Applications of Informatics and Microsystems
Christoph Stiller
Institute for Measurement and Control
University of Karlsruhe
76131 Karlsruhe, Germany

Received 29 January 2008; accepted 23 June 2008

This paper reports on AnnieWAY, an autonomous vehicle that is capable of driving through urban scenarios and that successfully entered the finals of the 2007 DARPA Urban Challenge competition. After describing the main challenges imposed and the major hardware components, we outline the underlying software structure and focus on selected algorithms. Environmental perception mainly relies on a recent laser scanner that delivers both range and reflectivity measurements. Whereas range measurements are used to provide three-dimensional scene geometry, measuring reflectivity allows for robust lane marker detection. Mission and maneuver planning is conducted using a hierarchical state machine that generates behavior in accordance with California traffic laws. We conclude with a report of the results achieved during the competition.

1. INTRODUCTION

The capability to concurrently perceive a vehicle's environment, to stabilize its motion, and to plan and conduct suitable driving maneuvers is a remarkable competence of human drivers. For the sake of vehicular comfort, efficiency, and safety, research groups all over the world have worked on building autonomous technical systems that can in part replicate such capability (Bertozzi, Broggi, & Fasciol 2000; Dickmanns et al., 1994; Franke et al., 2001; Nagel, Enkelmann, & Struck, 1995; Thorpe, 1990).

The 2007 DARPA Urban Challenge was a competition introduced for expediting research on this kind of system. Its finals took place on November 3, 2007, in Victorville, California. As in its predecessors, the Grand Challenges of 2004 and 2005 (DARPA, 2005; Thrun et al., 2006), the vehicles had to conduct missions fully autonomously without intervention of human team members (see Figure 1). In contrast to the earlier competitions, the Urban Challenge required operation in a mock urban scenario, including traffic made up of both competing autonomous vehicles and human-driven cars. The major challenge imposed was collision-free driving in traffic in compliance with traffic rules (e.g., right-of-way at intersections) while completing the given mission. This required passing parked cars, performing U-turns, parking, and merging into the regular flow of traffic. Finally, recovery strategies had to be demonstrated in deadlock situations or in traffic congestions that cannot be handled solely by strictly following traffic rules.

The scope of Team AnnieWAY was to extract early research results from the Cognitive Automobiles project that would allow real-time operation of the vehicle under the restricted traffic environment in the Urban Challenge. Its team members are professionals in the fields of image processing, three-dimensional (3D) perception, knowledge representation, reasoning, real-time system design, driver assistance systems, and autonomous driving. Some of the team members were on the Desert Buckeyes team of Ohio State University and Universität Karlsruhe (TH) and developed the 3D vision system for the intelligent off-road navigator (ION) that traveled successfully 29 miles (46 km) through the desert during the Grand Challenge 2005 (Hummel, Kammel, Dang, Duchow, & Stiller, 2006; Özgüner, Stiller, & Redmill, 2007).

Figure 1. AnnieWAY stopping at an intersection on track A during the NQE.

2. HARDWARE ARCHITECTURE

The basis of the AnnieWAY automobile is a VW Passat Variant (see Figure 2). The Passat was selected for its ability to be easily updated for drive-by-wire use by the manufacturer.

2.1. Computing System

AnnieWAY relies on an off-the-shelf quad-core computer offering enough processing capacity to run all required software components for perception, situation assessment, and trajectory generation. The chosen hardware architecture is optimally supported by the real-time-capable software architecture that is described in Section 3.

The main computer is augmented by an electronic control unit (ECU) for low-level control algorithms. It directly drives the vehicle's actuators. Both computer systems communicate over an Ethernet link. The drive-by-wire system as well as the car odometry are interfaced via the Controller Area Network (CAN) bus. The differential global positioning system (DGPS)/inertial navigation system (INS)

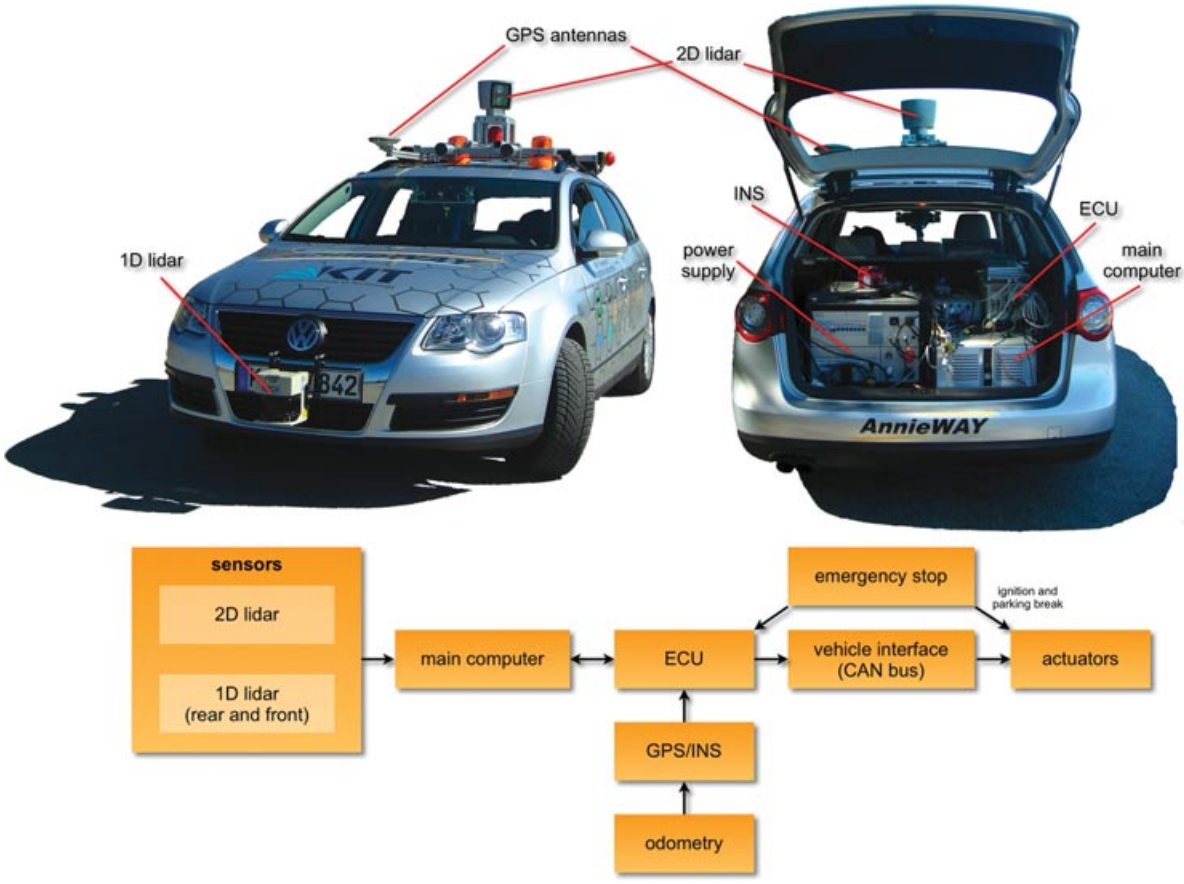


Figure 2. Architecture and hardware components of the vehicle.

allows for precise localization and connects to the main computer and to the low-level ECU.

2.2. Laser-Based Range and Intensity Sensors

Because LIDAR units produce their own light, low-light conditions have no effect on this kind of sensor. In our car we use a rotating laser scanner comprising 64 avalanche photodiodes that are oriented with constant azimuth and increasing elevation covering a 26.5-deg vertical field of view. The lasers and diodes are mounted on a spinning platform that rotates at a rate of 600 rpm. Thus, the LIDAR provides a 360-deg field of view around the vehicle, producing more than 1 million points per second at an angular resolution of 0.09 deg horizontally and a distance resolution of 5 cm with distances up to 100 m. The result is a dense, highly accurate scan representation of almost the entire scene surrounding the vehicle. For each point, the sensor measures range and reflectivity. The reflectivity map is well suited for monoscopic image analysis tasks such as lane marker detection. The inherent association of each reflectivity pixel with a range measurement alleviates information fusion of these data significantly. For parking maneuvers, the main LIDAR is supported by two two-dimensional (2D) laser scanners that cover the area directly in front of and behind the vehicle.

2.3. DGPS/INS

A precise localization is provided by a dead-reckoning system that consists of an advanced six-axis INS with an integrated real-time kinematic (RTK)/GPS receiver for position and a second GPS receiver for accurate heading measurements. Odometry is taken directly from AnnieWAY's wheel encoders. The dead-reckoning system delivers better than 0.02-m positioning accuracy under dynamic conditions using differential corrections and 0.1-deg heading accuracy using a 2-m separation between the GPS antennas.

2.4. Emergency Stop System

As the vehicle had to operate unmanned, a wireless stop system was integrated for safety reasons as required by DARPA. This E-stop system allows remote command of run, pause, or emergency stop mode. The system is connected directly to the ignition and the parking brake to assert appropriate emergency stop regardless of the state of the computer system. Run and pause modes are signaled to the low-level control computer.

3. SOFTWARE ARCHITECTURE

The core components of the vehicle are the perception of the environment, an interpretation of the situation in order to select the appropriate behavior, a path planning component and an interface to the vehicle control. Figure 3 depicts a block diagram of the information flow in the autonomous system. Spatial information from the sensors is combined to a static 2D map of the environment. Moving objects are treated differently. Such dynamic objects also include traffic participants that are able to move but have zero velocity at the moment. To detect moving objects, the spatial measurements of the LIDAR sensor are clustered and tracked with a multihypothesis approach. To detect possibly moving objects, a simple form of reasoning is used: If an object has the size of a car and is located on a detected lane, it is considered to be probably moving. Lane markings are detected in the reflectance data of the main LIDAR. Together with the road network definition file (RNDF), the absolute position obtained from the dead-reckoning system, and the mission data file (MDF), this information serves as input for the situation assessment and the subsequent behavior generation. Most of the time, the behavior will result in a drivable trajectory. If a road is blocked or the car has to be parked, modules for special maneuvers, such as the parking zone navigation module, are activated.

All data exchange between processes is done via a central communication framework, the real-time database for cognitive automobiles KogMo-RTDB (Goebl & Färber, 2007b). All data within the RTDB are represented as time-stamped objects. The centralized data storage gives the opportunity to easily log and replay all or selected objects. For performance reasons the database is completely memory based. It is capable of distributing even large data objects, such as LIDAR raw sensor data, to several processes and at the same time relay vehicle control commands at a rate of 1 kHz between a vehicle control process and the ECU (Goebl & Färber, 2007a).

4. PERCEPTION: ENVIRONMENTAL MAPPING

Accurate and robust detection of obstacles at a sufficient range is an essential prerequisite to avoid

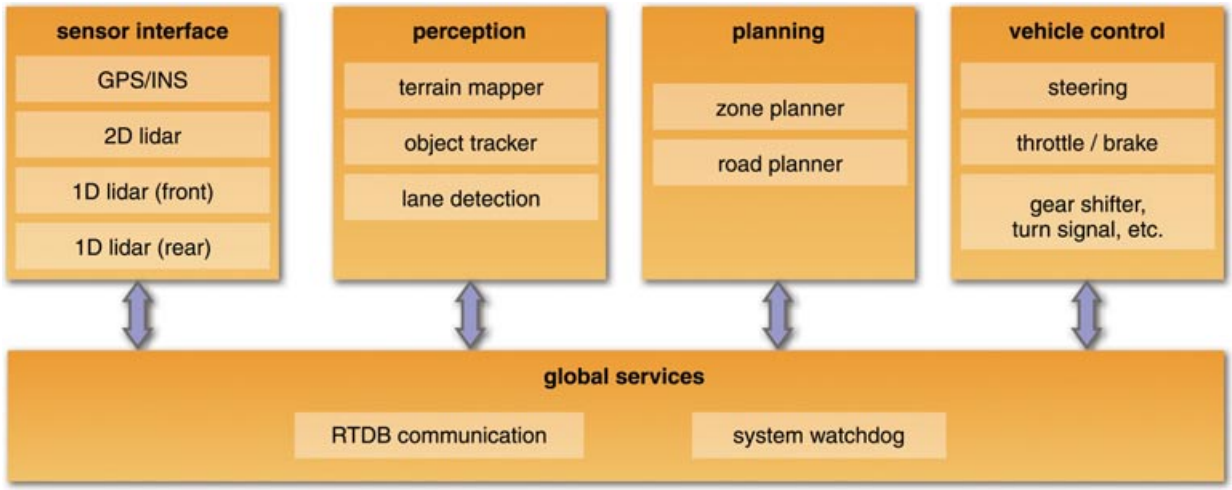


Figure 3. Overview of the software architecture and the information flow.

obstacles on the road and in unstructured environments such as parking lots. The basic idea is to maintain an evenly spaced 2D grid structure g, where each cell g_i represents a random variable. Each random variable is binary and corresponds to the occupancy it covers. Therefore, in the literature this approach is also called *occupancy grid mapping* (Thrun, 2002, 2003), which has the goal to calculate the posterior over maps $p(g|\mathbf{z}, \mathbf{x})$, where $\mathbf{z}$ is the set of all measurements and $\mathbf{x}$ is the path of the vehicle defined through a sequence of poses. An example of a resulting evidence map is depicted in Figure 4.

AnnieWAY uses a grid that is always centered at the vehicle position but aligned with a global coordinate system. The grid is shifted at each time step to account for the new vehicle position. This restricts the size of the map to an area around the vehicle while the cells are bound to an absolute position. The size of each grid cell is 15×15 cm. Figure 5 shows an example of our mapping algorithm. The grid is generated mainly from multilayer, high-resolution LIDAR data. Algorithms for the integration of low-resolution LIDAR data can be found in Biber and Strasser (2006), Bosse et al. (2003), and Thrun (2002, 2003).

Integrating the data of the laser scanners into an environmental map consists of three steps. In the first step the range measurements $z_{l \epsilon L}$ of one revolution L are projected into a global coordinate system under consideration of the vehicle's motion x_l. In the second step, different measures are extracted from the data for each cell g_i. Two straightforward measures are the number of measurements n_i and the number of different laser beams b_i. The most important measure we use is the elevation difference

$$e_i(g_i, z_l) = \max_{l \epsilon L} h(g_i, z_l) - \min_{l \epsilon L} h(g_i, z_l), \qquad (1)$$

where h is the vertical component of each measurement.

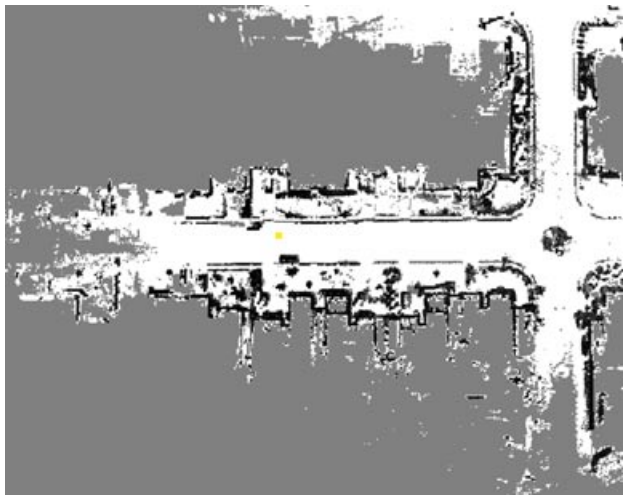
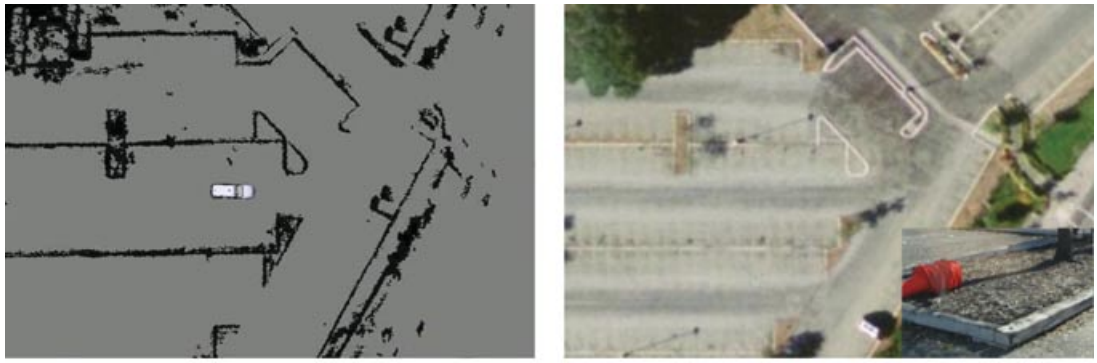


Figure 4. Example for the evidence mapping of 3D LIDAR data onto a 2D grid. Darker spots correspond to high evidence for an obstacle, and white cells correspond to drivable area. Unknown cells are gray.



(a) Map generated from a parking lot

(b) Aerial imagery of the parking lot with a detail photo of the curb in the lower-right-hand corner

Figure 5. Example for a generated evidence map and an aerial image of the corresponding region.

In the third step, we compute the evidence for each measure by using an inverse sensor model. For example, the inverse sensor model for the elevation difference returns l_{occ} if e_i exceeds a certain threshold (e.g., 15 cm) and l_{free} otherwise. The inverse models for n_i and b_i are slightly more complex because they are learned by a supervised learning algorithm. The result of the learning procedure is a forward model that accepts g_i and n_i or b_i as parameters and returns the appropriate evidence.

Finally, we can compute the combined occupancy evidence $o_{i,t}$ as a weighted sum of the three partial evidences:

$$o_{i,t} = o_{i,t-1} + \alpha_1 \cdot n_i + \alpha_2 \cdot b_i + \alpha_3 \cdot e_i, \tag{2}$$

and the estimated occupancy for a single cell

$$p(g_i|\mathbf{z}, \mathbf{x}) = 1 - \frac{1}{1 + \exp o_i}. \tag{3}$$

As already mentioned, AnnieWAY is equipped with different sensors, and ideally one wants to integrate information from all sensors into a single map. A naive solution is to update the map for each sensor separately, which neglects the different characteristics of each sensor, that is, field of view, maximal range, and noise characteristic. To ensure safe driving we use the most pessimistic approach to fuse sensor data: We compute the maximum of all estimated occupancies, where K is the number of sensors:

$$p(g_i) = \max_{k \epsilon K} p\left(g_i^k\right). \tag{4}$$

If any sensor detects a cell as occupied it will be occupied in the combined map.

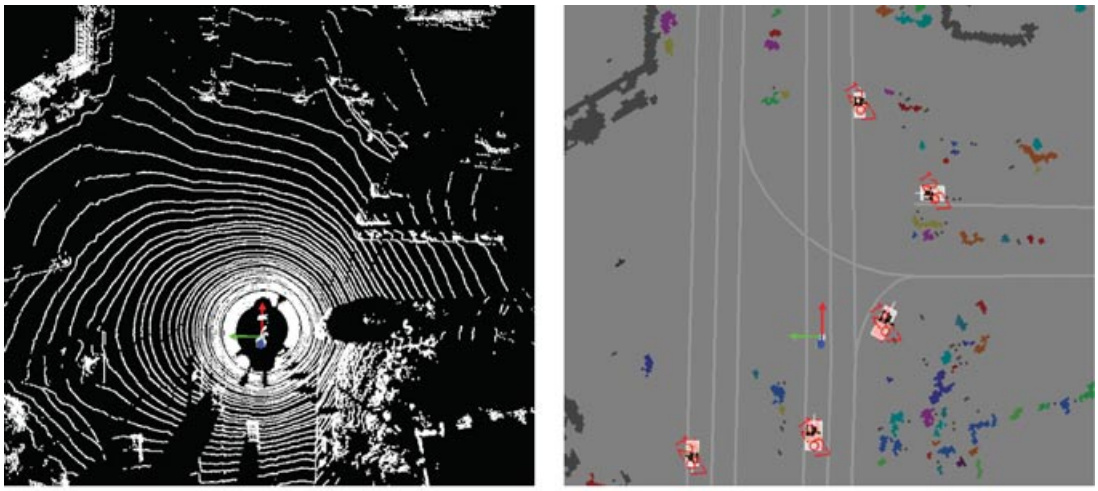
The standard occupancy grid mapping algorithm suffers from a major drawback: It is suitable only for static environments. Driving environments are typically highly dynamic, and the result is very poor without modifications. Moving objects create virtual obstacles with high evidence while moving. To overcome this problem we introduce a temporal evidence decay. The evidence is reduced at each time step by a factor ϵ_t for cells that are not updated. The intuition is that the uncertainty increases for cells not augmented by any sensor. Equation (2) turns now into

$$o_{i,t} = \text{argmax}(0, o_{i,t-1} + \alpha_1 \cdot n_i + \alpha_2 \cdot b_i + \alpha_3 \cdot e_i - \epsilon_t), \tag{5}$$

where the argmax operator enforces positive evidence.

5. TRACKING OF DYNAMIC OBJECTS

Driving in urban environments requires the capture and estimation of the dynamics of other traffic participants in real time. AnnieWAY uses a processing pipeline that takes raw sensor data (from different lasers) and generates a list of dynamic obstacles,



(a) High-resolution LIDAR data (b) Tracked vehicles on segmented grid map

Figure 6. Tracking of dynamic objects with occupancy grid map and linear Kalman filter.

along with their estimated locations, sizes, and relative velocities. This pipeline consists of a number of parts, including the following:

1. **Data preprocessing:** Removing irrelevant readings: noise, ground readings, readings from obstacles outside the road, etc.
2. **Obstacle detection:** Creating a list of obstacle raw readings; includes segmentation for laser.
3. **Obstacle tracking:** Corresponding obstacles time step with those of another time step in order to determine their headings, relative velocities, etc.
4. **Obstacle postprocessing and publishing.**

The data preprocessing step used for tracking was discussed in Section 4.1. The result of this part is a grid map with occupancy probabilities attached to each cell. All the sensors' information has been condensed within this grid.

The first stage of dynamic object tracking is the object detection, which is—in the sense of a statistical approach—equivalent to the identification of object hypotheses. AnnieWAY uses an occupancy grid map that has been segmented using a connected components approach. Therefore, we treat each grid cell as a node in a graph G. Two points are connected if and only if the distance between them is within a threshold d (e.g., 0.5 m). We then find all the connected components in the graph and assign the same label to those cells. To reduce noise, we discard any connected component with fewer than a minimum number of cells. Owing to the uniform angle resolution of the scanners, the number of cells an object consists of depends on its distance. The closer an object is located to the scanner, the more laser rays will hit the object.

The connected components are analyzed in a second step for their probability of being traffic participants. Several heuristics are used based on their shape and location relative to the road network. Only "good" candidates are augmented in the following tracking step. Figure 6(b) displays the resulting objects after postprocessing.

With this procedure, not all captured and tracked objects are relevant to be published to other modules. This is due to noisy observations, occlusion, dynamic objects leaving our sensors' fields of view, etc. All these effects lead to unlikely object hypotheses, but nevertheless they are internally tracked. To decide when to publish relevant obstacles, we define a notion of confidence that works similarly to log-likelihood updates in an occupancy grid map as mentioned earlier. If an obstacle is observed, we increment its confidence; in case it goes unobserved in our field of view, we decrement it. Thus defined, the confidence allows us to set minimum thresholds for the tracking and publishing of obstacles: If the object's confidence exceeds the threshold, the obstacle is published to all other attached modules. If its confidence undercuts a certain threshold, the object is removed

from the obstacle list. Hypotheses within both thresholds are internally tracked but not published.

Tracking of dynamic objects mainly serves two purposes. First, it aids the correspondence of obstacles detected in one sensor frame at time $t = k$ with those in subsequent sensor frames at time $t = k + 1$. This can be easily achieved with distance-based methods or more sophisticated 3D fitting and registration algorithms such as iterative closest point (ICP). However, these methods do not take into account the noise and uncertainty of our sensors. The second and equally important purpose of tracking is to return estimates of another vehicle's relative velocities and headings.

AnnieWAY uses a linear Kalman filter (Kalman, 1960) to model a simplified dynamic obstacle with its appropriate state vector $[x, y, \dot{x}, \dot{y}]^T$. Obviously, this model ignores completely the underlying physical and nonlinear behavior of a car, but the frequency of sensor updates (10 Hz) means that cars move very little between them, which allows us to assume linear dynamics. Transition updates are linear with an overlaid Gaussian noise characterized by its covariance matrix **Q**:

$$\mathbf{T} = \begin{bmatrix} 1 & 0 & \Delta t & 0 \\ 0 & 1 & 0 & \Delta t \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}, \quad \mathbf{Q} = \begin{bmatrix} 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \\ 0 & 0 & \sigma^2_{q,\dot{x}} & 0 \\ 0 & 0 & 0 & \sigma^2_{q,\dot{y}} \end{bmatrix}. \quad (6)$$

Because we are extracting the obstacle's position $[x, y]^T$ from the measurement, the observation matrix **O** looks as described below. Further, we assume mutual independent Gaussian noise sources characterized by the covariance matrix **R**:

$$\mathbf{Q} = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \end{bmatrix}, \quad \mathbf{R} = \begin{bmatrix} \sigma^2_{r,x} & 0 \\ 0 & \sigma^2_{r,y} \end{bmatrix}. \quad (7)$$

After performing an observation, we do not know which detected obstacles within the measurements are already tracked or whether they are new objects. Thus, we are required to solve a problem of correspondence between observations and the internally tracked dynamic obstacles. This is a nontrivial problem, requiring that we define both a measure of distance and a procedure for finding the optimal correspondence. AnnieWAY uses a maximum-likelihood matching algorithm to find the optimal assignment of observations to existing Kalman filters. This matching is a one-to-one function from filters to observations.

6. LANE MARKER DETECTION

Digital maps of a road network are often not up to date or resemble the real road network only approximately. Therefore, a local offset between the digital and the real road network may exist. The detection of lane markings helps to minimize this offset. An accurate and continuous detection of lane markings even enables the creation of new road network maps.

In the context of this paper, lane markings can be either painted markings or curbs. Painted lane markings are detected within the intensity readings of the LIDAR, whereas curbs cause small height changes in the range data of the LIDAR. A combined intensity/range plot is depicted on the left-hand side of Figure 8. Both kind of lane markings form one-dimensional structures that can be approximated by line segments locally. In contrast to camera-based intensity images, the laser reflectivity and range data are insensitive to background light and shadows. However, the sensor samples the road very sparsely, especially at a distance. To increase the density of lane marker information, subsequent scans are registered spatially and accumulated employing absolute positioning information from the dead-reckoning system. The first step in order to obtain a dense bird-eye's-view representation of lane marker features is a classification of data points in each scan into obstacle and ground by the algorithms described in Section 4.1. Lane markings are expected to occur on the road surface (painted markings) or at its borders (curbs) only. Therefore, points of each individual laser labeled as ground are searched for large continuous chunks (chunks that do not exhibit height changes exceeding the height of curbs) representing the road. Only within those large chunks are high-intensity gradients detected. In addition, only measurements exhibiting absolute intensities larger than the median intensity of each laser scan are taken into account. Both types of features—painted markings and curbs—are mapped into a feature grid $g(x)$ similar to the evidence grid described in Section 4.1; see Figure 7 (right). Features are detected first in the single scans and mapped afterward (instead of creating a dense map first and extracting the features afterward) to minimize the effect of errors in the vehicle

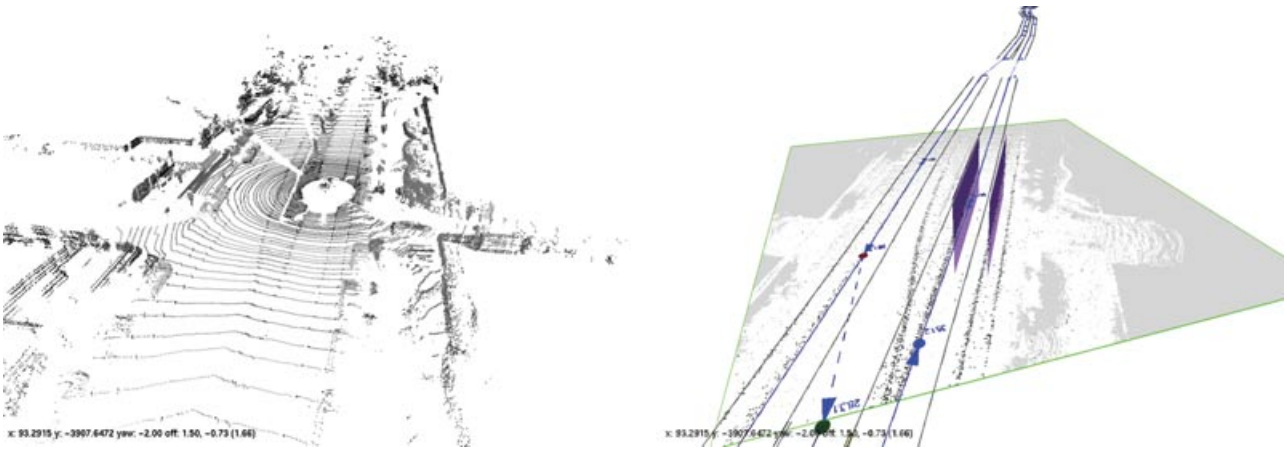


Figure 7. Combined range and intensity readings of the LIDAR (left) and lane marker map with the estimated current lane segment and an overlay of a part of the original road network map (right).

localization. A summary of the detection algorithm is shown in Figure 8.

Lane segments are detected by applying the Radon transform to the accumulated feature map data. Because the Radon transform is an algorithm operating globally on the map, it proved to be robust against occlusions, noise, and outliers. Compared to the Hough transform, the Radon transform exhibits the advantage of a calculation time independent of the numbers of lane markings and the capability to handle gray-scale images efficiently and without thresholding. For a real-time calculation in the car, an implementation exploiting the central-slice theorem was used (Bracewell, 1990). The position and direction of lane boundaries can be calculated by locating their corresponding maxima in the Radon plane. Because we observed a systematic error of RNDF data in some areas, it appeared to be sensible to determine a correcting offset from the detected lane markings. To accomplish this, the lane markings specified in the RNDF are first projected into the Radon plane. Assuming that the offset of the road map data does not exceed one lane width, the deviation is obtained in a second step from the distances to the maxima in the Radon plane closest to the predicted positions. Assuming further that predicted and estimated lane boundaries are close to parallel, the vertical distance is sufficient to determine the offset.

7. REACTIVE LAYER

Our system integrates a reactive layer that allows AnnieWAY to modify a planned trajectory based on GPS way points. Although the obstacle tracker easily handles objects such as cars, small or extended objects such as rocks or pavement edges are more difficult to track explicitly. Hence, we integrated a reactive mechanism that gets as input a vehicle-centered occupancy grid (built from the LIDAR data) and the trajectory planned so far. The algorithm then first evaluates whether the given trajectory is clear and, only if it is not, starts a more complex evaluation of the grid that results in a modification of the initially given trajectory. This mechanism is biologically

Figure 8. Overview of the offset estimation and street topology mapping.

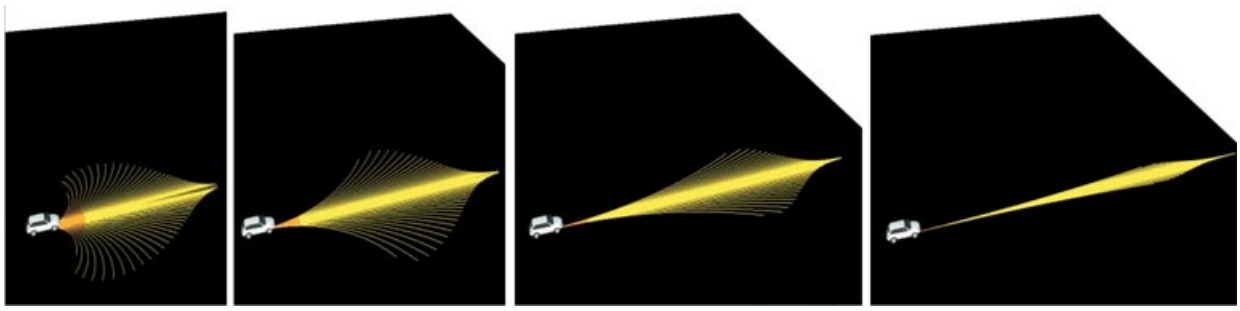


Figure 9. The reactive system uses a precomputed set of motion primitives that vary with the speed of the vehicle. As detailed in von Hundelshausen et al. (2008), those primitives are used to evaluate a vehicle-centered occupancy grid to avoid obstacles.

motivated and resembles an insect's use of its antennae to avoid obstacles. Corresponding to the antennae of an insect are precomputed trajectory primitives (we call them *tentacles*) in our system. Here, all tentacles are simple circular arcs, but depending on the speed of the vehicle, the parameters of these arcs vary such that at high speeds no dangerous actions can be taken (see Figure 9). To select the appropriate primitive, the occupancy grid is investigated in an area around and underneath that primitive. The final selection is done on the basis of four aspects:

1. Could the vehicle drive the primitive without causing damage? In particular, within a distance the vehicles needs to stop, is the ground along the tentacle clear of anything having a height above 0.1 m?
2. How smooth is the terrain under the primitive?
3. How far is the next obstacle along that primitive?
4. How well does the primitive follow the original trajectory?

By considering these aspects as detailed more precisely in von. Hundelshausen, Himmelsbach, Mueller, and Wuensche (2008), the vehicle follows the given trajectory if possible but avoids obstacles if not. To coordinate this reactive layer with the obstacle tracker, tentacles were evaluated up to only the first explicitly tracked obstacle. In this way, only unexpected obstacles were avoided.

As detailed in von Hundelshausen et al. (2008), the overall reactive mechanism was tested excessively by intentionally defining bad GPS trajectories, e.g., those having a large offset to the real road (passing through the front gardens of neighboring houses), passing through a traffic circle (instead of leading around it), abbreviating a crossing through a complete house, and completing other tests including moving vehicles. At the final of the Urban Challenge this mechanism was important at narrow passages.

8. PLANNING

The major challenge imposed by the competition was collision-free driving in traffic in compliance with traffic rules, e.g., right-of-way at intersections. It included special maneuvers, such as overtaking, U-turns, parking, and merging into the regular flow of traffic while completing the given missions. To accomplish this, the robot must be capable of analyzing the situation, assessing developments, and choosing the appropriate behavior and executing it in a controlled way. AnnieWAY uses a planning module organized in three layers to address these problems:

1. **Mission planning** computes a strategic plan to accomplish the mission.
2. **Maneuver planning** applies California traffic rules and plans actual driving maneuvers (e.g., turns, intersection, passing) and generates a corresponding path.
3. **Collision avoidance** tests whether the planned path is collision free, taking into account the obstacle map acquired from the perception module. If a collision is probable it chooses an alternative path.

In a first preprocessing step, all elements of the RNDF (lanes, checkpoints, exits, etc.) are converted to a graph-based, geometrical representation. RNDF

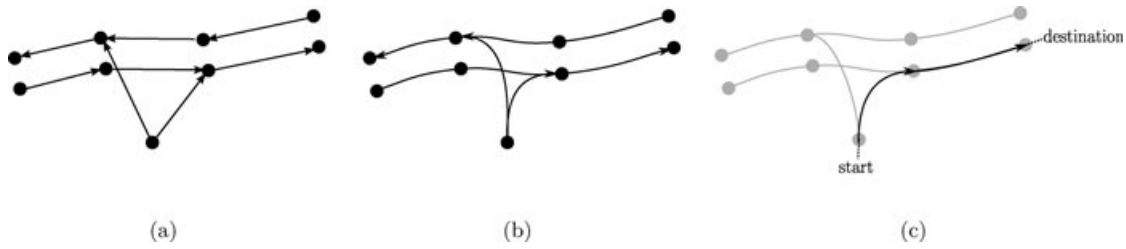


Figure 10. Preprocessing of RNDF data. (a) Geometric graph derived directly from RNDF. (b) Smoothing with splines. (c) Complete path for mission. It has continuous curvature.

way points form the vertices of the graph; lanes and exits are represented by graph edges [Figure 10(a)]. Edges yield a geometric representation by smoothing them by spline interpolation [Figure 10(b)]. Information such as distances, lane boundaries, and speed limits annotates the graph edges. These annotations can be updated dynamically to incorporate results from the perception module (e.g., road blockages).

Dynamic objects recognized by the perception module are matched to the most probable edge of the geometrical graph representation, based on their position and orientation. This allows for attributing a role to every object, e.g., identification of a leading vehicle, or semantically localizing an object within an intersection scenario.

Mission planning is the most abstract form of planning used by AnnieWAY. It finds the optimal route from one checkpoint to another using an optimal graph search algorithm operating on the geometric graph representation of the road network. The criterion that is minimized by the search process is travel time. The search process is repeated for every pair of subsequent checkpoints in the MDF. In this way the mission planner finds the optimal route traversing all mission checkpoints. It is a piecewise-defined spline curve, as shown in Figure 10(c). Generally the mission planner runs only once while loading the mission file and whenever AnnieWAY has to diverge from the planned route for situation-dependent reasons (e.g., blocked roads). The route is passed to downstream maneuver planning.

The high-level plan and the AnnieWAY's current position are used by maneuver planning to compute actual driving maneuvers. The maneuver planner is implemented as a hierarchical state machine (HSM), with every state representing a driving behavior. The key aspect of a HSM is to design and group the states in a way that a substate is a specialization of its parent state, and only extensions to the more general behavior of the parent state have to be modeled explicitly. Thereby, the functional redundancy of the states and the amount of transitions are reduced, and so it is easier to capture the complex reactional behavior of a system. Figure 11 shows the UML state chart of the machine's main level, with important substates annotated as well.

Every behavior the car is capable of is modeled as a state organized within a state hierarchy. The state Drive comprises all regular driving maneuvers on normal roads. It has several substates that cover different situations, such as following the course of a lane (DriveOnLane), making a k-turn (DriveKTurn), or changing the lane (LaneChange). All behavior at intersections is handled by the Intersection state. It comprises some specialized substates for different types of intersections. Some more insight on the real functionality and architecture of the state machine is given in Section 9, where handling of moving traffic in an intersection scenario is explained in detail. The navigation in unstructured environments and parking maneuvers is controlled by the state Zone and its substates. These states control invocation of the navigation module described in Section 10. In some situations it becomes necessary for the robot to replan its route, e.g., when the road ahead is blocked. This is triggered by the state Replan, which reactivates the mission planning module. Most states implement a recovery state that is activated whenever the car makes no progress at all for a certain amount of time. If all situation-dependent recovery handling fails, a global recovery state is invoked to navigate back on track using the navigation module.

When all situation assessment has taken place and all state transitions are made, the reached state

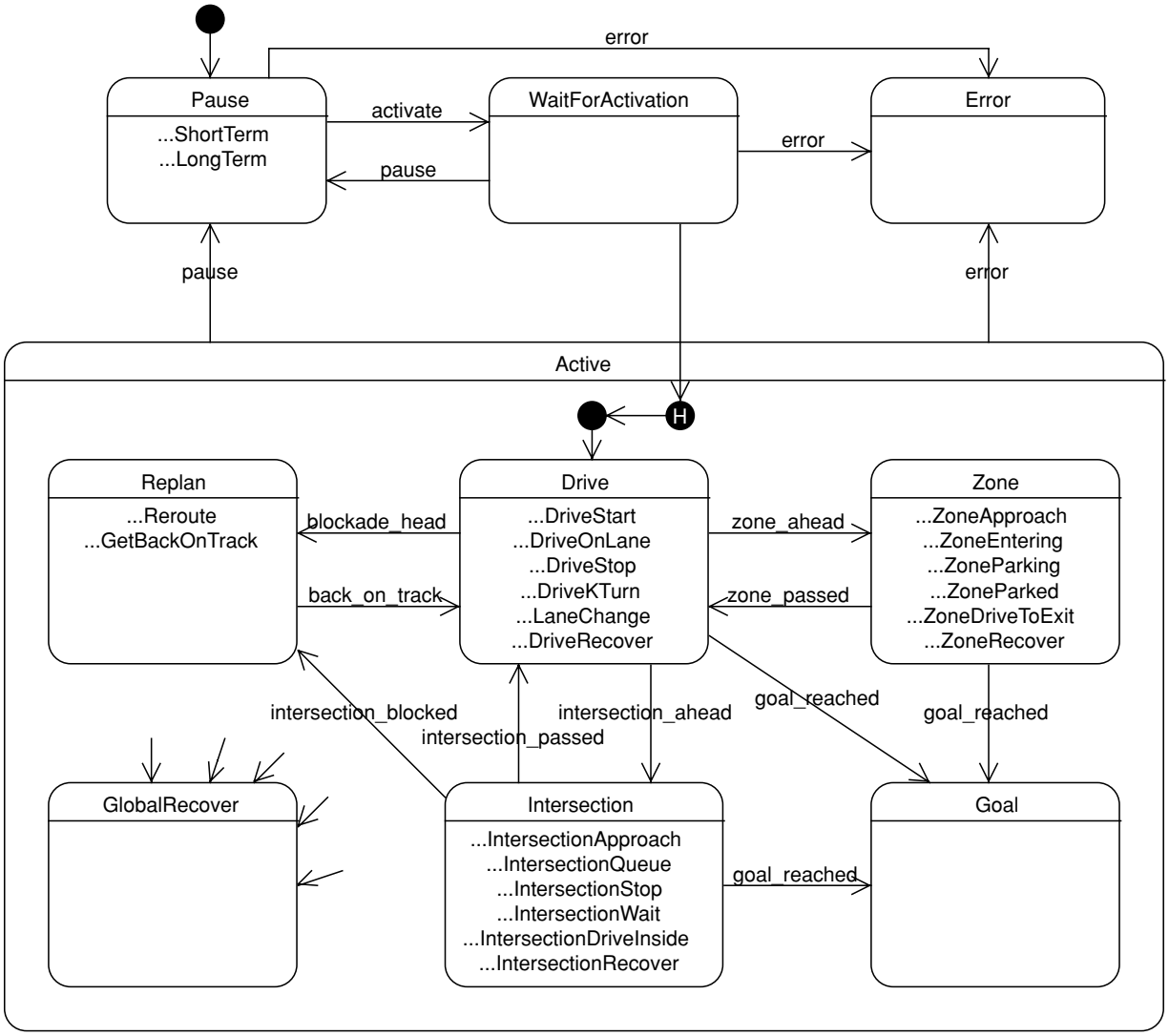


Figure 11. Overview of the HSM used to model traffic situations and behavior.

generates a path stub that is input to the closed-loop control module (Section 11). It reaches approximately 30 m ahead and consists of densely sampled way points combined with heading and curvature information. In the most common case, when the car is driving on roads stored within the graph representation, the trajectory is generated in a straightforward way by sampling the graph edges ahead. These points are smoothed by a spline approximation to generate a continuous-curvature path. In areas that lack road geometry description and whenever sensible localization within the road network graph is not possible, the free navigation module in Section 10 is used to plan a collision-free path to a given target configuration.

Paths generated by the state machine may be overwritten by the low-level avoidance system described in Section 7.

9. MOVING TRAFFIC

This section describes an algorithm that reduces dynamic maneuvers, such as merging into moving traffic and crossing intersections with oncoming traffic, to static maneuvers, such as simple turns. Unfortunately, the actual behavior of the other traffic

participants cannot be exactly predicted. Therefore certain assumptions, simplifications, and conservative estimates have to be made in an appropriate way, such that the unmanned vehicle operates safely as well as effectively.

9.1. Problem Abstraction and Simplifications

In the following, it is assumed that (1) the other traffic participants with the right-of-way neither slow down nor speed up, (2) the other traffic participants stay in the middle of the road, (3) AnnieWAY's longitudinal controller accelerates at a known constant rate until the desired maneuver velocity is met, and (4) all traffic participants' velocities and positions are known.

Assumptions 1 and 2 have to be made because the actual behavior of the other vehicles (B_i) cannot be precisely predicted. Therefore it is assumed that the considered vehicles travel at a constant velocity in the center of the priority road. Introducing $t_{\overline{BP}}$ as the time needed for traveling a distance $d_{\overline{BP}}$ in the road center and v_B as the other vehicle's constant velocity leads to

$$t_{\overline{BP}} = \frac{d_{\overline{BP}}}{v_B}. \tag{8}$$

Assumption 3 is based on the longitudinal control strategy, which is described in Section 11. The resulting drive-off characteristic $v(t)$ from a start velocity v_0 to a new desired velocity v_d can be seen on the left in Figure 12 as a dashed line along with the approximation $\hat{v}(t)$ as a solid line. Here t_{sw} denotes the time when the approximated velocity $\hat{v}(t)$ reaches v_d. It can be calculated by

$$t_{sw} = \frac{v_d - v_0}{a_{\text{sat}}}. \tag{9}$$

An integration of $\hat{v}(t)$ over time (see Figure 12) yields

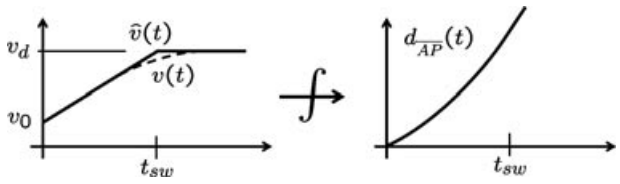


Figure 12. Actual and approximated drive-off characteristics.

the traveled distance of AnnieWAY (A)

$$d_{\overline{AP}}(t) = \begin{cases} v_0 t + \frac{a_{\text{sat}}}{2} t^2, & t \le t_{sw} \\ v_0 t_{sw} + \frac{a_{\text{sat}}}{2} t_{sw}^2 + v_d (t - t_{sw}), & t > t_{sw} \end{cases}. \tag{10}$$

Solving Eq. (10) for t with

$$t_a = \frac{d_{\overline{AP}} - v_0 t_{sw} - (a_{\text{sat}}/2) t^2 sw}{v_0 + a_{\text{sat}} t_{sw}} + t_{sw} \tag{11}$$

yields

$$t_{\overline{AP}} = \begin{cases} t_a, & t_a > t_{sw} \\ \frac{1}{a_{\text{sat}}} \left(-v_0 + \sqrt{v_0^2 + 2 a_{\text{sat}} d_{\overline{AP}}} \right), & t_a \le t_{sw} \end{cases}, \tag{12}$$

whereas the ambiguity of the solution was resolved.

Figure 13 illustrates the transfer of different traffic scenarios to the equivalent graphs, whose generic graph can be found to the left in Figure 14 along with the four relevant quantities to be measured, the current distances $d_A(t)$ and $d_B(t)$ to MP, and the current velocities $v_A(t)$ and $v_B(t)$ (assumption 4). As can be seen, traffic participants are all assumed to be point masses. Based on the previous equations and graphs, the movement of the vehicles can be predicted and used for collision detection in the next section.

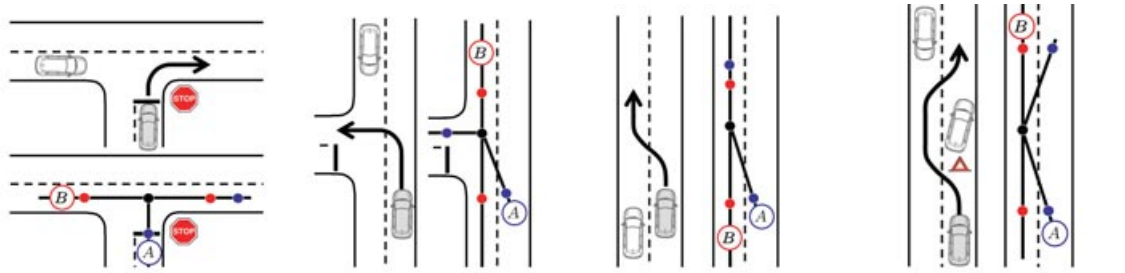
9.2. Spatial and Temporal Verification

On the one hand, at low speed it has to be guaranteed that the autonomous vehicle avoids collisions by not getting too close to other traffic participants. Therefore spatial safety distances were introduced (see Figure 14, right-hand side). On the other hand, spatial safety distances are not a proper measure at higher speeds. In this case a temporal safety distance ensures certain time gaps between AnnieWAY and the other traffic participants. Because time gaps become too small referred to the ground at low speed in turn, both spatial and temporal conditions have to be fulfilled at the same time.

For the sake of simplicity, only a single vehicle is considered initially. To be the first to enter the critical area, the following two conditions have to be met:

1. At time $t_{\overline{BP}_{B1}}$, when B reaches P_{B1}, A has to be beyond P_{A2}:

$$\text{free}_{\text{spat},AB} = \left(d_{\overline{AP}}(t_{\overline{BP}_{B1}}) > d_A + D_{A2} \right). \tag{13}$$



(a) Right turn with stopping (b) Left turn without stopping (c) Lane change maneuver (d) Double lane change maneuver with oncoming traffic

Figure 13. Different moving traffic scenarios.

2. After A has passed MP, a given time span ΔT_{AB} has to elapse before B reaches MP:

$$\text{free}_{\text{temp},AB} = (t_{\overline{BMP}} > t_{\overline{AMP}} + \Delta T_{AB}). \quad (14)$$

To be the second to enter the critical area, the following two conditions have to be met:

1. At time $t_{\overline{BP}_{B2}}$, when B reaches P_{B2}, A may not have passed P_{A1} yet:

$$\text{free}_{\text{spat},BA} = \left(d_{\overline{AP}}\left(t_{\overline{BP}_{B2}}\right) < d_A - D_{A1}\right). \quad (15)$$

2. After B has passed MP, a given time span ΔT_{BA} has to elapse before A reaches MP:

$$\text{free}_{\text{temp},BA} = (t_{\overline{AMP}} > t_{\overline{BMP}} + \Delta T_{BA}). \quad (16)$$

This means that if

$$\text{free} = (\text{free}_{\text{spat},AB} \wedge \text{free}_{\text{temp},AB}) \vee (\text{free}_{\text{spat},BA} \vee \text{free}_{\text{temp},BA})$$

is true, it is ensured that neither is A between P_{A1} and P_{A2} as long as B is between P_{B1} and P_{B2} nor are the time gaps in MP shorter than permitted.

The extension from a single vehicle B to n vehicles B_i is straightforward. As long as one vehicle fails the verification, A is not allowed to enter the critical zone:

$$\text{free}_{\text{tot}} = \text{free}_1 \wedge \text{free}_2 \wedge \cdots \wedge \text{free}_n. \quad (17)$$

9.3. Integration into the State Machine

The planner of Section 8 always deploys the moving traffic check (MTC) when AnnieWAY might come into conflict with other traffic participants demanding the same traffic space (conflict spaces). Contingent upon the result obtained from the MTC and the particular situation (conflict situations), state transitions are triggered and the resulting state generates the desired path and approves the free section for the longitudinal control.

To prevent frequent switching back and forth between states due to measurement noise and control

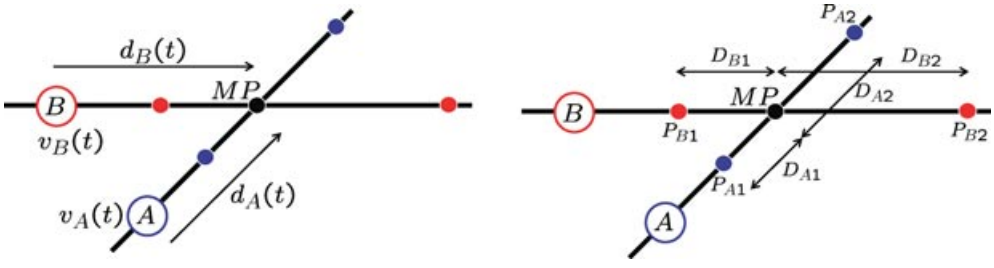


Figure 14. Measured quantities and geometric parameters of the graph.

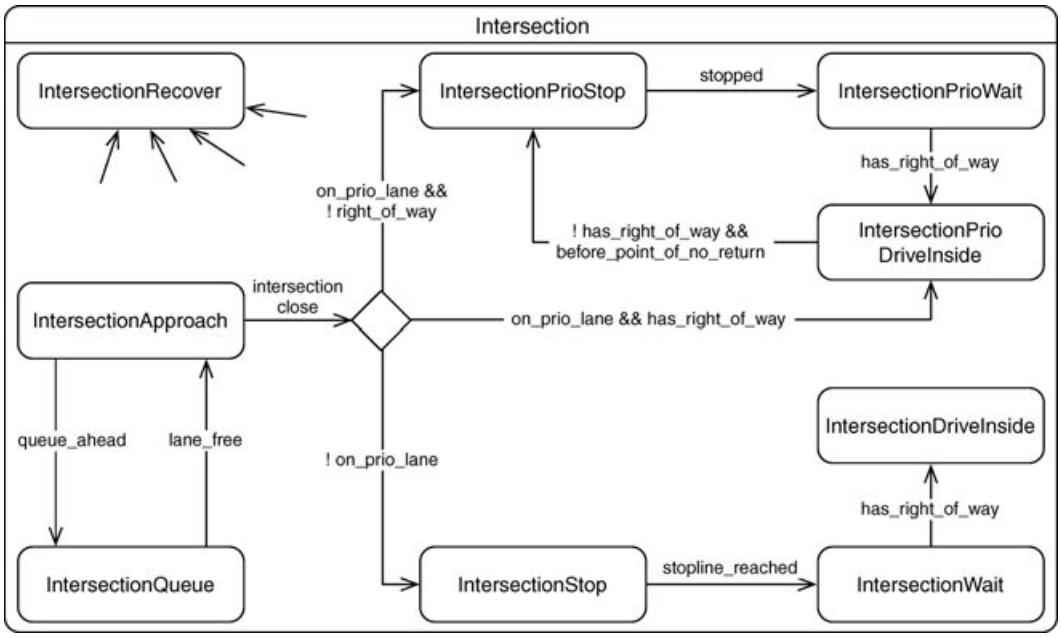


Figure 15. UML diagram of substate Intersection.

inaccuracy, hysteresis in the MTC is introduced by slightly reducing the requirements once the autonomous vehicle has set itself in motion.

Because the actual behavior of the other traffic participants can be roughly predicted at best, additional safety layers are introduced that prevent imminent collisions (see Section 8) in ticklish situations with emergency braking.

The conflict situations that arise from the competition are limited to

- intersections
- passing other cars
- changing lanes

Owing to the general formulation of the MTC, the different traffic situations can be accounted for with a corresponding parameter set.

For expository purposes the integration of the MTC in the intersection scenario will be described. Figure 15 shows the corresponding block diagram in UML notation. When the vehicle approaches the intersection, the HSM changes into the substate Intersection with the entry state IntersectionApproach. This state is active until the vehicle enters the intersection unless another traffic participant is perceived on the same lane between AnnieWAY and the intersection. In this case IntersectionQueue is activated until the other vehicle has passed the intersection and the lane is free.

In IntersectionApproach, as soon as AnnieWAY gets close to the intersection, the state transition splits up into

(a) IntersectionStop if AnnieWAY is on a stop road,
(b) IntersectionPrioDriveInside if AnnieWAY is on a priority road and no other vehicle has the right-of-way, or
(c) IntersectionPrioStop if AnnieWAY is situated on a priority road but needs to yield the right-of-way to priority vehicles, e. g., approaching traffic, before it may turn left.

In case a, AnnieWAY stops at the stop line and changes into the state IntersectionWait. In this state all vehicles are registered that are already waiting on another stop line that have the right-of-way according to the driving rules (four-way stop). As soon as these vehicles have passed the intersection and the MTC turns out positive for all visible priority vehicles, the state machine changes to IntersectionDriveInside and AnnieWAY merges into the moving traffic according to the safety parameters.

In case b, AnnieWAY drives into the intersection without stopping. If a priority vehicle is perceived shortly after driving inside the intersection (point of no return has not been passed yet) and the MTC turns out negative, the state machine switches to IntersectionPrioStop, which is equivalent to case c.

In case c, in IntersectionPrioStop AnnieWAY stops before crossing the opposing lane, waits until the MTC confirms that no danger comes from priority vehicles anymore, and turns left.

10. NAVIGATION IN UNSTRUCTURED ENVIRONMENT AND PARKING

As was described in Section 8, paths can be generated in a straightforward way by sampling from the geometric road network graph, when sufficient road geometry information is available. However, Urban Challenge regulations require navigating in unstructured environments (zones) that are described only by a boundary polygon. In the Urban Challenge, zones are used to outline parking lots and off-road areas. In this kind of area, a graph for path planning is not available. AnnieWAY's navigation system comprises a path planning algorithm that transcends the requirement for precise road geometry definition. It has also proven to be useful to plan narrow turns and as a general recovery mechanism when the vehicle gets off track, the road is blocked, or a sensible localization within the given road network is impossible.

10.1. Configuration Space Obstacles

We restrict search to the collision-free subset of configuration space (the vehicle's free space) by calculating configuration space obstacles from an obstacle map obtained from a 360-deg laser range scanner (see Section 4.1). The discrete nature of this obstacle map motivated dealing with configuration space obstacles in a discrete way as well (Kavraki, 1995), as opposed to more traditional approaches that require obstacle input in the form of polygonal data (Schwartz & Sharir, 1983; Šwestka & Overmars, 1997). Figures 16(a) and 16(b) illustrate how the robot's free space can be generated for a discrete set of orientations. By precomputing the free space in discretized form, a collision check for a certain configuration can be performed quickly in $\mathcal{O}(1)$ by a simple table lookup.

10.2. Search Graph and A*

We define an implicit search graph in which all paths are feasible. It is directly derived from a kinematic model of the car and not only guarantees feasibility of the generated path but also allows for straightforward design of a combined feed forward/feed backward controller (see Section 11).

A node of the search graph can be completely described by a tuple $(\mathbf{x},\psi,\delta)$, with $\mathbf{x}$, ψ, and δ denoting position, orientation, and steering angle (i.e., the deflection of the front wheels) of an instance of a kinematic one-track model [see Figure 17(a)]. Steering angle δ is from a set of n_δ discrete steering angles that are distributed equidistantly over the range of feasible steering: $D = \{\delta_1 \ldots \delta_{n\delta}\}$. To generate successors of a node, the kinematic model equations are solved for initial values taken from the node, a fixed arc length s, and a constant steering rate $\dot{\delta} = (\delta_p - \delta_i)/s$, spanning

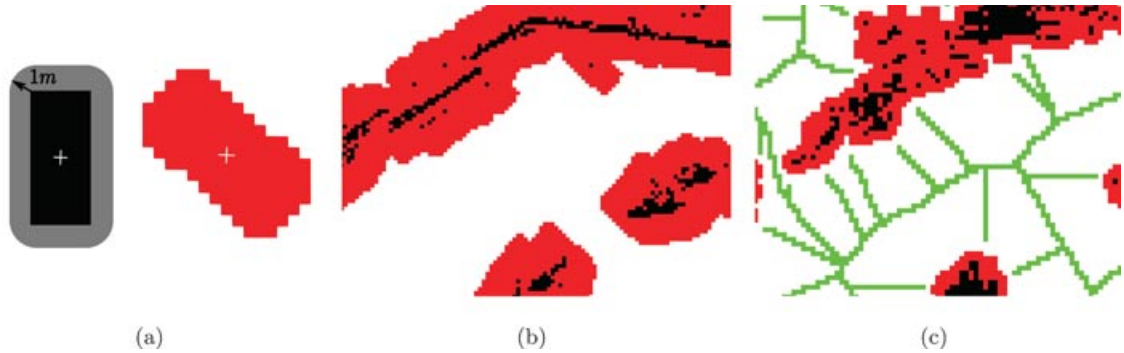


Figure 16. Configuration space obstacles. (a) A 1-m safety distance is added to the shape of the vehicle. Subsequent rotation and rasterization yields a convolution kernel for configuration space obstacle generation. (b) Result of convolving obstacle map with kernel from panel a. If the robot has the same orientation as the kernel and is placed in the red area, it must intersect with an obstacle. (c) Voronoi lines are generated as a set of eight connected pixels.

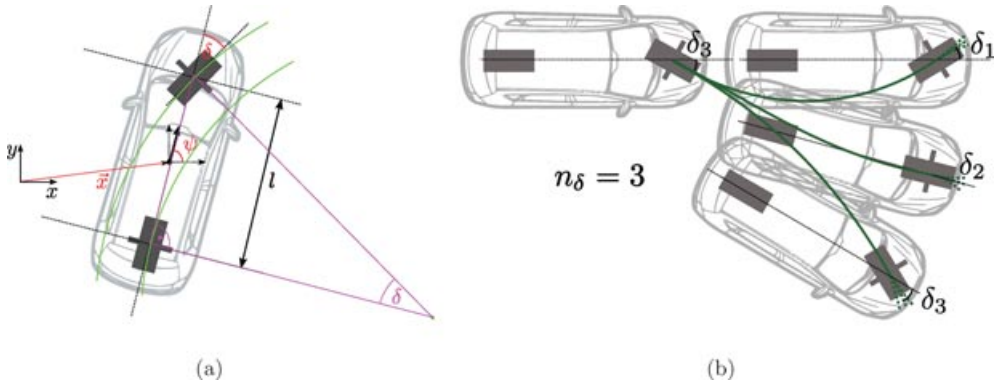


Figure 17. (a) Kinematic one-track model underlying both search graph and closed-loop control. (b) Search graph. Successors are generated for n_δ discrete steering angles.

clothoid-like arcs between the nodes. It is equivalent of driving the car model over a distance s at constant speed while uniformly turning the front wheels from δ_p to δ_i. For the set of nodes $\{(\mathbf{0}, 0, \delta_i),\ \delta_i \epsilon D\}$, this results in n_δ^2 successors and another n_δ^2 if backward motion is allowed. Successors of other nodes can be generated quickly from this precomputed set by subsequent rotation and translation [Figure 17(b)].

The search graph is expanded in this way by an A* search algorithm. A* search is a well-known concept in the domain of robotic path planning (Hwang & Ahuja, 1992) that allows for accelerating exploration of the search space by defining a cost function that gives a lower bound of expected cost to go for each node of the search graph. If the cost function underestimates the actual distance to the goal, A* is guaranteed to find the least-cost path. If the error of the cost function is big, A* quickly degenerates to an exponential time algorithm. This is common when a metric cost function is used that does not account for obstacle positions, so that search can get stuck in a dead-end configuration. We avoid this problem by designing an obstacle-sensitive cost function that accounts for the topology of the free space.

10.3. Cost Function

To guide the search process, we combined two different cost functions. The first one accounts for kinematic constraints of the vehicle, and the second one is derived from the Voronoi graph of the vehicle's free space and thus incorporates knowledge of shape and position of the obstacles.

10.3.1. Local Cost Function

As a local cost function, the so-called *RTR metric* is used. RTR (rotation-translation-rotation) paths connect two configurations by two circular arcs of minimum turning radius and a straight segment tangenting both. It can be shown easily (see Šwestka & Overmars, 1997) that for every pair of configurations a finite number of such paths can be constructed. The RTR metric is the arc length of the shortest such path. RTR paths neither have continous curvature nor are they optimal. The optimal, in terms of arc length, solutions to the local navigation problem are the so-called Reeds and Shepp paths (Reeds & Shepp, 1991). However, we prefer the RTR metric due to its computational simplicity. Figure 18(a) illustrates the RTR metric.

10.3.2. Voronoi-Based Cost Function

We construct a powerful, obstacle-sensitive cost function based on the Voronoi graph of the free space of the vehicle. Actually, a superset of the free space is used that is invariant to the vehicle's orientation. It is generated by generating configuration space obstacles for a disk-shaped structure that is the intersection of all structuring elements from Figure 16(a).

Our algorithm to calculate Voronoi lines from a binarized obstacle map is similar to that of Li and Vossepoel (1998); however, instead of using the vector distance map, we use the approximate chamfer metric to be able to label Voronoi lines using only two passes over the obstacle map. The method is derived from an algorithm (Borgefors, 1986; Li &

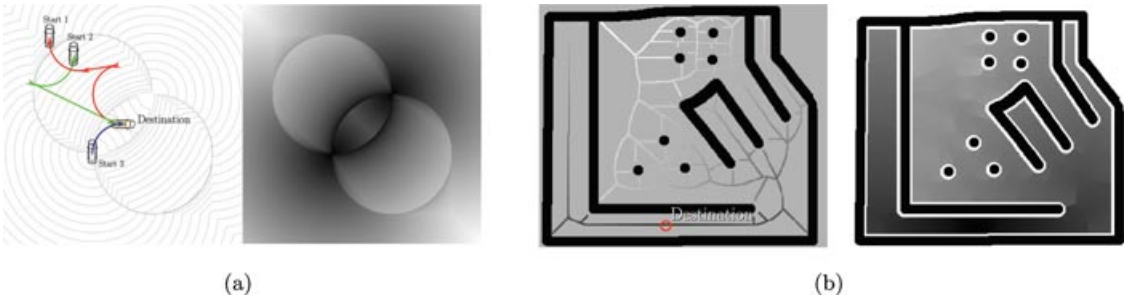


Figure 18. Cost functions. (a) RTR metric for three different starting positions. Left-hand side shows the minimum RTR paths; right-hand image the value of the RTR metric, densely evaluated on $\mathbb{R}^2$ (bright: high value; dark: low value). (b) Voronoi-based cost function. (Left) Voronoi graph labeled with distance by Dijkstras algorithm. (Right) Voronoi-based cost function evaluated densely on $\mathbb{R}^2$ by matching to the Voronoi graph.

Vossepoel, 1998) for calculating the Euclidean distance transform. It gives the Voronoi lines as a set of eight connected pixels.

After matching the target position to the closest point on the Voronoi graph, the Dijkstras algorithm is used to calculate the shortest-path distance to the target position for every point on the graph. Cost for a position not on the graph is derived by matching to the closest point on the graph and incorporating the matching distance in a way that yields a gradient of the cost function that is slightly sloped toward the Voronoi lines. Figure 18(b) shows an example.

Using this heuristic function is appealing for several reasons. Because the Voronoi lines comprise the complete topology of the free space, search cannot get stuck in a dead-end configuration, as is common with heuristics that do not incorporate knowledge of free-space topology and therefore grossly underestimate the cost in such a case. Additionally, the Voronoi lines have, as the centers of maximum inscribing circles, the property of being at the greatest distances possible from any obstacle. This is conveyed to the planned paths, giving reserves to account for control and measurement errors.

10.3.3. Combination of Cost Functions

We combine the two cost functions into one by the maximum operator. This procedure can be justified from the admissibility principle for heuristics in the context of an A* search. A heuristic is called admissible if it consistently underestimates the cost to the target node. Consequently, combining two heuristics via the maximum operator still gives an admissible heuristic. The result of comparing the two costs coincides with the practical experience that in the vicinity of the target position, cost is dominated by the necessity to maneuver in order to reach the destination in the right orientation, whereas the cost at large distances often is caused by the necessity to avoid obstacles. Figure 19 shows some results of the A* search using the search graph from Section 10.2 and the combined cost function.

11. VEHICLE CONTROL

The last step of the processing chain is the vehicle control, which can be separated into lateral and longitudinal controls. Because the distances to dynamic objects are fairly big in the Urban Challenge 2007 competition, for high-level decision making the problem of trajectory planning (coordinates of the desired vehicle position as a function of time) can be reduced to a combination of path planning (path geometries with no time dependencies) and determining the free section of the path rather than an exact desired position. The longitudinal strategy is thereby assigned to a lower level, which evaluates the free section of the path and induces the vehicle to go faster or slower. The information transfer of the interface is undertaken by so-called curve points, a discrete representation of the path geometry.

As the emphasis of the competition is on low to medium velocities, the nonholonomic single-track model holds and an orbital tracking controller (e.g., Müller, 2007) is chosen for the lateral dynamics in Section 11.1. This offers the advantage of a velocity-independent transient lateral behavior for the closed-loop system. Suppose that the vehicle had an offset from the planned path of a couple of centimeters

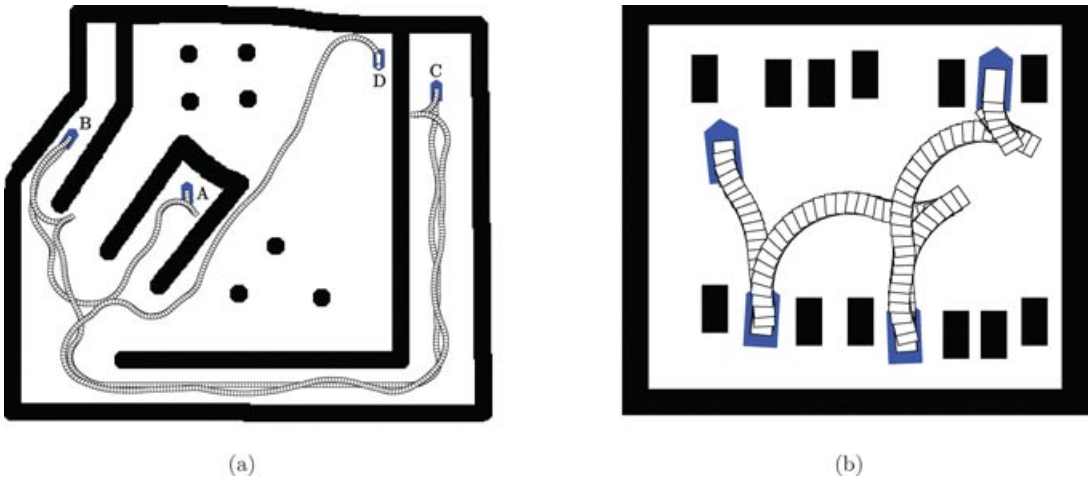


Figure 19. Some results of path planning on simulated map data. (a) Navigating large distances in a maze-like environment. Planning was from A to B, B to C, and C to D subsequently. (b) Some difficult parking maneuvers performed subsequently. Robot started on the right.

caused by sensor drift of the navigation system: the lateral controller would reduce the error over a certain traveled distance rather than over time and avoids unpredictable overshoots of the front end, which might lead to collisions.

From the longitudinal controller's point of view, the vehicle drives on rails, as the lateral controller minimizes the lateral offset. Thus, the longitudinal control strategy faces solely the task of following moving objects, stopping at certain points, maintaining the maximum speed, and changing direction along the given path. For this purpose different controllers are designed in Section 11.2 that are included in an override control strategy ensuring bumpless transfers between them. The output of every longitudinal controller is the vehicle's acceleration a. This acceleration will be converted to the manipulated variables accelerator pedal value ϕ_{gas} and brake pressure p_{brake} in a cascaded acceleration controller exceeding the scope of this contribution.

11.1. Orbital Tracking Controller

The dynamics of a nonholonomic vehicle (Figure 20) in local coordinates s_c, d, and $\Delta\psi$ are given by

$$\frac{\mathrm{d}}{\mathrm{d}t}\begin{bmatrix} s_c \\ d \\ \Delta\psi \end{bmatrix} = \begin{bmatrix} \dfrac{\cos\Delta\psi}{1-d\kappa_c(s_c)} \\ \sin\Delta\psi \\ \dfrac{\tan\delta}{l} - \kappa_c(s_c)\dfrac{\cos\Delta\psi}{1-d\kappa_c(s_c)} \end{bmatrix} v, \quad (18)$$

where the steering wheel angle δ and the longitudinal velocity v are the system's input, d is the lateral offset to the path, $\Delta\psi$ is the angle between the vehicle and the tangent to the path, and l is the distance between the rear and the front axles. The singularity at $1-d\kappa_c(s_c)=0$ is no restriction in practice since $d \ll 1/\kappa_c(s_c)$.

Because orbital tracking control does not have any time dependencies, Eq. (18) can be rewritten with the arc length s_c as the new time parameterization.

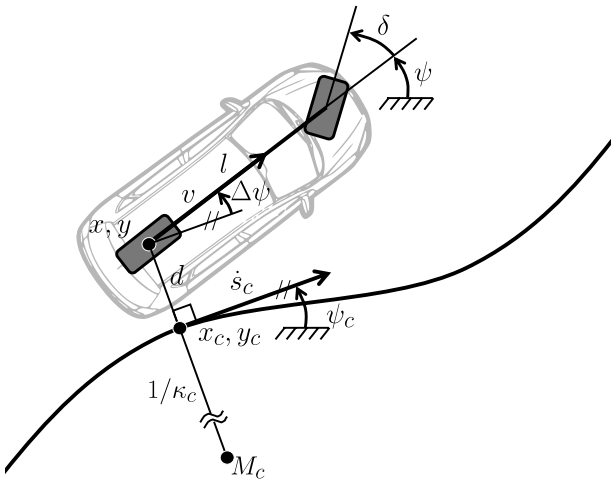


Figure 20. Nonholonomic one-track model.

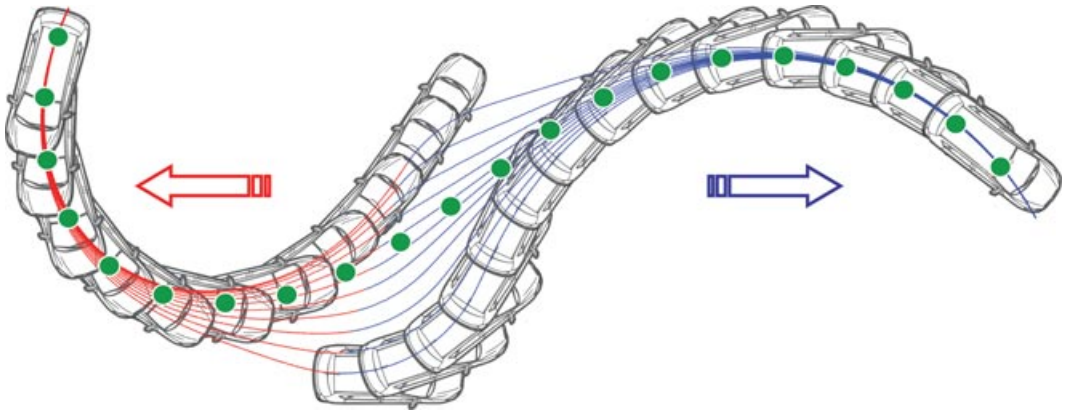


Figure 21. Trajectories for different initial positions.

With $\mathrm{d}/\mathrm{d}t\,() = \mathrm{d}/\mathrm{d}s_c\,() \cdot \mathrm{d}s_c/\mathrm{d}t$, it becomes

$$\frac{\mathrm{d}}{\mathrm{d}s_c}\begin{bmatrix} s_c \\ d \\ \Delta\psi \end{bmatrix} = \begin{bmatrix} 1 \\ \sin\Delta\psi \cdot \dfrac{1 - d\kappa_c(s_c)}{\cos\Delta\psi} \\ \dfrac{\tan\delta}{l} \cdot \dfrac{1 - d\kappa_c(s_c)}{\cos\Delta\psi} - \kappa_c(s_c) \end{bmatrix}. \tag{19}$$

For small deviations d and $\Delta\psi$ from the desired curve and $\mathrm{d}/\mathrm{d}s_c\,() = ()'$, a partial linearization leads to

$$\begin{bmatrix} d \\ \Delta\psi \end{bmatrix}' = \begin{bmatrix} 0 & 1 \\ 0 & 0 \end{bmatrix}\begin{bmatrix} d \\ \Delta\psi \end{bmatrix} + \begin{bmatrix} 0 \\ -1 \end{bmatrix}\kappa_c + \begin{bmatrix} 0 \\ \frac{1}{l} \end{bmatrix}\tan\delta. \tag{20}$$

The linearizing control law

$$\delta = \arctan(-lk_0 d - lk_1\Delta\psi + l\kappa_c) \tag{21}$$

$$= \arctan(-k_1^* d - k_2^*\Delta\psi + l\kappa_c) \tag{22}$$

with $k_0, k_1 > 0$ yields the stable linear error dynamics

$$\frac{\mathrm{d}}{\mathrm{d}s_c}\begin{bmatrix} d \\ \Delta\psi \end{bmatrix} = \begin{bmatrix} 0 & 1 \\ -k_0 & -k_1 \end{bmatrix}\begin{bmatrix} d \\ \Delta\psi \end{bmatrix} \tag{23}$$

with respect to s_c with the characteristic polynomial $\lambda^2 + k_1\lambda + k_0 = 0$. As long as $\dot{s_c} > 0$, the system is also stable with respect to time. For backward driving, the signs of k_0 and k_1 have to be adjusted to the applied sign convention and yield exactly the same error dynamics as for forward driving.

Figure 21 shows the transient behavior to different initial errors $\Delta\psi$ and d for forward (blue) and backward driving (red) simulated with MATLAB/SIMULINK. As parameters for the simulation, the Passat's axis distance $l = 2.72$, a maximum steering angle of $\delta_{\max} = 30$ deg, the controller parameters $k_0 = 0.25\ l$ and $k_1 = 1.25\ l$, and equidistant curve point with $\Delta = 2$ m were chosen. Obviously neither the input saturation $\delta_{\max}$ nor the discrete representation of the curve causes any significant problems.

11.2. Longitudinal Controller System

11.2.1. Following Controller

Because the acceleration of a leading vehicle is hard to determine, it is assumed that the vehicle keeps its velocity v_B constant. Choosing the distance d_f and its time derivative $\dot{d}_f$ as the state variables and AnnieWAY's acceleration $af = \dot{v}$ as the input, the system's dynamics are given by

$$\frac{\mathrm{d}}{\mathrm{d}t}\begin{bmatrix} d_f \\ \dot{d}_f \end{bmatrix} = \begin{bmatrix} 0 & 1 \\ 0 & 0 \end{bmatrix}\begin{bmatrix} d_f \\ \dot{d}_f \end{bmatrix} + \begin{bmatrix} 0 \\ -1 \end{bmatrix} a_f. \tag{24}$$

As DARPA requires the vehicle to maintain a minimum forward vehicle separation of one vehicle length minimum and one length for every additional 10 mph (16 km/h), the desired distance $d_{f,d}$ can be calculated by

$$d_{f,d} = d_{f,0} + \tau v \tag{25}$$

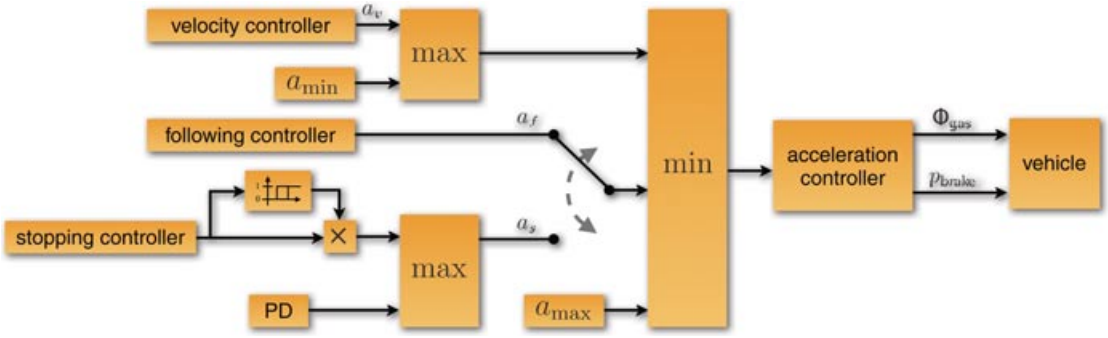


Figure 22. Longitudinal override control strategy.

with the according parameters $d_{f,0}$ and τ. Considering the acceleration $\dot{v}_B$ of the leading vehicle an unmeasurable disturbance, the linear set-point control law

$$a_f = c_0(d_f - d_{f,d}) + c_1\dot{d}_f \tag{26}$$

$$= c_0(d_f - d_{f,d}) + c_1(v_B - v) \tag{27}$$

and $v = v_B - \dot{d}_f$ yields the total system

$$\frac{\mathrm{d}}{\mathrm{d}t}\begin{bmatrix} d_f \\ \dot{d}_f \end{bmatrix} = \begin{bmatrix} 0 & 1 \\ -c_0 & -c_0\tau - c_1 \end{bmatrix}\begin{bmatrix} d_f \\ \dot{d}_f \end{bmatrix} \tag{28}$$

$$+ \begin{bmatrix} 0 \\ c_0(d_{f,0} + \tau v_B) \end{bmatrix}. \tag{29}$$

The characteristic polynomial $\lambda^2 + (c_0\tau + c_1)\lambda + c_0 = 0$ can directly be read off from Eq. (28). A double eigenvalue $\lambda_{1/2} = -1$ leads to a pleasant and yet safe following behavior.

11.2.2. Stopping Controller

The following controller of the preceding section leads to a behavior that can best be described as *flowing with the traffic*. By contrast, the stopping controller should come to a controlled stop at a certain point as fast as possible without exceeding any comfort criteria. The control law

$$a_s = -\frac{v^2}{2(d_f - d_\Delta)} \tag{30}$$

leads to a constant deceleration until the vehicle is d_Δ away from the stop point. To prevent the controller from decelerating too soon and switching on and off, a hysteresis with the thresholds $a_{s,\max}$ and $a_{s,\min}$, as shown in Figure 22, is introduced. The singularity at $d_f = d_\Delta$ is avoided by a PD position controller that takes over via a min-operator and ensures a smooth and safe stop at the end.

11.2.3. Velocity Controller

As $\dot{v} = a$, the simple proportional velocity control law

$$a_v = -c_v(v - v_d) \tag{31}$$

stabilizes AnnieWAY's velocity v to the desired velocity v_d with a PT_1 behavior.

11.2.4. Override Control Strategy

All three previously introduced controllers are combined by an override control strategy depicted in Figure 22. The bumpless transfer between velocity control and following/stopping control is ensured by the max operator. Additional saturation, realized by $a_{\max}$ and $a_{\min}$, prevents the vehicle from inappropriately high acceleration or deceleration without reducing safety.

12. RESULTS AND LESSONS LEARNED

Originally 89 teams entered the competition, 11 of which were sponsored by the organizer. After several stages, 36 of those teams were selected for the semifinal. There, AnnieWAY accomplished safe conduct of

a variety of maneuvers, including

- regular driving on lanes
- turning at intersections with oncoming traffic
- lane-change maneuvers
- vehicle following and passing
- following order of precedence at four-way stops
- merging into moving traffic

Although the final event was originally planned to challenge 20 teams, only 11 finalists were selected by the organizers due to safety issues. AnnieWAY entered the final and was able to conduct a variety of driving maneuvers. Owing to the limited resources, the team arrived in the United States only about 2 weeks before the competition. We therefore developed a powerful simulation environment that allowed us to replace most on-the-vehicle tests by open-loop tests and simulations. Nevertheless, the lack of an opportunity to conduct on-vehicle tests was always considered a significant risk for our team and actually resulted in late testing and software adaptation activities that proceeded even during the National Qualifying Event (NQE). Still, AnnieWAY entered the finals with software modules that mainly proved to be robust and were thoroughly validated in simulations but were hardly tested on-board. To illustrate the enhancement process and the lessons learned, we report some of the most recent modifications in the following.

During our first run on track, a problem with the object tracking appeared: Vehicles were detected and tracked only up to a distance of approximetely 60 m. As a result, our vehicle behaved rather shortsightedly in some situations. Increasing this distance proved to be more than a mere parameter-setting issue. For the smaller range of the grid map, a cell size of 30 cm proved to be sufficient for all tasks. However, at large distances, the decreasing spatial resolution of the sensor meant that individual cells had no or not enough measurements. At a distance of 60 m, the laser scanner resolution was some 25 cm. Hence a sparse set of distance cells did not experience any update in its occupancy data, which resulted in problems for the subsequent tracking module. Adaptation of the grid resolution to the actual sensor capabilities, i.e., reducing the grid resolution with distance, yielded satisfactory object tracking for the remainder of the competition on track A.

In still-ongoing research, we investigate the use of probabilistic inference for automated behavior generation that in the long term might be advantageous as far as design complexity is concerned (Stiller, Färber, & Kammel, 2007). However, after initial simulations, simple and proven methods such as the state machine were selected. It contained a logical design error that omitted consideration of other four-way-stop vehicles after 10 s. If two or more vehicles were already standing at a four-way-stop in track C, our vehicle stopped for 10 s only instead of $n \times 10$ s, with the result that the correct procedure was not kept. The correction of this mistake was quite trivial, so that the more important mistake was the late notice of it. Tests with multiple other dummy-vehicles were conducted in preparation for the race, but the case in which all of the vehicles were in complete stop at the intersection when the test vehicle approached did not occur. To avoid such surprises in the future, organized test plans with a list of possible configurations that covers all relevant cases will be designed in the future.

At the low speed during the race, one minor problem that remained unsolved was an insufficient

Figure 23. Three steps of AnnieWAY's course driven autonomously in the finals.

adjustment of our reactive component to the controller. This eventually caused slow or oscillating movements during the testing and race. The tentacle-based approach was extracted from a completely different system architecture and was not designed for parallel operation with any of the other components. The adaption of the controller interface was conducted by means of parameter optimization. However, the obviously better solution was to apply fundamental adjustments such as employment of identical dynamical models in the controller and other modules. The implementation of identical models in navigation and road planning modules indicates the effectiveness of such an approach. Hence, maybe the most important lesson learned is that the modular architecture and simulation framework proved to be powerful and even allowed substitution of many on-road experiments; however, the modular structure should employ identical world models in order to facilitate bidirectional feedback during development and the testing process. On the other hand, given the budgetary and time constraints of our team, the approach to adopt and adapt existing components from all partners permitted dedication of the few resources to the design of new components and of components that were insufficient for the Urban Challenge.

In the finals, AnnieWAY was one of the few cars that drove collision-free. However, it stopped due to a software exception that occurred while switching from road planning to zone navigation and was caught by an error handler. The result was a hanging process, which could not be detected by the watchdog module. In the case of a module crash, the watchdog should have taken care of restarting this process and possibly other processes, according to the dependencies. As this was not possible in this situation, the process was frozen but still alive. Permanent supervision of correct functioning of the modules could solve such problems. In enhancements of our system that were implemented after the competition, the watchdog determines functionality based on internal data flow and correct timings. As only a preliminary and mainly untested implementation of that feature was available at the race, it had intentionally not been activated at that time. Figure 23 depicts three examples of the vehicle's actual course taken from a log file and superimposed on an aerial image. The right-most figure shows the stopping position in the finals.

We now point out some results of the navigation module. Figure 24(a) illustrates one test driven in a parking area close to our test ground. Unlike the required navigation task in the Urban Challenge, the chosen setup features many surrounding obstacles such as other cars and curbs.

Search time remains below 2 s in all practical situations. Though the environment is assumed to be static, this is fast enough to cope with slow changes in the environment by continuous replanning. Additionally, to avoid collision with fast-moving objects, a lower-level process continuously determines the free

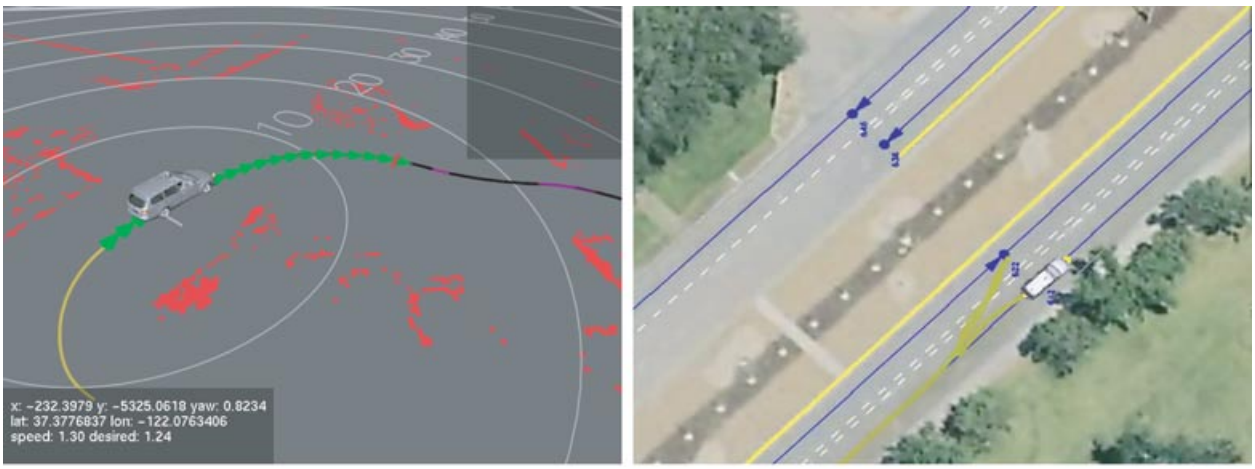


Figure 24. (a) Path planning in heavily occupied zone with mapper input (red) and sampled way points as output to the controller (green). (b) Recovery maneuver during final event; driven path is marked olive.

section of the planned path and, if necessary, invokes a new search. The lateral controller follows the generated paths precisely enough to implement all of the intended maneuvers.

Besides path planning in parking areas, the zone-navigation module was used as a recovery option in the case of continuous blocking of lanes or intersections. An erroneously detected obstacle on the left lane forced activation of the navigation module during the final event. The vehicle was successfully brought back on track after a back-up maneuver, as can be seen in Figure 24(b).

13. CONCLUSIONS

The autonomous vehicle AnnieWAY is capable of driving through urban scenarios and successfully entered the finals of the 2007 DARPA Urban Challenge competition. In contrast to earlier competitions, the Urban Challenge required the conduct of missions in "urban" traffic, i.e., in the presence of other autonomous and human-operated vehicles. The major challenge imposed was collision-free and rule-compliant driving in traffic. AnnieWAY is based on a simple and robust hardware architecture. In particular, we rely on a single computer system for all tasks but low-level control. Environment perception is mainly conducted by a roof-mounted laser scanner that measures range and reflectivity for each pixel. Whereas the former is used to provide 3D scene geometry, the latter allows robust lane marker detection. Mission and maneuver selection is conducted via a HSM that specifically asserts behavior in accordance with California traffic laws. More than 100 h of urban driving without human intervention in complex urban settings with multiple cars, correct precedence order decision at intersections, and—last not least—the entry in the finals underline the performance of the overall system. Through the Urban Challenge, DARPA has impressively stimulated research on autonomous vehicles not merely in the United States but even to some extent on the international level. Future challenges organized in cooperation with other international authorities could further expedite technology for the vision of collision-free traffic.

ACKNOWLEDGMENTS

The authors gratefully acknowledge the great collaboration of their partners from Universität Karlsruhe, Technische Universität München, and Universität der Bundeswehr München. Special thanks are directed to Annie Lien for her engagement as our offcial team leader, for her dedication and goal-oriented attitude to the project, and last but not least for the brilliant organization that she provided. Most of the modules used were adapted from related research projects. In particular, this work would not have been possible without the ongoing research of the Transregional Collaborative Research Centre (TCRC) 28 "Cognitive Automobiles." The two projects cross-fertilized each other and revealed significant synergy. The authors gratefully acknowledge support of the TCRC by the Deutsche Forschungsgemeinschaft (German Research Foundation).

REFERENCES

Bertozzi, M., Broggi, A., & Fascioli, A. (2000). Vision-based intelligent vehicles: State of the art and perspectives. Journal of Robotics and Autonomous Systems, 32, 1–16.

Biber, P., & Strasser, W. (2006). nscan-matching: Simultaneous matching of multiple scans and application to SLAM. In IEEE International Conference on Robotics and Automation, Orlando, FL (pp. 2270–2276). IEEE.

Borgefors, G. (1986). Distance transformations in digital images. Computer Vision, Graphics, and Image Processing, 34(3), 344–371.

Bosse, M., Newman, P. M., Leonard, J. J., Soika, M., Feiten, W., & Teller, S. J. (2003). An Atlas framework for scalable mapping. In Proceedings of the 2003 IEEE International Conference on Robotics and Automation, Taipei, Taiwan (pp. 1899–1906). IEEE.

Bracewell, R. N. (1990). Numerical transforms. Science, 248(4956), 697–704.

DARPA (2005). Grand Challenge '05. http://www.darpa.mil/grandchallenge/overview.html.

Dickmanns, E. D., Behringer, R., Dickmanns, D., Hildebrandt, T., Maurer, M., Thomanek, F., & Schielen, J. (1994). The seeing passenger car "VaMoRs-P." In Proceedings of the Symposium on Intelligent Vehicles, Paris (pp. 68–73). IEEE.

Franke, U., Gavrila, D., Gern, A., Goerzig, S., Jansen, R., Paetzold, F., & Wöhler, C. (2001). From door to door—Principles and applications of computer vision for driver assistant systems. In L. Vlacic, F. Harashima, & M. Parent (Eds.), Intelligent vehicle technologies: Theory and applications (chapter 6, pp. 131–188). Oxford: Butterworth Heinemann.

Goebl, M., & Färber, G. (2007a). Eine realzeitfähige Softwarearchitektur für kognitive Automobile. In K. Berns & T. Luksch (Eds.), Autonome Mobile Systeme 2007, Informatik Aktuell (pp. 198–204). Berlin: Springer-Verlag.

Goebl, M., & Färber, G. (2007b). A real-time-capable hard- and software architecture for joint image and

knowledge processing in cognitive automobiles. In Proceedings of the IEEE Intelligent Vehicles Symposium, Istanbul, Turkey (pp. 734–739). IEEE.

Hummel, B., Kammel, S., Dang, T., Duchow, C., & Stiller, C. (2006). Vision-based path-planning in unstructured environments. In IEEE Intelligent Vehicles Symposium, Tokyo, Japan (pp. 176–181). IEEE.

Hwang, Y. K., & Ahuja, N. (1992). Gross motion planning—A survey. ACM Computing Surveys, 24(3), 219–291.

Kalman, R. (1960). A new approach to linear filtering and prediction problems. Journal of Basic Engineering, 82(1), 35–45.

Kavraki, L. (1995). Computation of configuration-space obstacles using the fast Fourier transform. IEEE Transactions on Robotics and Automation, 11(3), 408–413.

Li, H., & Vossepoel, A. M. (1998). Generation of the Euclidean skeleton from the vector distance map by a bisector decision rule. In CVPR '98: Proceedings of the IEEE Computer Society Conference on Computer Vision and Pattern Recognition (p. 66), Washington, DC. Washington, DC: IEEE Computer Society.

Müller, B., & Deutscher, J. (2007). Orbital tracking control for car parking via control of the clock. Methoden und Anwendungen der Regelungstechnik, Erlangen-Münchener Workshops 2005 und 2006 (pp. 1–8). Aachen: Shaker Verlag.

Nagel, H.-H., Enkelmann, W., & Struck, G. (1995). FhG-Co-Driver: From map-guided automatic driving by machine vision to a cooperative driver support. Mathematical and Computer Modelling, 22, 185–212.

Özgüner, Ü., Stiller, C., & Redmill, K. (2007). Systems for safety and autonomous behavior in cars: The DARPA Grand Challenge experience. IEEE Proceedings, 95(2), 397–412.

Reeds, J., & Shepp, R. (1991). Optimal paths for a car that goes both forward and backward. Pacific Journal of Mathematics, 145(2), 144–154.

Schwartz, J. T., & Sharir, M. (1983). On the piano movers' problem, ii: General techniques for computing topological properties of real algebraic manifolds. Advances in Applied Mathematics, 4, 298–351.

Stiller, C., Färber, G., & Kammel, S. (2007). Cooperative cognitive automobiles. In Proceedings of the IEEE Intelligent Vehicles Symposium, Istanbul, Turkey (pp. 215–220). IEEE.

Šwestka, P., & Overmars, M. (1997). Motion planning for car-like robots using a probabilistic learning approach. International Journal of Robotics Research, 16(2), 119–143.

Thorpe, C., Hebert, M. H., Kanade, T., & Shafer, S. A. (1988). Vision and navigation for the Carnegie-Mellon Navlab. IEEE Transactions on Pattern Analysis and Machine Intelligence, 10(3), 362–373.

Thrun, S. (2002). Robotic mapping: A survey. In G. Lakemeyer & B. Nebel (Eds.), Exploring artificial intelligence in the new millennium. San Francisco: Morgan Kaufmann.

Thrun, S. (2003). Learning occupancy grid maps with forward sensor models. Autonomous Robots, 15(2), 111–127.

Thrun, S., Montemerlo, M., Dahlkamp, H., Stavens, D., Aron, A., Diebel, J., Fong, P., Gale, J., Halpenny, M., Hoffmann, G., Lau, K., Oakley, C., Palatucci, M., Pratt, V., Stang, P., Strohband, S., Dupont, C., Jendrossek, L.-E., Koelen, C., Markey, C., Rummel, C., Niekerk, J., Jensen, E., Alessandrini, P., Bradski, G., Davies, B., Ettinger, S., Kaehler, A., Nefian, A., & Mahoney, P. (2006). Stanley: The robot that won the DARPA Grand Challenge. Journal of Field Robotics, 23(9), 661–692.

von Hundelshausen, F., Himmelsbach, M., Mueller, A., & Wuensche, H.-J. (2008). Tentacles—A biologically inspired approach for robot navigation. Journal of Field Robotics, 25(9), 640–673.