Driving with Tentacles: Integral Structures for Sensing and Motion

Felix von Hundelshausen, Michael Himmelsbach, Falk Hecker, Andre Mueller, and Hans-Joachim Wuensche
Autonomous Systems Technology
Department of Aerospace Engineering
University of the Federal Armed Forces Munich
85579 Neubiberg, Germany
e-mail: felix@unibw.de,
michael.himmelsbach@unibw.de,
falk.hecker@unibw.de, andre.mueller@unibw.de,
joe.wuensche@unibw.de

Received 18 January 2008; accepted 24 June 2008

In this paper we describe a LIDAR-based navigation approach applied at both the C-Elrob (European Land Robot Trial) 2007 and the 2007 DARPA Urban Challenge. At the C-Elrob 2007 the approach was used without any prior knowledge about the terrain and without global positioning system (GPS). At the Urban Challenge the approach was combined with a GPS-based path follower. At the core of the method is a set of "tentacles" that represent precalculated trajectories defined in the ego-centered coordinate space of the vehicle. Similar to an insect's antennae or feelers, they fan out with different curvatures discretizing the basic driving options of the vehicle. We detail how the approach can be used for exploration of unknown environments and how it can be extended to combined GPS path following and obstacle avoidance allowing safe road following in case of GPS offsets.

1. INTRODUCTION

In this paper, we put forth a very simple method for autonomous robot navigation in unknown environments. The method is not restricted to a particular robot or sensor; however, we demonstrated it on the two vehicles in which the approach was integrated. Our method was applied at the Civilian European Land Robot Trial 2007 (C-Elrob 2007) on our VW-Touareg MuCAR-3 (Munich Cognitive Autonomous Robot, third generation) and the 2007 DARPA Urban Challenge (UC07) on the VW-Passat of Team AnnieWAY. Both vehicles had an almost identical hardware setup, in particular both being equipped with a Velodyne 64-beam, 360-deg LIDAR (see Figure 1).

Although quite complex approaches to mobile robot navigation exist [e.g., solving the SLAM (simultaneous localization and mapping) problem (Dissanayake, Newman, Clark, Durrant-Whyte, & Csorba, 2001; Julier & Uhlmann, 2001), methods based on trajectory planning (Sariff & Buniyamin, 2006)], our research was driven by the search for simplicity: *What is the simplest approach that lets a robot safely drive in an unknown environment?* Intentionally, we write "drive" instead of "explore" because we do

(a) (b)

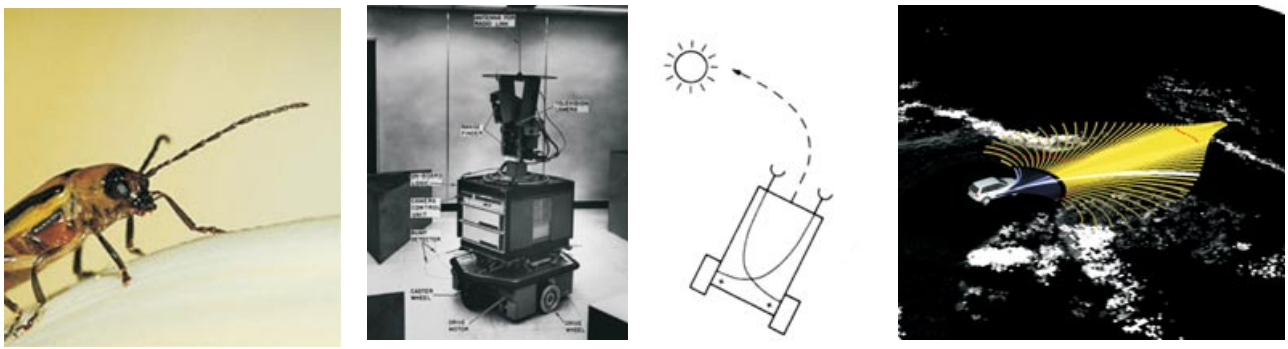
Figure 1. The two vehicles our approach was tested with. (a) The VW-Passat of DARPA Urban Challenge finalist Team AnnieWAY. (b) The VW-Touareg MuCAR-3 (Munich Cognitive Autonomous Robot, third generation), a C-Elrob 2007 champion. Both vehicles are equipped with a 360-deg Velodyne LIDAR as primary sensor scanning its environment at 10 Hz using 64 laser beams.

not demand to construct a map of the environment [in contrast to SLAM approaches (Dissanayake et al., 2001; Julier & Uhlmann, 2001)] but just demand safe driving within that environment. Our basic intention underlying this paper is to let our robot move within an unknown environment similarly to how a beetle would crawl around and uses its antennae to avoid obstacles. Indeed, our basic approach consists of using a set of virtual antennae that we call "tentacles" probing an ego-centered occupancy grid for drivability. Of course, the idea of using antennae is not new: In fact, one of the first robots, "Shakey" (Nilsson, 1984), used "cat-whiskers" (microswitches actuated by a 6-in.-long coil spring extended by piano wires to provide longer reach) to sense the presence of a solid object within the braking distance of the vehicle when traveling at top speed [see Figure 2(b)]. In his cybernetic thought games Braitenberg (1984) showed that complicated behavior can emerge by very simple mechanisms, and almost all of his vehicles (known under the term *Braitenberg vehicles*) use sensors resembling an insects's antennae [Figure (2c)].

Some systems in mobile robotics integrate modules that are similar to our approach in that some sort of precalculated trajectories are verified to be drivable: Kelly and Stentz (1998) follow the idea of model-referenced control (Landau, 1979) implemented through command space sampling and evaluating a set of candidate trajectories. In Lamon, Kolski, Triebel, Siegwart, & Burgard, 2006) a set of feasible arcs is used as a first step in a path planning algorithm. Stanley, the robot that won the DARPA Grand Challenge in 2005, used a set of candidate paths ("nudges" and "swerves") for path planning (Thrun et al., 2006).

In the DAMN (Distributed Architecture for Mobile Navigation) framework (Rosenblatt, 1997), four different arbitration schemes for integrating the results of different distributed behaviors are proposed. One of those arbitration schemes is actuation arbitration, and the work of Rosenblatt (1997) sketches an example of that scheme, in which different turn behaviors cast votes on candidate vehicle curvature commands. The difference of our approach is that the candidate commands are not only used as candidate instances in a command space but our "tentacles" are also used as perceptual primitives, and a very detailed process of how those primitives are used to evaluate an occupancy grid is proposed. This evaluation process includes some new structures and aspects, including the differentiation between a support and a classification area, the use of a longitudinal histogram for classifying tentacles as being drivable or not, determining the distance to the first obstacle along a tentacle, and a speed-dependent evaluation length (the crash distance) that permits reduction of the number of required primitives drastically.

Another work in the context of planetary rover exploration that looks very similar to our approach at first glance is GESTALT (Grid-based Estimation



(a) (b) (c) (d)

Figure 2. (a) Among other functions, insects use antennae as tactile sense. (b) Shakey (Nilsson, 1984)—one of the first robots—uses mechanical cat-whiskers to detect collisions. (c) Braitenberg vehicles use some sort of antennae with simple, hard-wired reactive mechanisms to produce complex behaviors. (d) In our approach we use speed-dependent sets of antennaes that we call "tentacles."

of Surface Traversability Applied to Local Terrain) (Goldberg, Maimone, & Matthies, 2002). Similar to our approach, GESTALT selects an arc with maximum goodness according to various criteria; however, the main difference with our approach is that GESTALT uses a global grid and accumulates data over time in this grid. In doing so, the approach becomes an instance of the SLAM problem:

> At the end of each step, sensors are expected to provide a reasonably accurate estimate of the rover's new position. GESTALT does not require that the rover motion exactly match that with the commanded, but it does assume that wherever the rover ended up, its relative position and orientation can reasonably be inferred and provided as input. That is one limitation of the system, that it relies on other modules to deal with myriad position estimation problems (slipping in sand, getting stuck on a rock, freeing a jammed wheel, etc). (Goldberg et al., 2002; p. 4, last paragraph)

In contrast, our approach does not accumulate data and uses a local ego-centered grid, thereby avoiding the SLAM problem. Another difference with our approach is that it provides speed-dependent mechanisms for evaluating tentacles. This speed dependency is a crucial point in our approach because it allows us to restrict the motion primitives to a small set while still being able to drive through narrow passages having shapes other than circular arcs.

The work most similar to our approach is that of Coombs, Murphy, Lacaze, and Legowik (2000): Here, a fixed tree structure with five levels (the root being located at the robot's position) is used. The first level consists of 20-m-long precalculated clothoids, and the other levels consist of straight-line connections. Within this tree, all possible paths from the root to the leaves form the set of possible trajectories. The whole tree consists of about 4,000 edges, resulting in a combinatorial power of more than 15 million trajectories. In contrast to this, our approach works well with only 81 precalculated tentacles at a given speed, and no combinations of adjacent path fragments are required. This low number of possible path fragments is possible due to the special speed-dependent mechanism in our tentacle-evaluation process (to be detailed later) and is sufficient even in scenarios with narrow curved roads. All path fragments take the shape of circular arcs. In contrast to the aforementioned approaches, the basic samples for maneuvers are more short termed and more reactive in the sense of simple Braitenberg vehicles in our work; however, as will be shown, the way of evaluating the different driving options is much more detailed and multifaceted than existing approaches. The approach as detailed in this paper was tested excessively on common residential roads and offroad terrain: Our algorithm was successfully demonstrated at both the C-Elrob 2007 [driving record time in a combined urban and nonurban course including serpentines without using global positioning system (GPS), driving 90% of the course autonomously] and the 2007

DARPA Urban Challenge (getting into the final).[1] Our approach is fully described, simple to implement, and ready to use.

The remainder of the paper is organized as follows: In Section 2 we detail the generation process of an ego-centered occupancy grid, the input domain for our method. In Section 3 we detail the structure and generation of our tentacles. Section 4 describes how the "best tentacle" is selected in each iteration and Section 5 how this tentacle is executed. Section 6 analyzes our approach with respect to vehicle dynamics, computing upper bounds for path deviations. Section 7 describes experiments and the performance at competitions in which our approach was used and details its overall system integration. Also, some lessons learned are described in this section. Finally Section 8 concludes our paper. Throughout our paper, we specify the values of all parameters, such that our approach may be reproduced and serve as a reference for future improvements.

2. OCCUPANCY GRID

We use a two-dimensional occupancy grid with 512×512 cells, each covering a small ground patch of 25×25 cm. Each cell stores a single floating point value expressing the degree of how occupied that cell is by an obstacle. In our implementation this value is a metric length with the physical unit meters. Before we detail its meaning and calculation, note that in our approach we create a new occupancy grid on each new LIDAR rotation, i.e., every 100 ms as our LIDAR is set to turn at 10 Hz. Thus, we do not accumulate data for a longer time. The reasons for this decision are as follows. First, one rotation of our 64-beam Velodyne sensor supplies about 100,000 three-dimensional (3D) points, which proved to be sufficient. Second, the quality of an accumulated occupancy grid can easily deteriorate if the physical movement of the sensor is not estimated with very high precision. Small angular deviations in the estimate of the sensor's pose can result in large errors. Registering scans against each other, e.g., using the iterative closest point (ICP) (Besl & McKay, 1992) algorithm or some of its derivatives, could solve this problem but would require substantial additional computational load. Similarly, accumulating data in a grid requires additional time-consuming maintenance operations, such as copying or removing data from the grid. A scrolling or wrappable map as proposed in Kelly and Stentz (1998) is not expedient in our case because our method requires an ego-centered grid for efficiency. Hence, we decided not to accumulate data, although this question might remain controversial:

Consider, for example, the case of the vehicle approaching a pavement edge. Depending on how the sensor is mounted, the Velodyne LIDAR typically has a "blind area" of some meters around the vehicle not hit by any beam. Thus, part of the pavement edge might become invisible. When no explicit representation (besides the grid) of such a pavement edge is used, grid accumulation would be helpful. However, this accumulation should be done with care, as will become clearer when detailing how the grid values are computed. We first assume that every single LIDAR measurement of the last frame has instantly been transformed to a global Cartesian 3D coordinate system, taking into account the vehicle's own motion [exploiting inertial measurement unit (IMU) and odometric information]. This is done by simultaneously moving the coordinate system of the vehicle while transforming the local LIDAR measurements to global 3D space. After a frame is completed, all points are transformed back into the last local coordinate system of the vehicle, simulating a scan as if all measurements were taken at a single point of time instead of the 100-ms time period of one LIDAR revolution. Although integrating IMU and odometric data over a long period of time leads to large drifts, the drift can be neglected for the short time period of 100 ms. Each grid value is then computed to be the maximum difference in z coordinates of all points falling into the respective grid cell. We refer to this value as the *grid* or *occupancy value* throughout the text. To see the rationale behind this computation, consider an obstacle in front of the vehicle, as shown in Figure 3. Such a z difference does not necessarily need to emerge from different beams. Owing to the high horizontal resolution of the LIDAR, even a single beam might slice an obstacle and produce different z values of points that fall into the same grid cell (see Figure 4).

Note that by using z-coordinate differences, we get only a $2\frac{1}{2}$D model of the world. Although this model can capture both obstacles that are in contact with the ground as well as ones that are not, it cannot differentiate between both: it is missing absolute z coordinates. As a result of this limitation, the obstacle avoidance behavior will be conservative, e.g., the

[1]The algorithm was integrated in the 2007 DARPA Urban Challenge team AnnieWAY (Kammel, 2007)

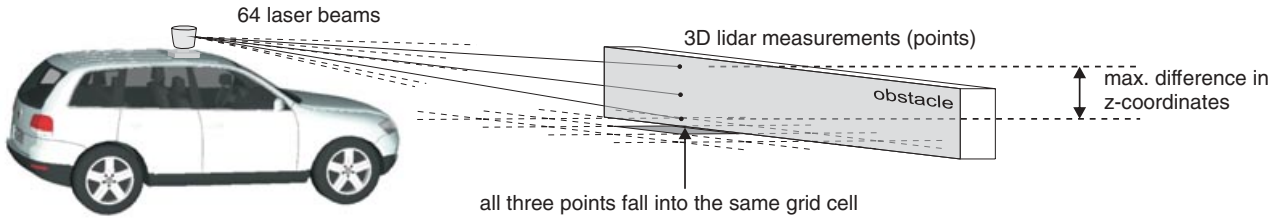


Figure 3. To show how the grid is computed, only 3 of the 64 laser beams of our sensor are shown as they hit an obstacle. The corresponding points in 3D space fall into the same grid cell. A grid value is computed to be the maximum difference of z coordinates of points in 3D space falling into the same grid cell.

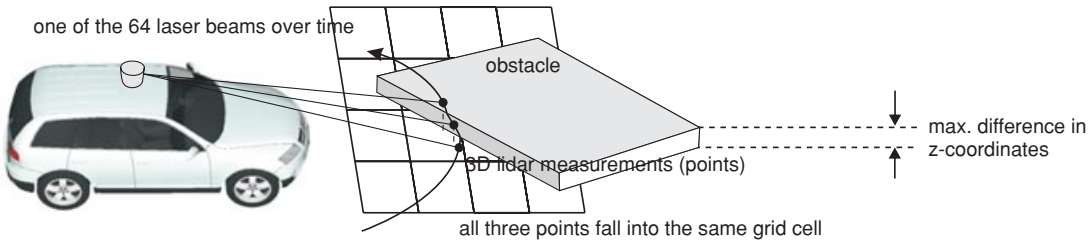


Figure 4. One of the 64 LIDAR beams slices an obstacle over time. Owing to the high horizontal resolution of the LIDAR, even a single beam might produce different z values of points falling into the same grid cell.

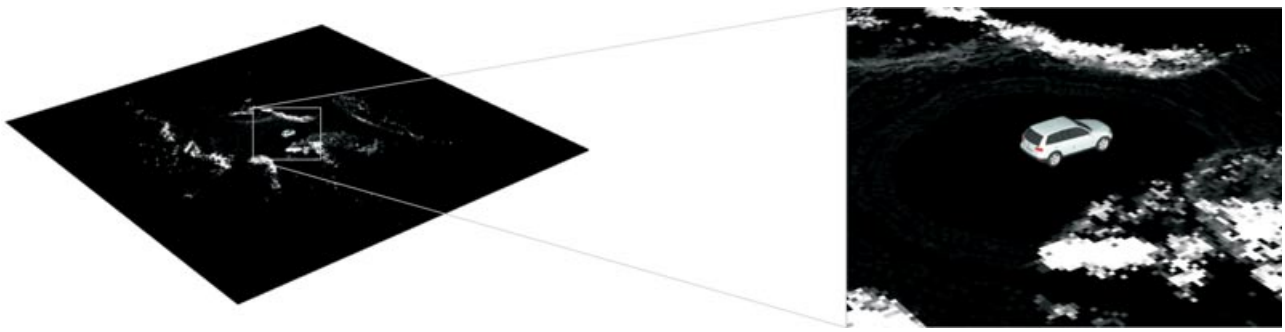


Figure 5. On the right-hand side, a cutout of the occupancy grid on the left is shown. The position of the vehicle in the grid is always in the center. The grid is "hard-mounted" to the vehicle. Cells with large z differences are shown white; smaller z differences are shown as gray values.

vehicle will try to avoid tree branches even if it could safely pass beneath them. Likewise, it will avoid obstacles such as barriers, as desired.

If our approach should be modified in the way that data accumulation is done, we recommend using points from only the same LIDAR turn when calculating the differences in z coordinates and accumulating the results rather than the raw 3D point data . Figure 4 shows an example of our grid (with no data accumulation).

3. TENTACLE STRUCTURE AND GENERATION

In our system we use 16 sets of tentacles. As shown in Figure 6, each of these "speed sets" contains 81 tentacles corresponding to a specific velocity of the vehicle. The speed sets cover a range of velocities from 0 to 10 m/s, with lower speeds being represented more frequently. All tentacles are represented in the local coordinate system of the vehicle. They start at the vehicle's center of gravity and take the shape of circular

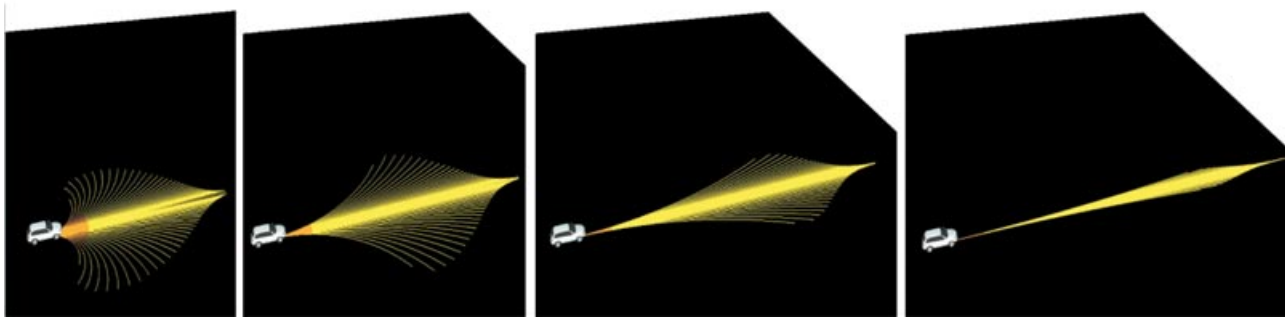


Figure 6. The range of speeds from 0 to 10 m/s is represented by 16 speed sets, each containing 81 tentacles. Only four of these sets are shown here. The tentacles are circular arcs and start at the center of gravity of the vehicle.

arcs. Each arc represents the trajectory corresponding to a specific steering angle. Higher curvatures are present at low speeds than at high speeds. The tentacle's lengths increase with higher speed sets, whereas within a set less curved tentacles are longer.

For a justification of the choice of circular arcs, consider that each tentacle will be executed for only 0.1 s and hence the overall resulting trajectory can be thought of as a concatenation of small circular fragments. For the above choice of a maximum speed of 10 m/s the maximum length of a fragment is 1 m. Hence, we can reasonably well approximate all possible clothoids the vehicle can drive.

3.1. Geometry

Our tentacles have the shape of circular arcs. They are used for both perception and motion execution. For perception, they span two areas, a classification and a support area that are used to probe the grid underneath the tentacle. For execution, an initial fragment of a selected tentacle is considered as a path to be driven for one LIDAR frame (0.1 s). Different execution options exist, with the simplest method just setting the appropriate steering angle corresponding to the selected tentacle, combined with a temporal filtering method to avoid all-too-sudden changes in curvatures. As we will see in Section 6, the execution modes will cause the vehicle to not precisely drive along the sequences of tentacles. Our rationale is not to care about the precise trajectory but just to ensure that the resulting path is within a corridor spanned by the tentacles. The trick is to make the tentacles broader than the vehicle, such that the maximum possible path deviation is included in the area that is evaluated to be free of obstacles.

Originally, our algorithm was conceived, implemented, and used at the C-Elrob and 2007 DARPA Urban Challenge without analyzing the vehicle dynamics. The design was put forth in an ad hoc manner and verified and refined by excessive experiments. Some design issues have a physical reasoning but with no real theoretic derivation that is founded on vehicle dynamics. Many of the parameters and formulas that define them were determined empirically. The justification for this is that all our choices are far behind the physical limits of the vehicle. And within this space of physical feasibility we exploit just the given freedom in design. In Section 6, we will provide a theoretic study of the vehicle dynamics, and we are able to theoretically justify our choices in the retrospective. The analysis also suggests some improvements; however, we first want to precisely describe our original ad hoc design, such that this paper is still a valid reference on the 2007 DARPA Urban Challenge implementation and the method can be exactly reproduced.

We briefly detail the geometry of the tentacles used: With $n = 16$ being the number of speed sets, the radius r_k of the kth tentacle in a set is given by

$$r_k = \begin{cases} \rho^k R_j & | \; k = 0, \ldots, 39 \\ \infty & | \; k = 40 \\ -\rho^{k-41} R_j & | \; k = 41, \ldots, 80 \end{cases}, \tag{1}$$

where the exponential factor $\rho = 1.15$ and the initial radius R_j of speed set $j = 0, \ldots 15$ is

$$R_j = \frac{l}{\Delta\phi(1 - q^{0.9})}, \tag{2}$$

and

$$l = 8\ \mathrm{m} + 33.5\ \mathrm{m}\, q^{1.2} \tag{3}$$

is the length of the outmost tentacles, with $q = j/(n-1)$ and $\Delta\phi = 1.2\ (\pi/2)$ being the angle subtended by the outmost tentacle of the lowest speed set (the most curved one). The length of the kth tentacle is given by

$$l_k = \begin{cases} l + 20\ \mathrm{m}\sqrt{\frac{k}{40}} \mid k = 0, \ldots, 40 \\ l + 20\ \mathrm{m}\sqrt{\frac{k-40}{40}} \mid k = 41, \ldots, 80 \end{cases}. \tag{4}$$

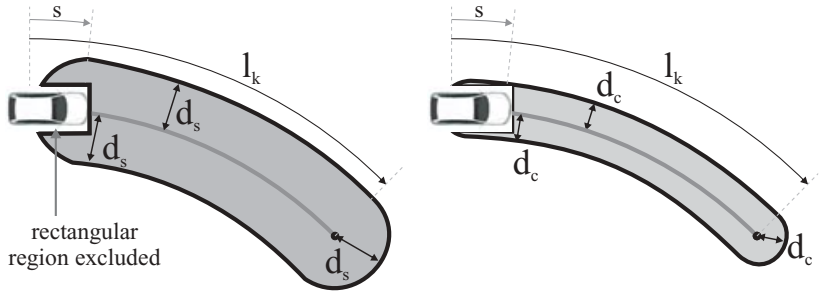
For the UC07, the velocity for speed set j was computed by

$$v_j = v_s + q^{1.2}(v_e - v_s), \tag{5}$$

where the speed of the lowest speed set is $v_s = 0.25$ m/s and the maximum speed is $v_e = 10$ m/s. This choice has no physical justification other than that the resulting curves are comfortably drivable with the specified speeds. The design was set up empirically and tested by experiments. The motivation for the exponential factor $q^{1.2}$ is to sample low speeds more frequently. Similarly, the reason for the exponential form in Eq. (1) is to have more tentacles with small curvatures and a coarser distribution at larger curvatures. The idea is that the tentacle approach should have more options for "fine-motor" primitives. However, there is no physical reason for this choice. A uniform distribution of radii would probably work, too. An interesting idea for the future is not to design the tentacles but to learn the distribution from a human driver by observing and segmenting the trajectories he drives. Equation (2) defines the base radius R_j of each speed set that is the radius of the outmost and most curved tentacle of speed set j. The tentacles that are directed straight need to have a certain length to ensure a sufficient look-ahead distance, depending on the speed. If all tentacles of a given speed set had this minimum length, the outmost tentacles would actually bend behind the vehicle. To avoid this, the outermost tentacles need to be shorter than the ones pointing straight. The reason that the straight tentacles should have a certain length is that we want to allow the selection mechanism to probe a larger portion of space, detecting obstacles far before they cause danger and require immediate braking. Hence, if we ignore the term $(1 - q^{0.9})$ for a moment, the initial radius R_j is calculated such that the angular scope of the tentacle arc is $\Delta\Phi$, slightly more than 90 deg for the slowest speed set. This means that at very low speeds the vehicle can look around, e.g., a road branch, slightly more than 90 deg. The term $(1 - q^{0.9})$ lets the radii of the outermost tentacles increase with successive speed sets. One could argue that it would have been better to define the outermost radii according to an underlying physical property, e.g., limiting the centripetal force at a given speed. However, although such a definition sounds more sophisticated at first, it is not pure physics that defines the proper structure. Of course, for safety reasons, we want to stay far beyond the physical limits of the vehicle. But within those bounds, physics might not be the desired property. Restricting or allowing curvatures might also depend on the environment. For instance, when driving on highways, one might want to restrict curvatures far beyond the physical limits, knowing that highways are not built to have those high curvatures. Hence, the design of the tentacles has an ecological perspective, too. According to our design, the lengths of the tentacles then increase from outer to inner tentacles and from lower to higher speed sets. The given design lets the center tentacle of the highest speed set together with its classification and support areas exactly reach the far end of the occupancy grid (see Figure 6).

3.2. Support and Classification Area

When a tentacle is evaluated, occupancy cells within a radius d_s to any point on the tentacle are accessed. Those cells form the *support area* of the tentacle. A subset of those cells form the *classification area* comprising cells within a radius $d_c < d_s$ to any point on the tentacle. The geometric definition of these areas is illustrated in Figure 7. Whereas the classification area is later used to determine tentacles that are drivable in principle, the support area is used to determine the "best" of all drivable tentacles. For low speeds the classification area is only slightly wider than the vehicle, allowing driving along narrow roads or through narrow gates. For higher speeds, the width of the area increases. The reason for this is that the additional width has to ensure that the trajectory that results from executing the tentacle is within the corridor defined by the classification area. This is because when executing a tentacle, the vehicle will not precisely follow its shape. This is no problem as long as we can guarantee that the resulting path is within the



(a) Support area (b) Classification area

Figure 7. (a) The support area covers all cells within a distance d_s of the tentacle (start point shifted about s). The cells of the support area are subject to different weights as explained in the main text. (b) The classification area is a subset of the support area covering all cells within a distance $d_c < d_s$ of the tentacle (start point again shifted about s). No weights are specified for the classification area. A rectangular region around the vehicle (no LIDAR measurements are made in this area) is excluded from both the support and the classification areas.

classification area and hence free of obstacles. A theoretical investigation on this issue will be conducted in Section 6. In contrast, the support area is much wider than the vehicle. This allows the algorithm to be parameterized in such a way that, e.g., the vehicle exploits free space and drives in the center of a wide path instead of driving always close to the border of the path. This behavior might not be desired in some scenarios and can be disabled as explained in later sections. The distances d_s and d_c increase with the speed of the vehicle (speed sets), such that the vehicle would take a larger lateral security distance when driving around obstacles at high speeds or declaring a narrow gate as impassable at high speeds while classifying it as drivable at low speeds.

An important point is that the geometric relation between a specific tentacle and the occupancy grid cells remains always the same because the grid is specified in the coordinate system of the vehicle. Thus, the cell offset $o = \Delta x \cdot y + x$ ($\Delta x = 512$, the width of the occupancy grid, x, y being cell indices) of the cells (x, y) indexed by a tentacle's support and classification area remains always the same. This static geometric relation allows precomputation of all cell offsets for each tentacle and storage of the n_s cells of the support area as the set $\mathcal{A} = \{\mathbf{c}_0, \ldots, \mathbf{c}_{n_s-1}\}$, with

$$\mathbf{c}_i := (o_i, w_i, k_i, f_i). \tag{6}$$

Here, o_i is the offset of the cell, uniquely specifying its position in the grid as described above and allowing fast memory access later. The value w_i is a weight and k_i is a histogram index (both to be detailed later) associated with the respective cell of the support area. The flags f_i are set such that the subset $A_c \subset A$ describing the classification area is given by $A_c = \{\mathbf{c}_i \in A | f_i = 1\}$. As illustrated in Figure 8 the weight w_i of a cell $\mathbf{c}_i$ is computed by passing the cell's distance d_i to the tentacle (orthogonal distance except at the start and end point of the tentacle) as argument to the cross-profile function

$$p(d) = \begin{cases} w_{\max} & | \; d \le d_c \\ \dfrac{w_{\max}}{\kappa + \dfrac{d - d_c}{\sigma}} & | \; d > d_c \end{cases}, \tag{7}$$

where $\kappa = 1$, $\sigma = 0.16$, and $w_{\max} = 10$ in our implementation. For the C-Elrob 2007 and the 2007 DARPA Urban Challenge the value d_c was empirically specified for each speed set j with corresponding velocity v by

$$d_c = \begin{cases} 1.7\,\text{m} + 0.2\,\text{m}\dfrac{v}{3\,\text{m/s}} & | \; v < 3\,\text{m/s} \\ 1.9\,\text{m} + 0.6\,\text{m}\dfrac{(v - 3\,\text{m/s})}{10\,\text{m/s}} & | \; 3\,\text{m/s} < v \le 10\,\text{m/s} \end{cases}. \tag{8}$$

A weakness of this design is that it is not theoretically but only empirically justified. However, we will provide a theoretical analysis in Section 6 showing that our choice was above a required minimum width that can theoretically be justified considering the dynamics of the vehicle.

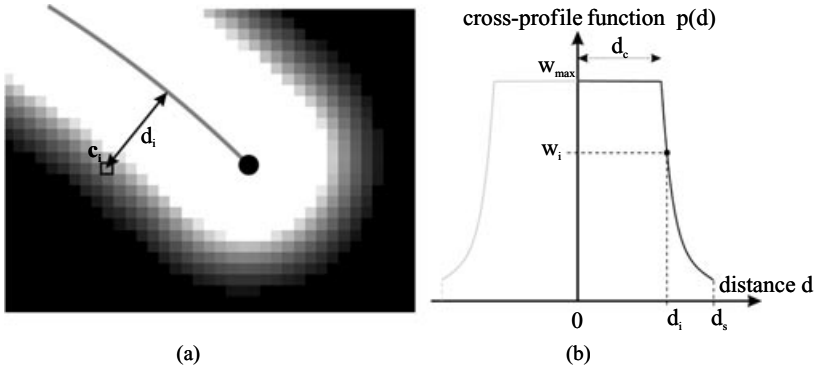


Figure 8. The weight w_i of a cell $\mathbf{c}_i$ is calculated by passing its distance d_i to the tentacle (a) to the cross-profile function $p(d)$ (b) [Eq. (7)].

3.3. Longitudinal Histogram

We aim to binary classify all tentacles whether they are drivable or not. In case they are occupied, we also wish to compute the distance to the first obstacle along the tentacle. For these calculations, we define a histogram for each tentacle with its bins aligned along the curve of the tentacle, discretizing its length into a sequence of n_h bins $0, \ldots, n_h - 1$ ($n_h = 200$ in our implementation). To determine the cells that contribute to a bins value, we orthogonally project every cell onto the tentacle retrieving the cell's histogram index k_i. To speed up later histogram calculations, all k_i are precomputed (see Figure 9). Here, k_i is the second to last value in Eq. (6).

4. SELECTION MECHANISM

With the description of the tentacle-related data structures and their computation being completed, in this section we show how a tentacle is selected with each new LIDAR frame (10 Hz). This tentacle is then used to derive the final trajectory to drive. To decide on

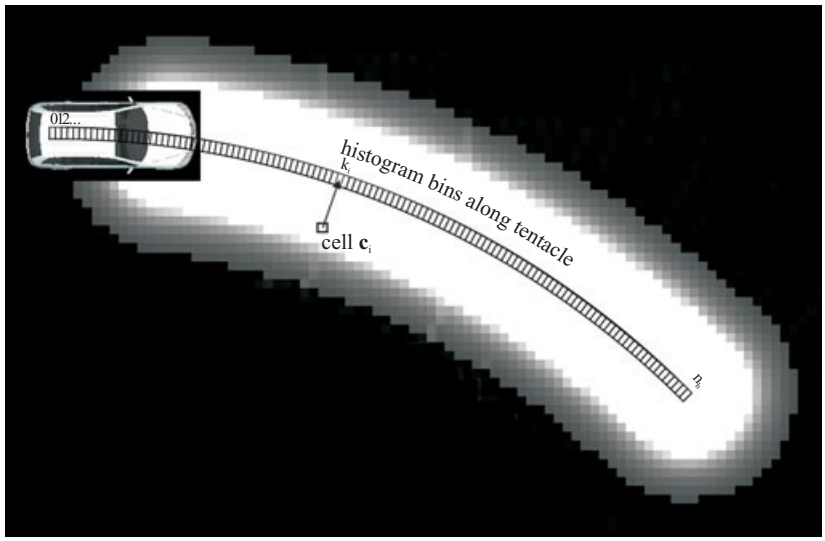


Figure 9. Each tentacle has a longitudinal histogram with its bins aligned along the tentacle, discretizing the tentacle's length. Grid cells are projected orthogonally onto the tentacle to compute the histogram index k_i of a cell $\mathbf{c}_i$ [see Eq. (6)].

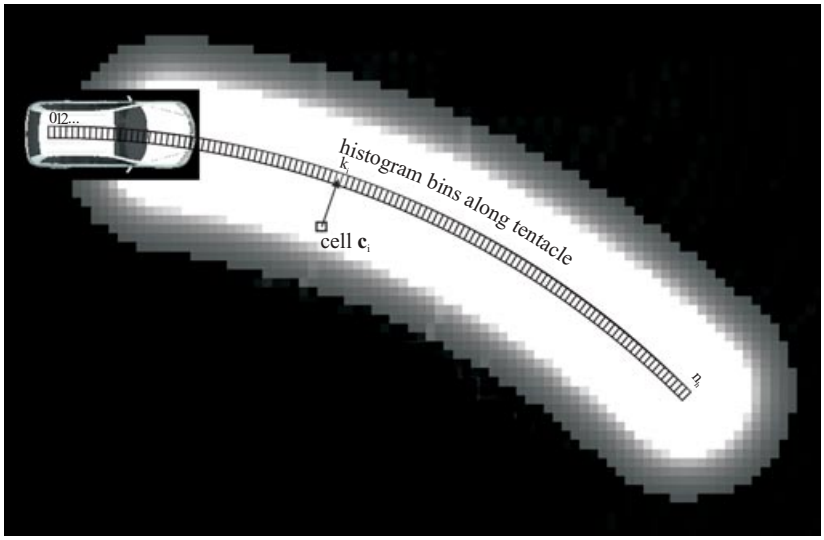


Figure 10. A sliding window is used to determine the position of the first obstacle. The window is initially placed at bin 0 and successively slid to higher bin indices. If the sum of bin values within this window exceeds a threshold n_o ($n_o = 1$ in our experiments), an obstacle is detected and the position of the sliding window yields the distance l_o to this first obstacle.

the "best" tentacle, first all drivable tentacles are determined. For these valid tentacles, three values are calculated (clearance, flatness, and trajectory value). Later, they will be linearly combined to derive a single decision value, which is minimized. Our description continues detailing the classification step and the three aforementioned decision affectors. We assume that a new cycle has started and the occupancy grid has been updated with the latest scan.

4.1. Tentacle Classification

Classifying a tentacle as to whether it is drivable happens at the same time as determining the distance to the first obstacle (if any) along that tentacle. The longitudinal histogram provided with each tentacle is used for this purpose: Initially, the bins are all cleared. Then, all cells $\mathbf{c}_i = (o_i, w_i, k_i, f_i) \in A_c$ of the classification area are used to access the occupancy grid with memory offset o_i, obtaining the occupancy value $v_i = g(o_i)$ at the cell's location. Here, $g(o)$ is the function that returns the z difference of the grid cell at location o. Owing to the wheel radius of our vehicle, we are interested in obstacles greater than 0.1 m. Thus, if v_i exceeds a threshold of $t_c = 0.1$ m, the histogram bin k_i is incremented (see Figure 9). Only the classification area and not the support area is used for this step, allowing the vehicle to drive through narrow areas only slightly wider than the vehicle. Also, no weighting takes place in this step, because an obstacle in the classification area will cause damage to the vehicle independent of its lateral position within the classification area. As illustrated in Figure 10, an obstacle is detected if the sum of all bins within a sliding window (of n_w bins) exceeds a threshold of n_o (in our implementation $n_w = 5$, $n_o = 2$, and the total number of bins is $n_h = 200$). If no obstacle is detected, the tentacle is classified as drivable. However, a peculiarity of our approach is that if an obstacle is detected, the tentacle is classified as undrivable solely if the distance to this obstacle is below a so-called *crash distance* l_c. Roughly speaking, if the obstacle is distant, the tentacle is still drivable for some time. Here, the crash distance is the distance the vehicle needs to stop using a constant convenient deceleration a plus a security distance l_s. It depends on the speed v of the vehicle and is calculated by

$$l_c = l_s + \frac{v^2}{2a}. \tag{9}$$

In our implementation, $l_s = 6$ m and $a = 1.5$ m/s^2. Summarizing, a tentacle is classified as nondrivable only if an obstacle is within a distance l_c along

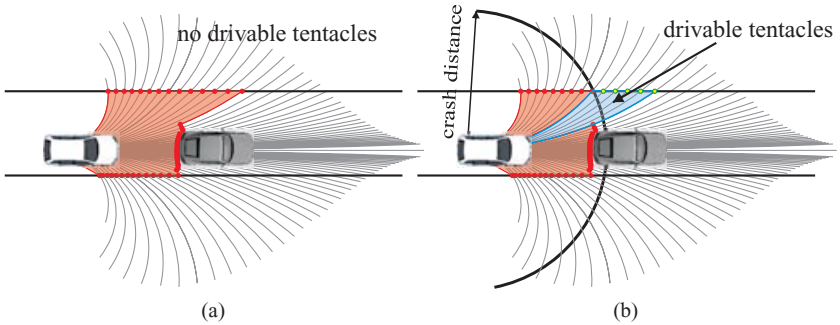


Figure 11. The case in which a car is blocking the right lane of a road and only a narrow passage is left to pass the car. The red points mark the locations along the tentacles where the vehicle would hit either the car or the road border. As can be seen, no tentacle is free of obstacles. Hence, by neglecting the distance to an obstacle, all tentacles would be classified undrivable (a). In contrast, panel b shows the concept of classifying tentacles as undrivable only in case of being occupied within a speed-dependent crash distance (see main text). In this case, some drivable tentacles remain, allowing a pass of the car.

that tentacle. Although this mechanism might seem like an optional gadget, it is actually very important. To see why, consider the scenario illustrated in Figure 11(a). Here, a car is blocking the lane, leaving only a narrow passage to drive around the car without hitting either the car or the pavement edges of the road. As can be seen, the geometry of the scenario renders all tentacles occupied by either the car or the road side, and no tentacle is completely free. Hence, an approach that would classify all occupied tentacles as nondrivable would not be able to drive around the car. This is the reason that similar approaches such as those of Thrun et al. (2006) and Coombs et al. (2000) require geometries different from arcs. However, when introducing the concept of classifying tentacles as "undrivable" only if an obstacle is within the crash distance, the case can be handled as illustrated in Figure 11(b). One might argue that introducing the crash distance is just the same as defining shorter tentacles. However, this is not the case, because tentacles with a more distant obstacle are still preferred. This mechanism will be detailed in the next sections, where we describe the calculation of three decision-affecting values. Those values will be calculated only for tentacles that were classified as drivable, and only these constitute the set the "best" tentacle is selected from.

4.2. Braking and Crash Distance

The influence of the crash distance l_c [see Eq. (9)] on classifying a tentacle as being drivable and its interplay with a simple braking mechanism is the main reason that our approach accomplishes using only 81 arcs but still can drive along narrow roads and avoid obstacles in difficult situations. The crash distance can be seen as cutting the evaluation length (only for the purpose of classification) of the tentacle before its end. This mechanism has to be seen in combination with the simple braking mechanism that acts as follows.

If no tentacle is drivable, the tentacle with the largest distance to the first obstacle is selected and the vehicle chooses this tentacle for braking. The vehicle then brakes with a constant deceleration along this tentacle. This execution of this *brake tentacle* is performed only for a period of 0.1 s, and then all tentacles are evaluated anew at the next time step.

Now consider the case that the vehicle is entering a narrow way with a speed set and crash distance that has no tentacle that is drivable, just because the shape of the narrow way does not correspond to any of the tentacles. Hence, the vehicle will brake and the velocity and crash distance reduce. However, reducing the crash distance lets some obstacles that were before the old crash distance now be behind the new crash distance, "freeing" some tentacles. In the extreme, the vehicle brakes until it almost stops. However, at low speeds the crash distance is very small, meaning that even quite close obstacles along a tentacle are not considered as making the tentacle undrivable for one time step, simply because there is enough space left in order to postpone braking to a later time. Hence, if a way is narrow, the vehicle typically just

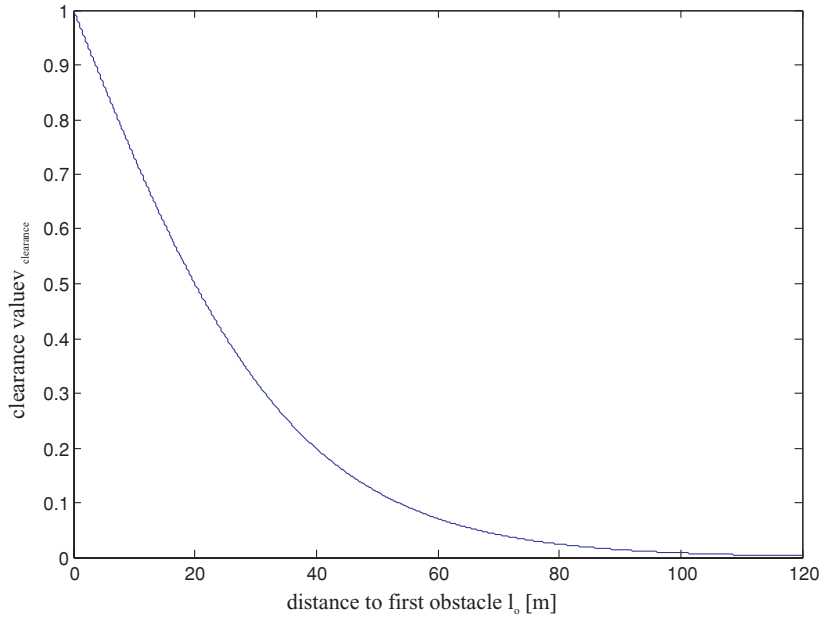


Figure 12. Plot of $v_{\text{clearance}}(l_o)$ [see Eq. (10) with $c_{\text{clearance}}$ as in Eq. (11)] shows that the greater the distance to the first obstacle, the smaller is the change in the clearance value and the change of impact in the overall tentacle selection process.

slows down, until the low crash distance allows selection of tentacles that do not match the exact shape. This mismatch in shape does not constitute a problem, because only a small fragment of the tentacles is executed before a new curvature is selected at the next time step.

4.3. Clearance Value

The *clearance value* is the first of three decision-affecting values computed for each drivable tentacle. To make these values comparable, they are normalized to the range $[0, \ldots, 1]$, where a value of 0 designates a preference for such a tentacle. The calculation of the clearance value directly uses the distance to the first obstacle l_o calculated in the classification step. It expresses how far the vehicle could drive along a tentacle before hitting an obstacle. It is calculated by the sigmoid-like function

$$v_{\text{clearance}}(l_o) = \begin{cases} 0 & \mid \text{if the tentacle is entirely free} \\ 2 - \dfrac{2}{1 + e^{-c_{\text{clearance}} \cdot l_o}} & \mid \text{otherwise} \end{cases} . \quad (10)$$

where the constant $c_{\text{clearance}}$ is calculated by Eq. (11) to yield $v_{\text{clearance}}(l_{0.5}) = 0.5$ at a distance $l_{0.5} = 20$ m in our implementation:

$$c_{\text{clearance}} = \frac{\ln 1/3}{-l_{0.5}}. \quad (11)$$

As shown by the plot of $v_{\text{clearance}}(l_o)$ in Figure 12, the clearance value converges against zero for $l_o \longrightarrow \infty$. The greater the distance to the first obstacle, the smaller is the change in the clearance value and the change of impact in the tentacle selection process. The clearance value is part of a linear combination (with positive coefficients) of three values that is minimized later, and thus, tentacles with a large distance to the next obstacle are preferred.

4.4. Flatness Value

The *flatness value* has the goal to prefer tentacles leading over smooth terrain. Although all tentacles passing the classification step are drivable without causing damage to the vehicle, it might still be desired to prefer a smooth path. The second purpose of the flatness value is to exploit free space, if possible: For instance, if there is a broad dirt road without any

obstacle, the flatness value lets the vehicle drive in the center of that path, instead of always close to the border of that path. It is computed accessing the whole support area of a tentacle by

$$v_{\text{flatness}} = \frac{2}{1 + e^{-c_{\text{flatness}} \cdot v_{\text{avg}}}} - 1, \tag{12}$$

where

$$v_{\text{avg}} = \frac{\sum_{\mathbf{c}=(o,w,k,f)\in\mathcal{A}} w g(o)}{\sum_{\mathbf{c}=(o,w,k,f)\in\mathcal{A}} w} \tag{13}$$

and c_{flatness} is computed in analogy to Eq. (11) such that v_{flatness} reaches a value of 0.5 at $v_{\text{avg}} = 0.3$ in our implementation. Set $\mathcal{A}$ describes the precomputed support area as described in Section 3.2, and $g(o)$ is the value of the occupancy grid at location o.

4.5. Trajectory Value

Whereas the latter two values aim at obstacle avoidance in an unknown environment, the *trajectory value* pushes the vehicle toward following a given trajectory, e.g., defined by GPS way points. For each tentacle, a quality of how much the tentacle follows or leads to a given trajectory is calculated. Different distance measures, such as the Hausdorff or Frechet distance (Preparata & Shanmos, 1985), can be considered. However, the simplest method is to consider a single point on the tentacle and a corresponding point on the trajectory. The point on the tentacle is taken at the crash distance l_c, such that at high speeds the point is more distant than at low speeds. If the steering angles of the tentacles are executed directly while following a given set of GPS way points, and ignoring the flatness and clearance contributions, the approach is very similar to "Pure Pursuit" (Coulter, 1992), which calculates an optimal circular arc to reach a GPS way point from the current robot's position and sets the steering angle to the curvature of the calculated arc. The corresponding point is calculated by first matching the robot on the trajectory as shown in Figure 13 and then sampling a point from the trajectory located at a distance l_c from the matched position. The effect of sampling the points at the crash distance is that the vehicle tries to follow the trajectory more closely at low speeds, whereas at high speeds the vehicle tries to recover a lost track farther on.

For each tentacle the distance measure v_{dist} and finally the trajectory value $v_{\text{trajectory}}$ are calculated by

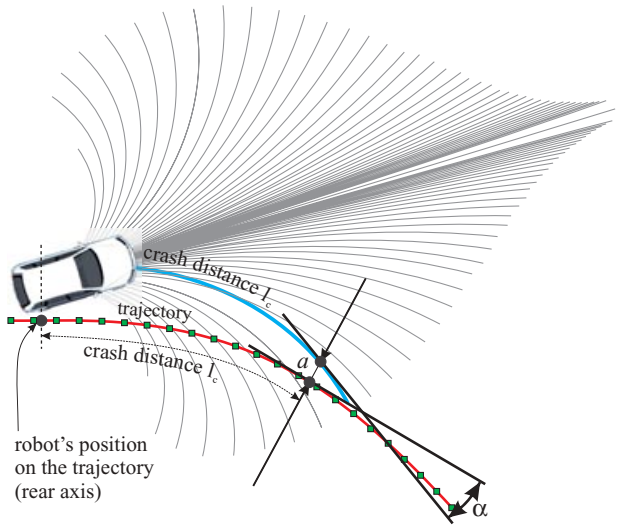


Figure 13. For each tentacle a trajectory value is computed by considering the distance and tangent orientations of two corresponding points, one on the tentacle and the other on the (GPS) trajectory to be followed.

taking both the distance a between the point on the tentacle and its corresponding point on the trajectory as well as its relative tangent orientations α into account:

$$v_{\text{dist}} = a + c_\alpha \alpha, \tag{14}$$

$$v_{\text{trajectory}} = \frac{v_{\text{dist}} - v_{\text{min}}}{v_{\text{max}} - v_{\text{min}}}. \tag{15}$$

Here, $c_\alpha = 3.0\,\text{m/rad}$ is a constant, a and α are illustrated in Figure 13, and $v_{\text{max}}, v_{\text{min}}$ are the maximum/minimum values of v_{dist} over all tentacles of the current speed set. Equation (15) produces normalized values within the range of $[0, \ldots, 1]$. Compared to the normalization procedures of the other two values, v_{flatness} and $v_{\text{clearance}}$, the normalization process is not independent of the v_{dist} values of the other tentacles here. The reason is that if the vehicle were distant from the trajectory (say, 20 m), Eq. (15) would produce quite high values. When these values were normalized using a sigmoid-like function (as is the case for v_{flatness} and $v_{\text{clearance}}$), all tentacles received similar values for $v_{\text{trajectory}}$, such that when later combined with the flatness and clearance values, the geometric variation of the tentacles with respect to the trajectory would have little influence in the final decision. However, in our case Eq. (15) produces normalized

values that reflect the tentacles' geometric variation independent of the vehicle's gross distance from the trajectory.

4.6. Combining Clearance, Flatness, and Trajectory Values

For each tentacle classified as *drivable*, the three values $v_{\text{clearance}}$, v_{flatness}, and $v_{\text{trajectory}}$ are within the range $0, \ldots, 1$. They are now linearly combined into a single value:

$$v_{\text{combined}} = a_0 v_{\text{clearance}} + a_1 v_{\text{flatness}} + a_2 v_{\text{trajectory}}. \quad (16)$$

Here, a_0, a_1, and a_2 are parameters that can be used to change the behavior of our approach at a gross level. For instance, at the C-Elrob 2007 we used $a_0 = 0$, $a_1 = 1$, and $a_2 = 0$, with the result that the vehicle chose to freely drive over flat area without following a given trajectory. In contrast, we used $a_0 = 1$, $a_1 = 0$, and $a_2 = 0.5$ for the 2007 DARPA Urban Challenge, letting the vehicle follow a given trajectory while avoiding obstacles at the same time. Note that the primary obstacle avoidance mechanism is not accomplished by the three values $v_{\text{clearance}}$, v_{flatness}, and $v_{\text{trajectory}}$ but by the classification step: Only drivable tentacles constitute the set for which v_{combined} is calculated. If no drivable tentacles exist, the tentacle with the largest clearance value is selected, and the vehicle is commanded to brake along this tentacle.

The linear combination of Eq. (16) means that different contributions can balance each other if the respective a_i values are nonzero. For instance, consider the case of driving along a dirt road with GPS having a drift of 2 m such that following the GPS path would cause the vehicle to drive at the very boundary of the dirt road. Assume that the wheels at one side of the vehicle constantly drive slightly aside of the road. If there was a pavement edge at this side, the respective tentacles would have been classified as nondrivable and the vehicle would not even have driven into this situation. But let us assume that the obstacles are just stones, with their size being at the limit of allowing to drive over them. Hence, let us assume that the tentacles classify the respective areas as being drivable. Now, the contributions in Eq. (16) can be seen as forces competing with each other. The trajectory would let the vehicle neglect the small stones and just drive along the drifted GPS trajectory. The clearance value is difficult to predict and depends on the situation. For instance, if within the area next to the road no high obstacles exist, the clearance value would not prevent the vehicle from continuing to drive aside of the road. In contrast, the flatness value gives the vehicle a tendency to prefer smooth areas. In this way, it acts as a force typically pulling the vehicle onto the road, because in most cases the road is the flattest area. By tuning a_0, a_1, and a_2, different behaviors can be produced. The fine tuning, however, occurs by the parameters of the individual functions $v_{\text{clearance}}$, v_{flatness}, and $v_{\text{trajectory}}$. For the UC07, we first tuned our system by experiments testing that our vehicle would drive along narrow passages in the presence of GPS drifts (just simulating them by adding an artificial drift). The important parameter for this is the crash distance l_c [see Eq. (9)]. Also, we performed various other experiments as described in the experimental results section in order to tune the system. For the UC07, we decided to ignore flatness and drive along GPS if drivable, even when driving over uneven terrain such as smaller rocks. However, this was not a decision without dissenting votes.

To avoid inconsistent selections of tentacles at successive time steps, we use the following hysteresis mechanism: We first determine the set S of drivable tentacles with a combined value v_{combined} that is at most $\epsilon = 0.00001$ worse than the value v_m of the best tentacle. From this set of "approximately equally good" tentacles we then finally select the one that is geometrically most similar to the tentacle selected in the last time step. Here, our similarity measure is simply the absolute difference of the tentacles' curvatures. Hence, in ambiguous situations, the one tentacle is selected that is most consistent with the last decision.

5. TENTACLE EXECUTION

There are various options for steering the vehicle according to the selected tentacle.

5.1. Direct Execution

By the fact that each tentacle corresponds to a circular arc and that the speed of the vehicle is given, a respective steering angle δ_F can be calculated. It is then possible to directly command this steering angle to a low-level controller at the frequency of the tentacle selection process (10 Hz). However, because the tentacles represent discrete curvatures, the resulting driving behavior might be jerky. Hence, it might be desired to generate smoothed steering angles $\delta_s(k)$,

e.g., by the simple recursive filter

$$\delta_s(k) = \kappa\delta_F(k) + (1-\kappa)\delta_s(k-1), \tag{17}$$

where $\delta_F(k)$ is the steering angle of the current selected tentacle, $\delta_s(k-1)$ is the last smoothed curvature, and $0 \leq \kappa \leq 1$ is a constant. However, such a filter introduces a temporal delay until a constantly sensed curvature takes effect. At the C-Elrob 2007 we used this simple filtering method with a value of $\kappa = 0.9$. At the 2007 Urban Challenge a different execution mode was used.

We also experimented with a more sophisticated approach using a clothoid model for the road, a bicycle model for the vehicle dynamics, and additional tentacles with different lateral displacements and yaw angles. Then selecting a tentacle can be seen as measuring a curvature, a displacement, and a yaw angle, and a Kalman filtering approach such as that of Dickmanns (1994, 2007) can be used to produce a model-based filtered estimate of the curvature, displacement, and yaw angle. However, this is at the cost of using many more tentacles and is either error prone in the case of discrepancies between the road model and the real shape of the environment or requires an explicit recognition of different road models (e.g., at crossings).

5.2. Fragment Execution

Another option for executing tentacles is to consider the currently selected tentacle as a trajectory to execute. In this case, the vehicle is controlled by a path controller that derives the steering commands from the geometric relationship between the vehicle's pose in some Cartesian space and the trajectory specified in the same coordinate system. For instance, when using GPS, a universal transverse mercator (UTM)-coordinate space could be used, and in this case the selected tentacle has to be transformed in that space. Note that the tentacles are originally specified in the vehicle-centered coordinate system. When no GPS is used, a global but drifting position can still be computed by integrating odometric and inertial measurements. The drift can be neglected for tentacle execution because a tentacle is used only for a small period of time in this *drift space* (0.1 s). The advantage of this method is that trajectory execution can be controlled at a higher frequency, allowing compensation of local disturbances using IMU information. However, as shown in Figure 14, the sequence of tentacles does not necessarily yield a continuous trajectory over time. Deviations from the true trajectory will occur due to effects of time delay, the mass of the vehicle, and imprecisions of the low-level controller for the steering angle. Because the new tentacle starts from

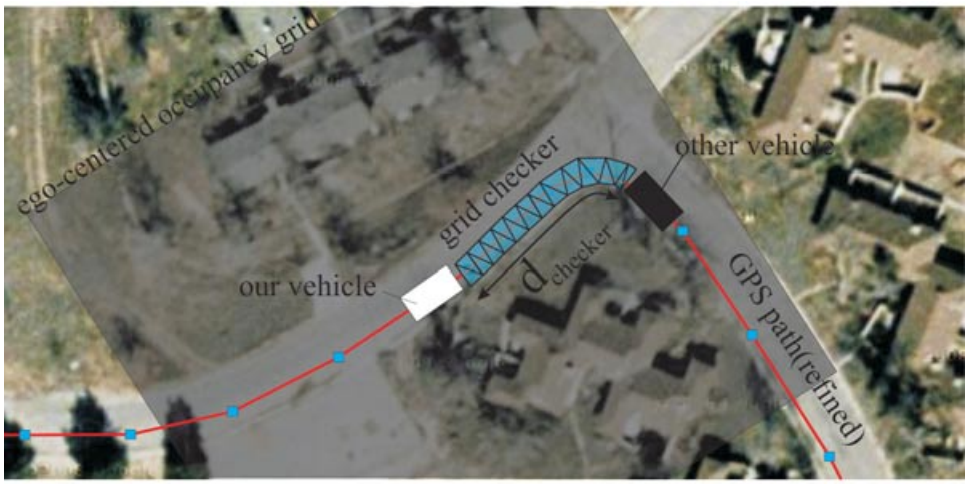


Figure 14. The sequence of selected tentacles typically does not yield a continuous trajectory, because only a small fraction of a tentacle is actually driven before a new tentacle is selected. (The fractions are shorter than shown in this figure. They have been exaggerated here for the sake of clarity.) When a new tentacle is selected, it starts at the vehicle's center of gravity again, not continuing the old tentacle in case of control deviations.

the new position again, it will start with an offset to the old tentacle not continuing the old trajectory.

5.3. Trajectory Blending

The fact that the sequence of selected tentacles does not form a continuous trajectory can conflict with the underlying path execution controller. Assume, for instance, that the path execution controller uses the lateral displacement of the vehicle with respect to the trajectory as one of its feedback values and that due to some imprecisions, a lateral displacement exists just before a new tentacle is selected. Then suddenly the displacement jumps to a value of zero, because the new tentacle starts with no displacement to the vehicle. If the path execution controller includes an integral term, this might cause undesired effects, because the controller aims at correcting a deviation that is suddenly reset to zero.

To avoid this discontinuity, we do not directly execute a selected tentacle but calculate a trajectory that continues the old trajectory and blends over to the end of the newly selected tentacle as shown in Figure 15.

Let the old trajectory be parameterized as

$$\mathbf{p}_{\text{old}} : [0, \ldots, 1] \longrightarrow I\!R^2, s \mapsto \mathbf{p}_{\text{old}}(s), \qquad (18)$$

such that $\mathbf{p}_{\text{old}}(0)$ is the vehicle's reference point on the trajectory as used by the path execution controller and $\mathbf{p}_{\text{old}}(1)$ is the end point of the old trajectory (see Figure 15). Let the trajectory of the current selected tentacle be parameterized as

$$\mathbf{p}_{\text{tentacle}} : [0, \ldots, 1] \longrightarrow I\!R^2, s \mapsto \mathbf{p}_{\text{tentacle}}(s), \qquad (19)$$

where $\mathbf{p}_{\text{tentacle}}(0)$ is the start and $\mathbf{p}_{\text{tentacle}}(1)$ is the end point of the tentacle. Then the blended trajectory is calculated by

$$\mathbf{p}_{\text{blend}}(s) = \beta(s)\mathbf{p}_{\text{old}}(s) + [1 - \beta(s)]\mathbf{p}_{\text{tentacle}}(s), \qquad (20)$$

where the blend function $\beta(s)$ is a monotonically increasing bijective mapping:

$$\beta : [0, \ldots, 1] \longrightarrow [0, \ldots, 1]. \qquad (21)$$

In our implementation we used the trivial blend function $\beta(s) = s$. Finally, we pass the blended trajectory to the path execution controller. In the next iteration, this trajectory serves as input $\mathbf{p}_{\text{old}}$ for the next blending step.

6. CONSIDERING VEHICLE DYNAMICS

At each time step our method uses the current velocity of the vehicle to choose the closest existing speed set, selects a tentacle from this set, and executes the tentacle. In the preceding section, we described different options for this execution. At the C-Elrob 2007

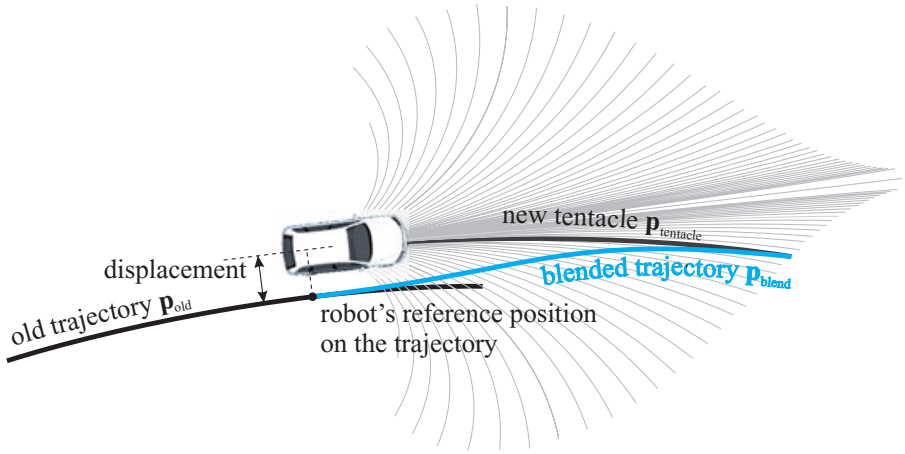


Figure 15. To avoid discontinuities in path execution when selecting a new tentacle, the vehicle does not directly execute the new tentacle but executes a blended trajectory continuing the old trajectory at the vehicle's reference pose and blending over to the newly selected tentacle. The reference pose is the position used by the path execution module to calculate its feedback values for control (e.g., displacement and yaw angle).

we used the direct method and at the Urban Challenge we used fragment execution, that is, we commanded the tentacles as trajectories to the low-level path following module. This decision to use the paths was due only to being compatible with the already existing low-level interface to the vehicle. For our dynamic analysis we will stick to the "direct execution" mode. This mode is simpler to analyze, because no control loop for path following has to be included in the analysis. Later, the question will be of how or whether our results are valid for the other execution modes. In this section we will see that our method commits various errors when considering a tentacle as a piece of trajectory that has to be precisely followed. However, this is not a required goal in our approach and hence the term "error" is inappropriate. In our approach, the final goal is to avoid obstacles, and we allow a deviation of the resulting trajectory from the tentacle's trajectory. This is possible because we can show that the deviations are small enough to be within a corridor (corresponding to the classification area of the tentacle) that is ensured to be free of obstacle. Before we detail this reasoning, it is first necessary to understand the different effects that let the vehicle's final trajectory deviate from the centerline of the tentacle. To understand and predict those effects, we have to consider vehicle dynamics.

6.1. Motion Equations

In this section we derive the motion equations to model and predict the dynamics of our vehicle. Although it might seem to be overambitious to derive those equations, in doing so, the variables and symbols we are using are clearly defined and well illustrated from the very beginning. In this way our argumentation is self-contained and comprehensible without having to refer to other literature. Also, our approach differs from the classical formulation (Mitschke & Wallentowitz, 2004) in that we do not make many of the linearization simplifications, but numerically integrate over the nonlinear differential equations. Our consideration of dynamics is limited to four-wheeled vehicles steered by two front wheels. Not our principal approach but rather our theoretical analysis in this paper is restricted to these types of vehicles. We consider a simplified model of such a vehicle assuming that the center of gravity lies in the ground plane. In doing so the centrifugal force that acts on the center of gravity does not change the load on the wheels. This allows us to reduce the precise geometric layout of the four wheels to the well-known "bicycle model" (Mitschke & Wallentowitz, 2004). Using this model, various assumptions are made, e.g., ignoring roll motions and assuming a uniform load onto inner and outer wheels. For a full enumeration of assumptions see Mitschke and Wallentowitz (2004). There exists a well-known, closed-form solution of the differential equations of the linearized version of the bicycle model (also known as "linearized bicycle model"). However, these equations are valid only for small steering angles and constant velocities and exhibit numerical instabilities for low velocities. Because some of our tentacles require high steering angles and because we want to be able to consider velocity changes, we do not make most of the linearization assumptions of the closed-form solution but derive our own set of nonlinear coupled differential equations in the following. For our analysis, we will later solve these equations by numerical integration.

As can be seen in Figure 16(a) the velocity $\mathbf{v} = \mathbf{v}_{CP}$ of the vehicle's center of gravity CG is tangent to the trajectory, the same being valid for the tangential acceleration $\dot{\mathbf{v}}$. In contrast, the centripetal acceleration v^2/ρ is directed to the center of curvature M. The distance ρ is the radius of curvature. The angle between $\mathbf{v}$ and the heading direction (direction of the centerline) is the sideslip angle β. The yaw angle Ψ is the angle of the heading direction measured against the global x axis x_0. The course angle of the vehicle is $\nu = \beta + \Psi$.

Figure 16(b) shows the forces acting on the vehicle. The longitudinal forces F_{xF} and F_{xR} are heading along the direction of the front F and rear R wheels. The lateral force of air A, F_{Ay} in the case of side wind, acts on the pressure point PP, its distance to the center of gravity being denoted with e_{CG}. The air resistance is expressed in terms of the force F_{Ax}.

Using the mass of the vehicle m, the moment of inertia around the z axis J_z, and the steering angle δ_F, we get the following equilibrium of forces in the longitudinal direction of the vehicle:

$$m(v^2/\rho)\sin\beta - m\dot{v}\cos\beta + F_{xR} - F_{Ax} + F_{xF}\cos\delta_F - F_{yF}\sin\delta_F = 0, \tag{22}$$

and in the lateral direction

$$m(v^2/\rho)\cos\beta + m\dot{v}\sin\beta - F_{yR} - F_{Ay} - F_{xF}\sin\delta_F - F_{yF}\cos\delta_F = 0. \tag{23}$$

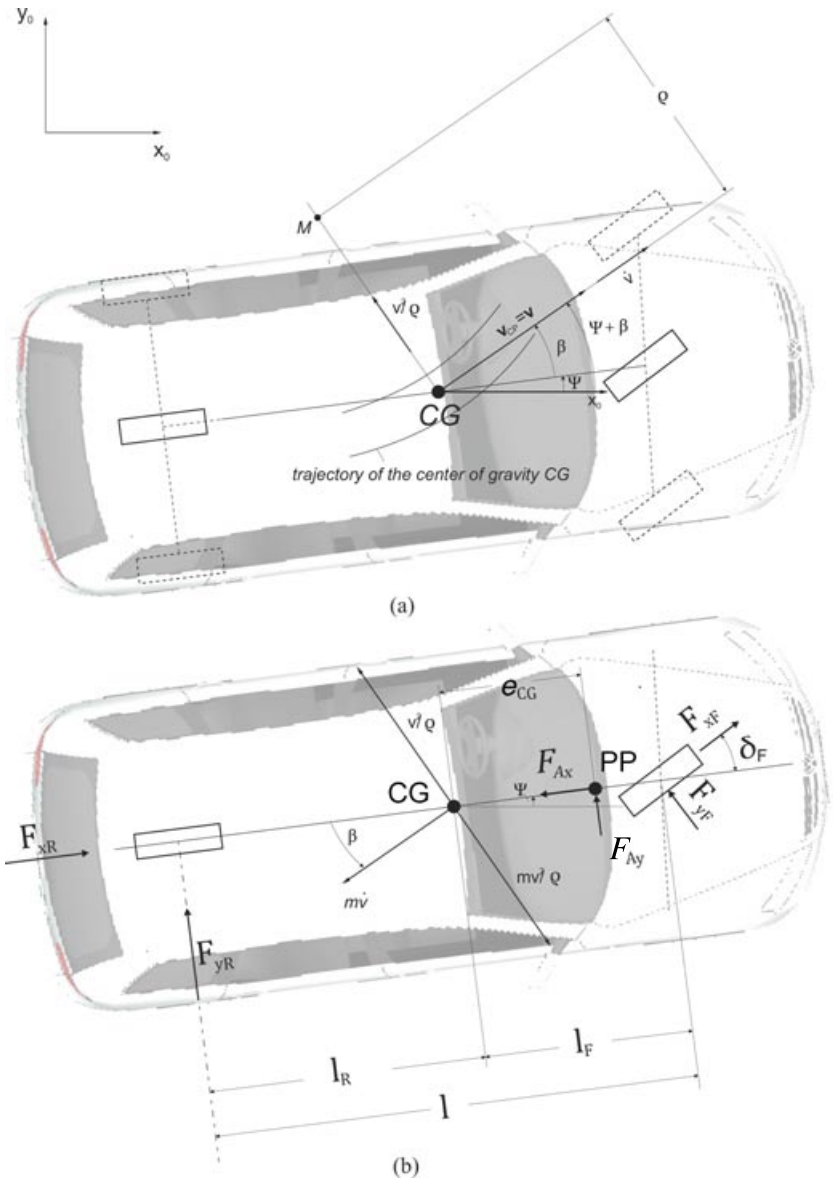


Figure 16. (a) Kinematic entities and (b) forces using a bicycle model. For a description of the symbols see the main text.

For the balance of moments we get

$$J_z\ddot{\Psi} - (F_{yF}\cos\delta_F + F_{xF}\sin\delta_F)l_F + F_{yR}l_R - F_{Ay}e_{\mathrm{CP}} = 0. \quad (24)$$

The lateral forces acting at the front wheel F_{yF} and real wheel F_{yR} emerge due to their slip angles α_R and α_F, the angle between the velocity vector at the wheel and the orientation of the wheel as depicted in Figure 17. The corresponding linearized relationships between slip angle and lateral forces are

$$F_{yF} = c_{\alpha F}\alpha_F, \quad (25)$$

$$F_{yR} = c_{\alpha R}\alpha_R, \quad (26)$$

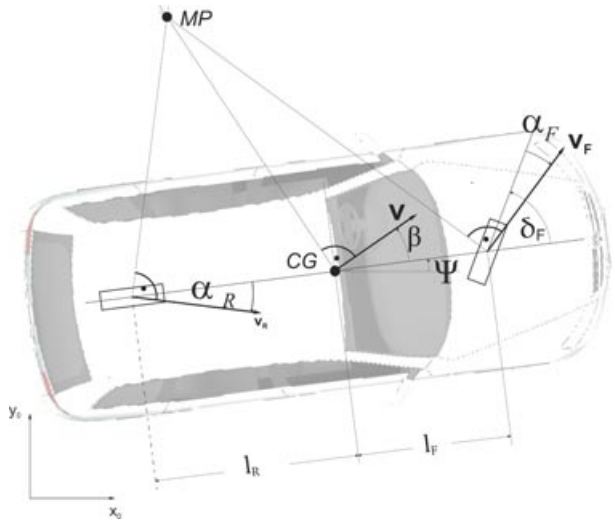


Figure 17. The slip angles α_R and α_F are the respective angular discrepancies between the velocity vector of a point at the center of the wheel and the orientation of the respective wheel. The lateral forces F_{yR} and F_{yF} that act on each wheel depend on these angles. The pole MP is in general different from the center of curvature M in Figure 16(a).

where the lateral constant force coefficients $c_{\alpha F}$ and $c_{\alpha R}$ with physical units [newtons/radian] depend on the tire. We aim at expressing the slip angles α_R and α_F as functions of the velocity vector $\mathbf{v}$ of the vehicle's center of gravity, the yaw rate $\dot{\Psi}$, and the sideslip angle β. To derive these dependencies, consider Figure 17 and the fact that the components of the velocity vectors $\mathbf{v}$, $\mathbf{v}_R$, and $\mathbf{v}_F$ in the longitudinal direction of the vehicle have to be equal. This is simply because the vehicle cannot stretch, resulting in

$$v\cos\beta = v_R\cos\alpha_R, \tag{27}$$

$$v\cos\beta = v_F\cos(\delta_F - \alpha_F), \tag{28}$$

where v, v_R, and v_F are the lengths of the vectors $\mathbf{v}, \mathbf{v}_R$, and $\mathbf{v}_F$. The velocity components in the lateral direction to the vehicle's centerline differ as a consequence of the yaw rate $\dot{\Psi}$ evolving different velocity contributions over the lengths l_R and l_F:

$$v_R\sin\alpha_R = l_R\dot{\Psi} - v\sin\beta, \tag{29}$$

$$v_F\sin(\delta_F - \alpha_F) = l_F\dot{\Psi} + v\sin\beta. \tag{30}$$

With $\tan x = \sin x/\cos x$ we combine the two pairs of equations, yielding

$$\tan\alpha_R = \frac{l_R\dot{\Psi} - v\sin\beta}{v\cos\beta}, \tag{31}$$

$$\tan(\delta_F - \alpha_F) = \frac{l_F\dot{\Psi} + v\sin\beta}{v\cos\beta}. \tag{32}$$

Resolving for α_F and α_R yields

$$\alpha_R = \arctan\frac{l_R\dot{\Psi} - v\sin\beta}{v\cos\beta}, \tag{33}$$

$$\alpha_F = \delta_F - \arctan\frac{l_F\dot{\Psi} + v\sin\beta}{v\cos\beta}. \tag{34}$$

Inserting the derived equations for α_R and α_F into Eqs. (26) and (25), the lateral forces can be expressed as functions of $\beta, \dot{\Psi}$, and v:

$$F_{yR}(\beta, \dot{\Psi}, v) = c_{\alpha R}\arctan\frac{l_R\dot{\Psi} - v\sin\beta}{v\cos\beta}, \tag{35}$$

$$F_{yF}(\beta, \dot{\Psi}, v, \delta_F) = c_{\alpha F}\left(\delta_F - \arctan\frac{l_F\dot{\Psi} + v\sin\beta}{v\cos\beta}\right). \tag{36}$$

Next, we want to express the centripetal acceleration v^2/ρ in Eqs. (22) and (23) in terms of the velocity v, the sideslip rate $\dot{\beta}$, and the yaw rate $\dot{\Psi}$. The curvature of the trajectory at the center of gravity of the vehicle is $1/\rho$. Another way to express the curvature of the trajectory is to consider the change of course angle $d(\beta + \Psi)$ along an infinitesimal step $\mathrm{d}u = v\mathrm{d}t$ within the infinitesimal time step $\mathrm{d}t$ along the trajectory, equating to

$$\frac{1}{\rho} = \frac{\mathrm{d}(\beta + \Psi)}{\mathrm{d}u} = \frac{\mathrm{d}(\beta + \Psi)}{v\mathrm{d}t} = \frac{\dot{\beta} + \dot{\Psi}}{v}. \tag{37}$$

Hence, the centripetal acceleration can be expressed by

$$\frac{v^2}{\rho} = v^2\frac{\dot{\beta} + \dot{\Psi}}{v} = v(\dot{\beta} + \dot{\Psi}). \tag{38}$$

Inserting Eqs. (35), (36), and (38) into Eqs. (22), (23), and (24) yields the following balances:

$$mv(\dot{\beta} + \dot{\Psi})\sin\beta - m\dot{v}\cos\beta + F_{xR} - F_{Ax} + F_{xF}\cos\delta_F - F_{yF}(\beta, \dot{\Psi}, v, \delta_F)\sin\delta_F = 0, \tag{39}$$

$$mv(\dot{\beta}+\dot{\Psi})\cos\beta + m\dot{v}\sin\beta - F_{yR}(\beta,\dot{\Psi},v) - F_{Ay} - F_{xF}\sin\delta_F - F_{yF}(\beta,\dot{\Psi},v,\delta_F)\cos\delta_F = 0, \quad (40)$$

$$J_z\ddot{\Psi} - [F_{yF}(\beta,\dot{\Psi},v,\delta_F)\cos\delta_F + F_{xF}\sin\delta_F]l_F + F_{yR}(\beta,\dot{\Psi},v)l_R - F_{Ay}e_{CP} = 0. \quad (41)$$

Multipyling Eq. (39) with $\sin\beta$ and adding Eq. (40) multiplied with $\cos\beta$ yields

$$\begin{aligned} mv(\dot{\beta}+\dot{\Psi})(\sin^2\beta+\cos^2\beta) &+ \sin\beta[F_{xR} - F_{Ax} \\ &+F_{xF}\cos\delta_F - F_{yF}(\beta,\dot{\Psi},v,\delta_F)\sin\delta_F] \\ &+\cos\beta[-F_{yR}(\beta,\dot{\Psi},v) - F_{Ay} - F_{xF}\sin\delta_F \\ &-F_{yF}(\beta,\dot{\Psi},v,\delta_F)\cos\delta_F] = 0, \end{aligned} \quad (42)$$

$$\begin{aligned} \dot{\beta} = &-\dot{\Psi} \\ &-\frac{\sin\beta[F_{xR} - F_{Ax} + F_{xF}\cos\delta_F - F_{yF}(\beta,\dot{\Psi},v,\delta_F)\sin\delta_F]}{mv} \\ &-\frac{\cos\beta[-F_{yR}(\beta,\dot{\Psi},v) - F_{Ay} - F_{xF}\sin\delta_F - F_{yF}(\beta,\dot{\Psi},v)\cos\delta_F]}{mv} \\ =:& f_2(\beta,\dot{\Psi},v,\delta_F,F_{xF},F_{xR},F_{Ax},F_{Ay}). \end{aligned} \quad (43)$$

Resolving Eq. (41) for $\ddot{\Psi}$ yields

$$\begin{aligned} \ddot{\Psi} &= 1/J_z\big\{[(F_{yF}(\beta,\dot{\Psi},v,\delta_F)\cos\delta_F + F_{xF}\sin\delta_F]l_F \\ &\quad -F_{yR}(\beta,\dot{\Psi},v)l_R + F_{Ay}e_{CP}\big\} \\ &=: f_1(\beta,\dot{\Psi},v,\delta_F,F_{xF},F_{Ay}). \end{aligned} \quad (44)$$

Multipyling Eq. (39) with $\cos\beta$ and subtracting Eq. (40) multiplied by $\sin\beta$ yields

$$\begin{aligned} -m\dot{v}(\cos^2\beta+\sin^2\beta) &+ \cos\beta[+F_{xR} - F_{Ax} \\ &+ F_{xF}\cos\delta_F - F_{yF}(\beta,\dot{\Psi},v,\delta_F)\sin\delta_F] \\ &- \sin\beta[-F_{yR}(\beta,\dot{\Psi},v) - F_{Ay} - F_{xF}\sin\delta_F \\ &-F_{yF}(\beta,\dot{\Psi},v,\delta_F)\cos\delta_F] = 0. \end{aligned} \quad (45)$$

Solving for $\dot{v}$ yields

$$\begin{aligned} \dot{v} = &\frac{\cos\beta[+F_{xR} - F_{Ax} + F_{xF}\cos\delta_F - F_{yF}(\beta,\dot{\Psi},v,\delta_F)\sin\delta_F]}{m} \\ &-\frac{\sin\beta[-F_{yR}(\beta,\dot{\Psi},v) - F_{Ay} - F_{xF}\sin\delta_F - F_{yF}(\beta,\dot{\Psi},v,\delta_F)\cos\delta_F]}{m} \\ =:& f_3(\beta,\dot{\beta},\dot{\Psi},v,\delta_F,F_{xF},F_{xR},F_{Ax},F_{Ay}). \end{aligned} \quad (46)$$

Summarizing, we have the set of three coupled nonlinear differential equations

$$\ddot{\Psi} = f_1(\beta,\dot{\Psi},v,\delta_F,F_{xF},F_{Ay}), \quad (47)$$

$$\dot{\beta} = f_2(\beta,\dot{\Psi},v,\delta_F,F_{xF},F_{xR},F_{Ax},F_{Ay}), \quad (48)$$

$$\dot{v} = f_3(\beta,\dot{\beta},\dot{\Psi},v,\delta_F,F_{xF},F_{xR},F_{Ax},F_{Ay}). \quad (49)$$

6.2. Numerical Integration

Given an initial velocity $v(t_0)$, initial yaw rate $\dot{\Psi}(t_0)$, and sideslip angle $\beta(t_0)$ at time step $t_0 = 0$, our goal is to numerically calculate the functions $v(t), \dot{\Psi}(t)$, and $\beta(t)$ for a duration of $T = 0.1$ s (one LIDAR frame), because those functions can then be used to calculate the trajectory of the vehicle. The control commands are the steering angle $\delta_F(t)$ and the force $F_{xR}(t)$ for velocity control. We assume a rear wheel drive setting $F_{xF} = 0$. We ignore lateral wind $F_{Ay} = 0$. For numerical integration we use explicit forward Euler with temporal step size $\mathrm{d}t = 1.0^{-3}$ s. The overall integration scheme is

$$\begin{aligned} \dot{\Psi}(t_k+\mathrm{d}t) &= \dot{\Psi}(t_k) + f_1(\beta(t_k),\dot{\Psi}(t_k),v(t_k),\delta_F(t_k),F_{xF} \\ &\quad = 0, F_{Ay} = 0)\,\mathrm{d}t, \\ \beta(t_k+\mathrm{d}t) &= \beta(t_k) + f_2(\beta(t_k),\dot{\Psi}(t_k),v(t_k),\delta_F(t_k), \\ &\quad F_{xR}(t_k), F_{xF} = 0, F_{Ax})\,\mathrm{d}t, \end{aligned}$$

$$v(t_k + \mathrm{d}t) = v(t_k) + f_3[\beta(t_k), \dot{\beta}(t_k), \dot{\Psi}(t_k), v(t_k), \delta_F(t_k), F_{xR}(t_k), F_{xF} = 0, F_{Ax}, F_{Ay} = 0]\,\mathrm{d}t.$$

The trajectory $x(t)$, $y(t)$ of the vehicle's center of gravity is then computed by numerical integration of

$$x(t) = x(t_0) + \int v(t)\cos(\beta(t) + \Psi(t))\mathrm{d}t, \quad (50)$$

$$y(t) = y(t_0) + \int v(t)\sin(\beta(t) + \Psi(t))\mathrm{d}t. \quad (51)$$

For all our numerical integrations we use the following parameters:

- mass $m = 2{,}900$ kg
- inertial moment $J_z = 5{,}561$ kgm^2
- lateral force coefficient for front wheels $c_{\alpha F} = 80{,}000$ N/rad
- lateral force coefficient for rear wheels $c_{\alpha F} = 110{,}000$ N/rad
- distance from center of gravity to front axis $l_f = 1.425$
- distance from center of gravity to rear axis $l_f = 1.425$ (center of gravity right between the axes)
- integration step width $\mathrm{d}t = 0.001$ s

Because we have a low-level controller for the velocity and consider only slow changes in the velocity, we ignore the longitudinal forces, that is, we set F_{xR}, F_{xF}, and $F_{Ax} = 0$. We also ignore lateral wind, which is simply unknown; hence $F_{Ay} = 0$.

6.3. Can the Curvatures of Our Tentacles Be Executed?

When our vehicle drives with a constant velocity v and steering angle for some time period, the state $(\dot{\Psi}, \beta, v)^T$ of the vehicle becomes steady and the vehicle drives in a circle. This steady state and its geometric relation to the final circular trajectory are depicted in Figure 18(a). Note that the steering angle δ_F is not part of the system state. Rather it can be calculated from a given system state. For the closed-form solution of the linearized bicycle model, there exists an analytical solution for calculating the state $(\dot{\Psi}, \beta, v)^T$ and the corresponding steering angle δ_F. Indeed, the yaw rate $\dot{\Psi}$, with $c = 1/r$ being the curvature of the circle with radius r, is quite simple to calculate:

$$\dot{\Psi} = cv. \quad (52)$$

However, calculating the sideslip angle β and the required steering angle δ_F is more complicated: In the case of our nonlinear model we use a recursive search algorithm to calculate the values. The search algorithm exploits the fact that once the steering angle δ_F is known, one can run a simulation (by integrating

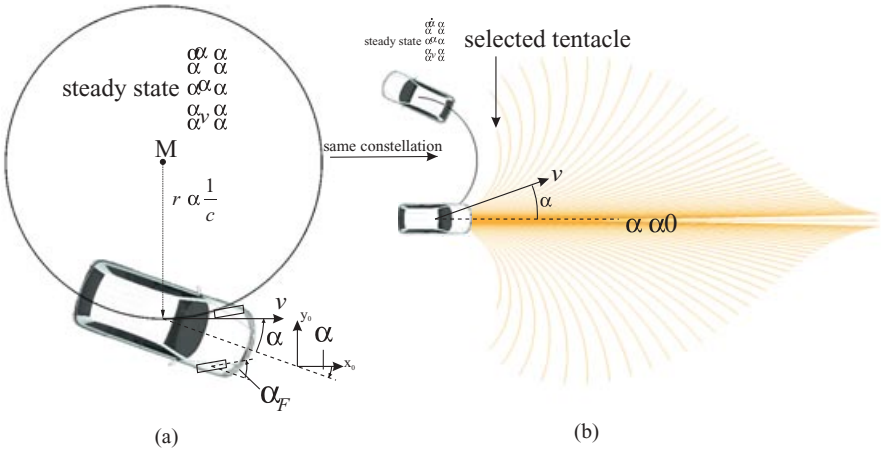


Figure 18. (a) Given a circle with radius r and a velocity v, a steady state $(\dot{\Psi}, \beta, v)^T$ and a steering angle δ_F can be calculated, such that the vehicle drives on the circle. (b) When calculating the steady state that corresponds to a tentacle and assuming that the vehicle has the correct state from the very beginning, the vehicle's resulting trajectory does not correspond to the tentacle. The reason lies in the sideslip angle β.

the differential equations), initializing the state to $(0, 0, v)^T$ and commanding the constant steering angle. Simulating over a period of T, one can observe whether the state becomes steady (by continuously comparing the current state and a state ΔT_{back} back in time) and read out the values for $\dot{\Psi}$ and β in this case (v is known from the very beginning). The search algorithm then divides the range of potential solutions for δ_F $(-0.5\pi, \ldots 0.5\pi)$ into n intervals ($n = 10$ in our implementation), determines the steady states (in case of convergence) for each of the $n + 1$ interval boundaries, finds the interval that encloses the solution [by comparing the respective curvatures that can be computed via Eq. (52)], and recursively narrows the interval. This method is possible because of the monotonic dependency of the steering angle δ_F and the resulting steady curvature $\dot{\Psi}/v$.

Because each tentacle is a circular arc, we can calculate the vehicle's corresponding steady state with the above method such that the vehicle drives a circle with the same curvature. We will call this steady state corresponding to the tentacle the *state of the tentacle*. The velocity of this state is the velocity of the tentacle's speed set. At first glance, it is surprising that even if the vehicle's state is initialized such that the final trajectory has the curvature of the tentacle, the resulting trajectory is rotated against the tentacle [see Figure 18(b)]. Even under perfect conditions, the vehicle cannot drive along the tentacle. The reason is that driving constantly along a circle requires a nonzero sideslip angle β, meaning that in contrast to the velocity vector the vehicle is not heading tangential to the circle. Thus, when starting as illustrated in Figure 18(b), the final trajectory is rotated against the selected tentacle. The radius of the resulting circular trajectory is correct, however. Hence, we can store with each tentacle a steering angle that at least produces the correct curvature once the state is steady. The error is not in a wrong system state but the initial geometric constellation. Right at the beginning the state of the vehicle has a nonzero sideslip angle β. Hence, if both trajectories should be aligned, the vehicle would have to start rotated about $-\beta$. This is illustrated in Figure 19.

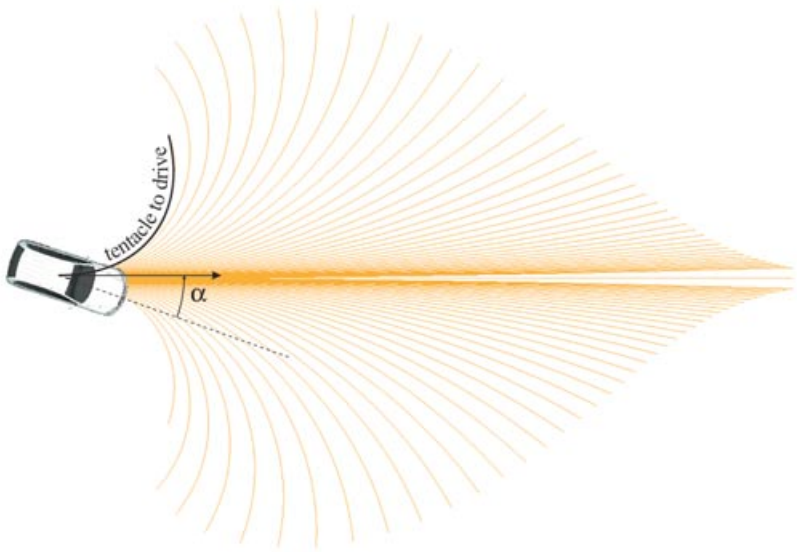


Figure 19. Given a tentacle that the vehicle should precisely drive, the tangent at the first point of the tentacle has to be aligned with the expected sideslip angle β. In other words, the whole tentacle has to be rotated about β in order to be precisely executable. This is a first potential improvement suggested by this analysis. Note that this does not mean that the tentacles as designed in Section 2 are invalid, because the resulting path will be shown to still be in the corridor of the classification area. This figure also shows the extreme case with a large sideslip angle β. The tentacle considered is the most curved one of all tentacles in all speed sets.

6.4. Determining Path Deviations

In general, the state of the vehicle before executing a new tentacle will not be the steady state of the new tentacle. Also, the steering angle δ_F will not correspond to the tentacle's curvature at the beginning. To predict the error that happens in such cases, we aim at predicting the trajectory the vehicle executes and comparing this trajectory against the tentacle's curve. To calculate the resulting trajectory, we assume that the steering angle is controlled to the goal steering angle δ_F by a low-level controller that linearly adjusts δ_F to the desired value. In our system this controller has a maximum rate of 0.3 rad/s. Note that on MuCAR-3 the steering angle controller was itself based on an angular rate controller. For the vehicle of Team AnnieWAY the angular rate controller was itself the lowest unit. A complication arises because in the direct execution mode the goal steering angle is not that of the tentacle but the value calculated by the recursive formula (17).

6.4.1. Worst-Case Transitions

To calculate upper bounds for path deviations, we will now proceed by considering transitions from a source tentacle t_s to a destination tentacle t_d. First, we consider only tentacles from the same speed set and we assume that the vehicle has initially acquired the steady state of the source tentacle and that it starts at the correct angle $\Psi(t_0) = -\beta$ [2] such that the resulting trajectory perfectly aligns with the source tentacle. Then we switch to the second tentacle, calculating the filtered goal steering angle by Eq. (17). When the transition to the new steering angle is executed, we simulate the low-level controller, changing the steering angle at maximum rate (0.3 rad/s).

The assumption that the vehicle initially has reached the steady state of the first tentacle is not true in general. However, if we select the extreme case for an initial tentacle—that is, the outermost left or outermost right one—and consider the transition from this tentacle, then this represents an upper bound on the deviation of the resulting path. Figure 20 shows the considered worst-case scenarios and the tentacle's trajectories against which we want to compare the simulated trajectories. The assumption made here is that the larger the deviation in the steady states of the source and destination tentacle is, the larger is the resulting deviation of the trajectory. This assumption seems to be valid for the normal case, that is, within the scope of validity of the bicycle model. To simulate the results we will let the vehicle drive along the source tentacle for a time period T_s and then switch to the destination tentacle in the above-described manner, executing it for a time period of T_d seconds. Whereas the length of the first time period T_s does not matter, because the path deviation is zero, the second tentacle is executed only for one LIDAR frame (0.1 s), because then a new tentacle is selected again. Nevertheless, to see the qualitative behavior of the deviations well, we will first consider the full-length execution of the second tentacle. Note that executing a tentacle longer than 0.1 s means that the tentacle is repeatedly selected and, hence, the smoothing filter (17) has to be applied consecutively every 0.1 s. For Eq. (17) we will use a value of $\kappa = 0.9$. Figure 21 shows the resulting trajectories when simulating the cases shown in Figure 20. At low speeds the results can be quite counterintuitive: The large sideslip angle driving the source tentacle can let the final deviated trajectory appear at an intuitively unexpected side of the destination tentacle (see Figure 20). The more intuitive case that occurs at higher velocities is shown in Figure 22.

[2] Not $\dot{\Psi}(t_0)$!; see Eq. (51) and Figure 19.

6.4.2. Calculating Deviations

We define a distance measure for the deviations by considering the vehicle's true center of gravity at time t while traveling along the true trajectory and a second point on the destination tentacle at the same travel length (but along the destination tentacle). We consider all correspondences while integrating over the differential motion equations. From all corresponding pairs of points and their distances, the largest distance is taken as the deviation between the two curves. In Figures 21 and 22 some of those corresponding points are sampled and shown by interconnecting them with straight-line segments. The maximum deviations often occur at the end of the curves; however, there exist cases in which the curves intersect, and because of this nonmonotonic behavior, we need to consider all correspondences. The measure is conservative in the sense that it punishes not only deviations in space but also deviations in time. This is because the correspondences are established by means of the velocity and time-dependent traveled length from the beginning of both respective curves.

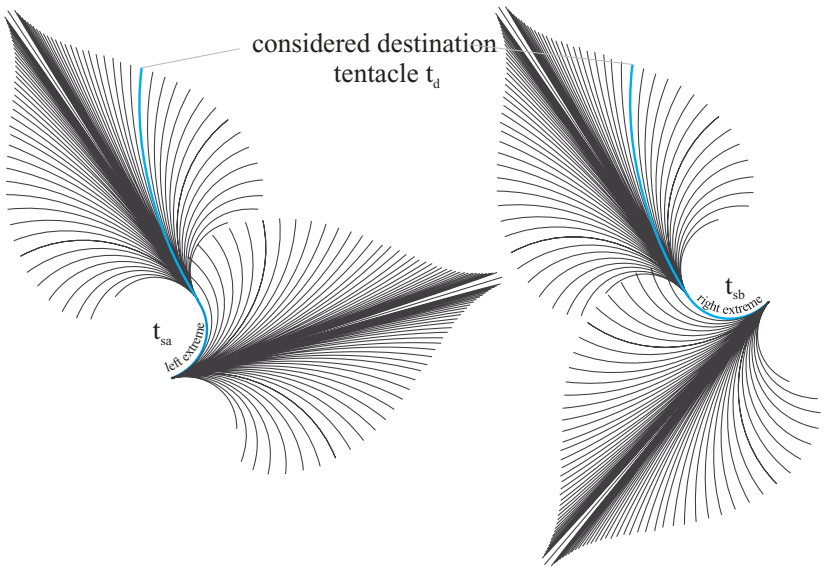


Figure 20. Two cases, in each of which two tentacles are executed sequentially. In both cases the top tentacle is the same destination tentacle t_d considered and only the first tentacle (source tentacle) varies (t_{sa} or t_{sb}). For a given destination tentacle t_d, we consider the left and right extreme transitions to this tentacle. We assume that the vehicle has reached a steady state on the source tentacle and then executes the destination tentacle. The source tentacles t_{sa} and t_{sb} are shown in full length such that they can be better identified within their speed set. However, a tentacle is executed only for the short time period of 0.1 s; hence the first tentacle will not be completely driven as shown in this figure. Only a small fragment of the first tentacle is driven, and the second tentacle (the tentacle considered) directly follows this small fragment. Before a transition to the destination tentacle is initiated, we assume that the vehicle is in the steady state that corresponds to the respective source tentacles such that the resulting trajectories and the source tentacles align perfectly. As shown in Figure 19, this requires the vehicle to start rotated about $-\beta$—the expected sideslip angle—in relation to the start orientation of the respective source tentacles.

In Figures 21 and 22 we consider the deviation of executing the full length of the destination tentacle. Now we are interested in the maximum deviation that can occur for each tentacle within one LIDAR frame, that is, within 0.1 s. For this sake, we consider each tentacle in each speed set as a destination tentacle in the sense of the above-described evaluation and calculate the maximum deviation that can happen within this 0.1 s. We need to consider only one extreme tentacle and calculate the deviations to all the other tentacles because of the symmetry of the tentacles. The other worst case then is automatically included in terms of the deviation of the corresponding mirrored destination tentacle. Figure 23 shows the resulting deviations dev(j,i) for all speed sets j and all tentacles i. Interestingly, the deviation increases first with higher velocities and than decreases again from speed set 6 on (3.5 m/s). As expected, the deviation is larger for more-curved tentacles. However, the important fact is that for each tentacle, we have calculated an upper bound for a deviation. For every tentacle, this upper deviation is the maximum of both extreme transitions to this tentacle. In terms of Figure 23 the upper bound d_{ji} for tentacle i in speed set j is

$$d_{ji} := \max(\mathrm{dev}(j, i), \mathrm{dev}(j, n_j - i)), \qquad (53)$$

where n_j ($n_j = 81$ for all speed sets j in our case) is the number of tentacles in speed set j. As can be seen, within 0.1 s no deviation exceeds 6 cm. If the classification area of the respective tentacle is about d_{ji} larger than the vehicle, then we can ensure that within the 0.1 s the vehicle will not collide with a (static) obstacle. This is because a tentacle is classified as nondrivable if an obstacle is within that area. It remains to show that neither can this happen at the next time step.[3]

[3]Note that only static obstacles are considered here. How the approach is combined with a separate obstacle tracker for moving objects (as was the case for the UC07) is detailed in the results section.

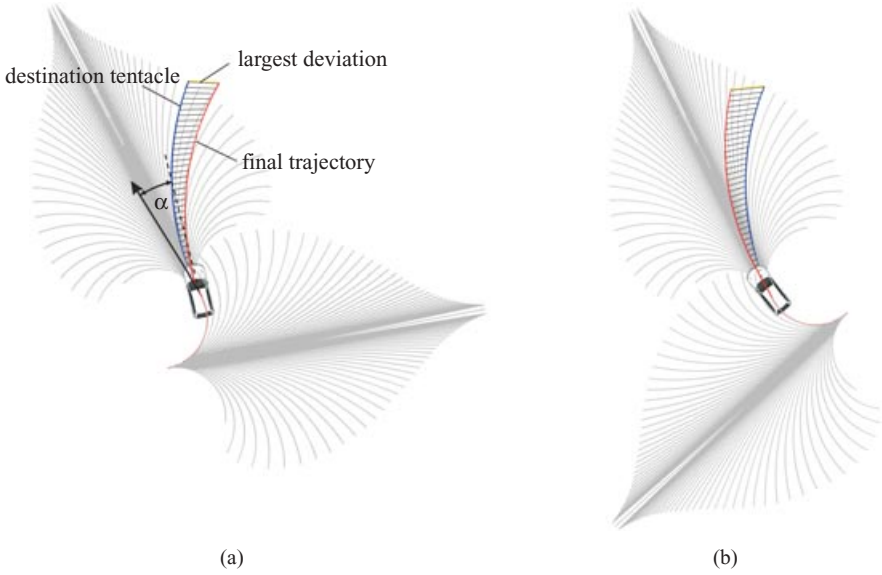


Figure 21. The resulting trajectories for the low velocity of 0.25 m/s simulated with the nonlinear bicycle model when executing the worst combinations shown in Figure 20 using the direct execution mode with $\kappa = 0.9$ [see Eq. (17)] and a steering rate limitation of 0.1 rad/s. For low velocities the result is counterintuitive: Whereas one might expect that the final trajectory would be to the left of the destination tentacle (as is the case in Figure 22), on the left-hand side, the resulting path is deviated to the right. The reason is the sideslip angle: While driving on the source tentacle the sideslip angle β is quite high, because of the high curvature. When the vehicle switches to the destination tentacle it has a heading direction pointing rightward of the destination tentacle. Because of the low speed, the new steering angle is adapted without the vehicle having moved far and the curve of the destination tentacle starts out at an angle that lets the final trajectory be located at the right-hand side of the destination tentacle. In panel b, the opposite happens. This behavior appears only at low speeds. The lines connecting the destination tentacle and the final trajectory show the deviations by interconnecting corresponding points at same lengths along the respective curves. The largest deviation is used to represent the overall deviation of the two trajectories.

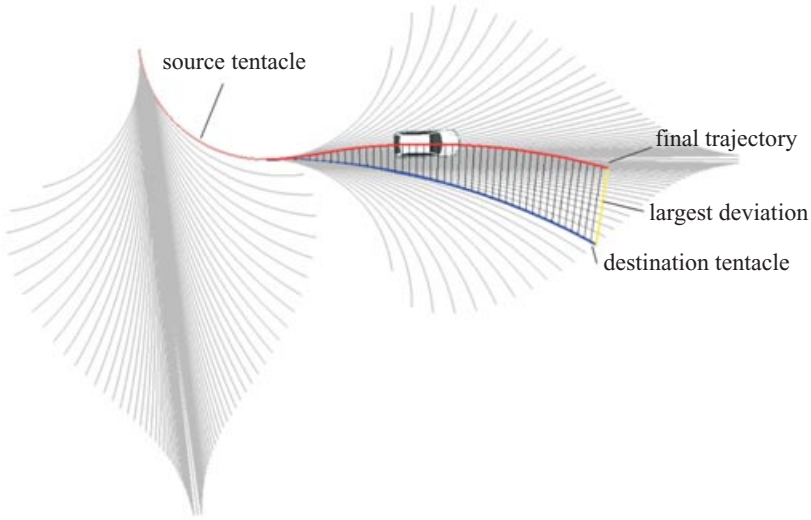


Figure 22. In this example the velocity is 1.66 m/s and the deviation of the resulting trajectory is as one would intuitively expect. A counterintuitive example is shown in Figure 21.

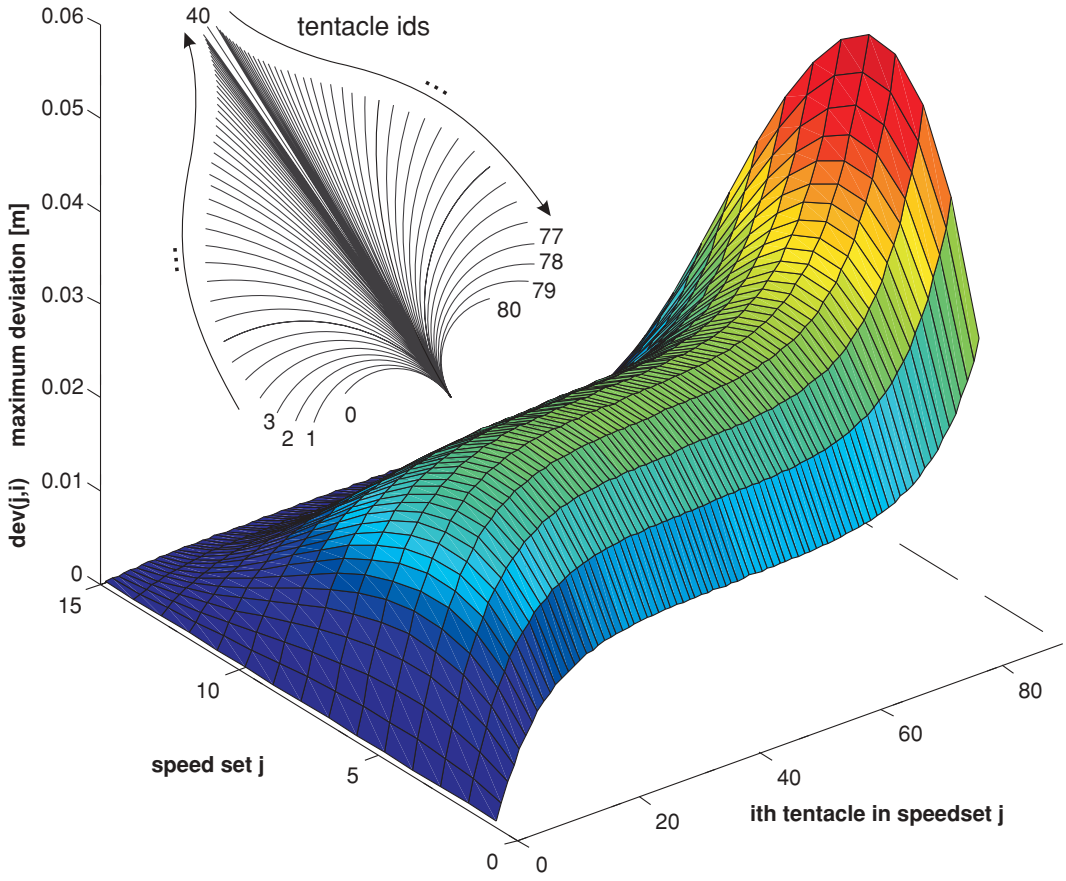


Figure 23. Trajectory deviations computed while letting a vehicle perform tentacle transitions, executing the destination tentacle for 0.1 s. The considered transitions are from the most curved tentacle 0 of a given speed set (left-hand side of the plot) to all the other tentacles i of the same speed set j. As can be seen, the ego transition from tentacle 0 to tentacle 0 has an expected deviation of zero. The maximum deviation is below 6 cm. Transitions from one speed set to another are not considered, assuming that the vehicle performs only slow velocity changes.

Note that this tolerance of 6 cm was well included in our definition of the classification areas we used for the UC07 [see Eq. (8)]. Here, even the smallest width of any classification area exceeds 2×1.7 m $= 3.4$ m but the VW-Passat is only about 2.01 m wide.

6.5. Considering an Extreme Case

We showed that if a tentacle is selected—and therefore was classified as drivable before—the method guarantees that the vehicle will not hit an obstacle within the next 0.1 s. However, it is not automatically given that no collision can happen in the successive frames. To avoid collisions the braking behavior plays an essential role. For the UC07, we use the simple mechanism that if no drivable tentacle was available, the vehicle selected the tentacle with the largest distance to the first obstacle and decelerated with $a = -0.5g$ for the next 0.1-s frame along that *brake tentacle*. The speed reduction leads to decreased crash distance and, depending on the case, can free new tentacles or, if all obstacles are too close, lead to a continuation of braking. Freeing tentacles can happen by the reduction of the crash distance itself, letting some obstacles move behind the new distance, or by switching to a new speed set and freeing new geometric primitives. The intricate cases do not emerge from obstacles in front of the vehicle,

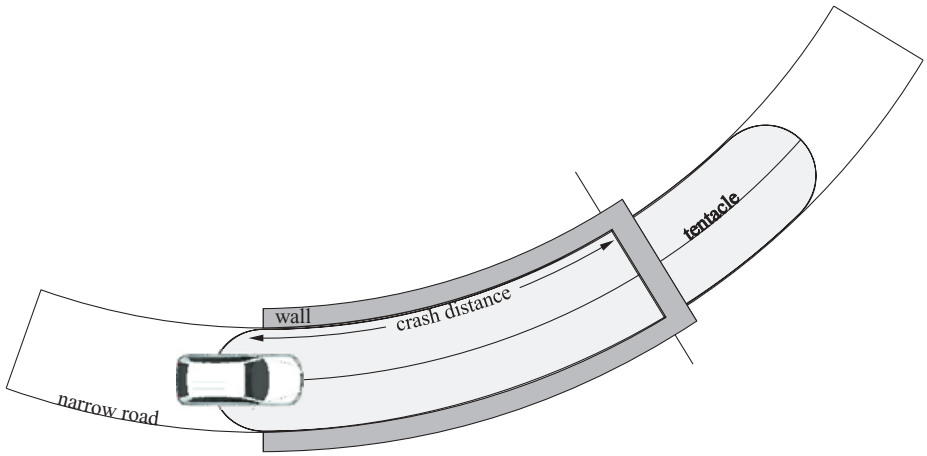


Figure 24. The constructed case, in which the vehicle follows a narrow road that has exactly the curvature of the tentacle that is being executed. The road has exactly the width of the classification area, such that the wall is not recognized as obstacle at the sides or at the front, because the wall is at a distance slightly above the crash distance. Hence, any little movement of the car will bring an obstacle into the classification area. This case is used in the main text to discuss whether our method could potentially fail. It is evident that only the selected tentacle will be classified as drivable; all the other tentacles intersect the wall before the crash distance.

because the crash distance is designed such that there is plenty of room for stopping, but from those obstacles the vehicle passes closely by. This shall be shown in an example. Consider the admittedly very unrealistic case shown in Figure 24. In the considered case the classification area of the last selected tentacle is enclosed by a wall, such that the classification area is still free of obstacles. Also the wall cuts the tentacle just behind the crash distance, such that the vehicle detects the obstacle but considers it being still too distant to be a threat for the current speed. Hence, the vehicle drives along the tentacle for 0.1 s. If the execution of the tentacle is perfect, no problem arises, because the crash distance will be undercut at the next time frame and the vehicle can safely brake using the remaining part of the crash distance. That is, the space for braking reduces only by $v \cdot 0.1$ s if v is the speed of the vehicle. However, if the vehicle deviates from the tentacle's trajectory in terms of an angular deviation, parts of the wall can suddenly protrude into the classification area at a distance far before the crash distance reduced by $v \cdot 0.1$ s. This situation is shown in Figure 25. Because the wall protrudes into the classification area, the tentacle is classified as nondrivable. It is evident that all the other tentacles are classified as nondrivable, too. Hence the vehicle will decelerate for the next 0.1 s, choosing the tentacle with the largest distance to the first obstacle as the path for braking. Braking with a deceleration of $a = -0.5g \approx -0.5 \times 9.81$ m/s^2 then takes place along the brake tentacle. The process then repeats until the vehicle has stopped. A formal proof that a collision can be avoided in all cases is very hard to achieve because of the complex interplay between geometric aspects, the process of drivability classification, the tentacle selection for braking, the modification of the crash distance, and the final deviations of the tentacle's trajectories due to vehicle dynamics.

7. EXPERIMENTS AND COMPETITIONS

Our approach was tested and demonstrated in several scenarios experimenting with different parameter settings and execution modes.

7.1. System Integration at the C-Elrob 2007

At the C-Elrob 2007 we ran the vehicle in exploration mode. Here, both factors a_0 and a_2 of Eq. (16) were zero. That is, the vehicle decided to drive always toward the flattest areas without regarding any given trajectory. Tentacle execution was done by temporal filtering of the selected tentacles and fragment execution as described above. Even this simple approach

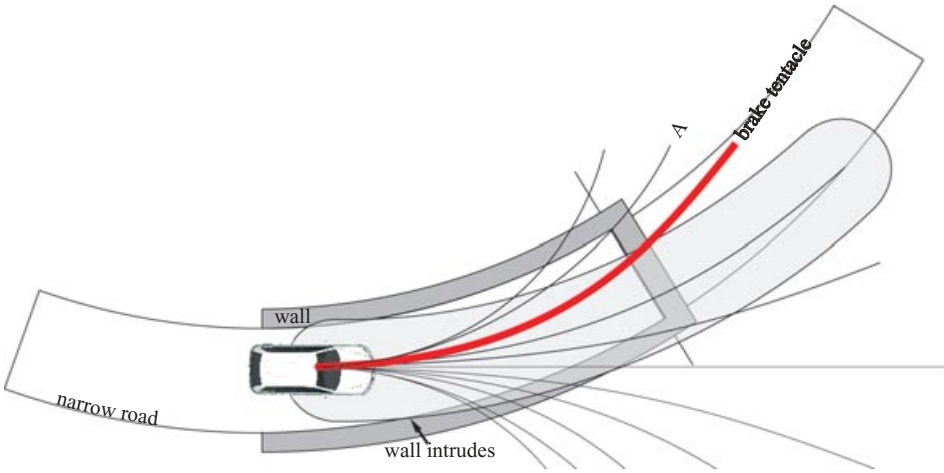


Figure 25. The vehicle has moved for 0.1 s, and we assume that a deviation from the tentacle's path occurred, such that the vehicle has an angular deviation from the desired trajectory. As a consequence the wall at the bottom intrudes into the classification area at a distance close to the vehicle. Hence, the tentacle will be classified as nondrivable. Evidently, all other tentacles will also be classified as nondrivable and the vehicle will start to brake for the next 0.1 s. For braking, it will choose the tentacle with the largest distance to the first obstacle. Note that this is not the tentacle marked with A, because obstacles are mapped orthogonally onto the classification histogram of the tentacle (see Section 3.3).

worked surprisingly well, and at the C-Elrob 2007 our CoTeSys[4] demonstrator MuCAR-3 drove a combined urban and nonurban course in record time.[5] Manual intervention was necessary only at some crossings to force the vehicle to take the correct course (because we did not exploit any GPS knowledge). In separate demonstrations we let the vehicle drive along narrow serpentines. It was able to drive the serpentines from the base camp at the top of Monte Ceneri[6] down to the main road without using any GPS information (see Figure 26). Occasionally, the vehicle decided to enter the free area at point **A** in Figure 26, driving around some parking trucks there, leaving the area again after some time or getting stuck in a dead end.

7.2. System Integration within the DARPA Urban Challenge 2007 Finalist AnnieWAY

Owing to the success of our approach at the C-Elrob 2007 we integrated our method into the system of the DARPA Urban Challenge Team AnnieWAY (Kammel, 2007). A description of the overall system architecture is given by Kammel et al. (2008). In this setup tentacles were used only if the areas in the occupancy grid that corresponded to the current relevant section of the GPS trajectory were unexpectedly not free of obstacles. To check this we computed a speed-dependent lookahead distance d_{check} and defined a triangular mesh structure along the GPS section as illustrated in Figure 27. We then projected each triangle into the ego-centered occupancy grid and scanned the cells of the triangles using a standard triangle scan-conversion method from computer graphics. The area was regarded as free if fewer than $n = 2$ grid cells exhibited a z difference more than 0.1 m. If the area was regarded as occupied, the tentacle approach was switched on. Note that a separate obstacle tracker existed in the overall system. This obstacle tracker was run before the grid-checking mechanism, and the distance d_{check} was cropped by the distance to the first obstacle. Hence, the tentacles reacted only on obstacles that were unseen by the obstacle tracker. The obstacle tracker did detect and track obstacles only of the approximate size of a vehicle and evaluated the occupancy grid for obstacles only at the expected area of the road as given by the GPS way points. Hence, in cases of GPS offsets the obstacle tracker could miss obstacles. In these cases the tentacles provided a reactive protective shelter. However, even when the tentacles were switched on, they tried

[4]German Cluster for Exellence: Cognition For Technical Systems.
[5]http://www.unibw.de/lrt13/tas/pressestimmen.
[6]Monte Ceneri, Swizerland.

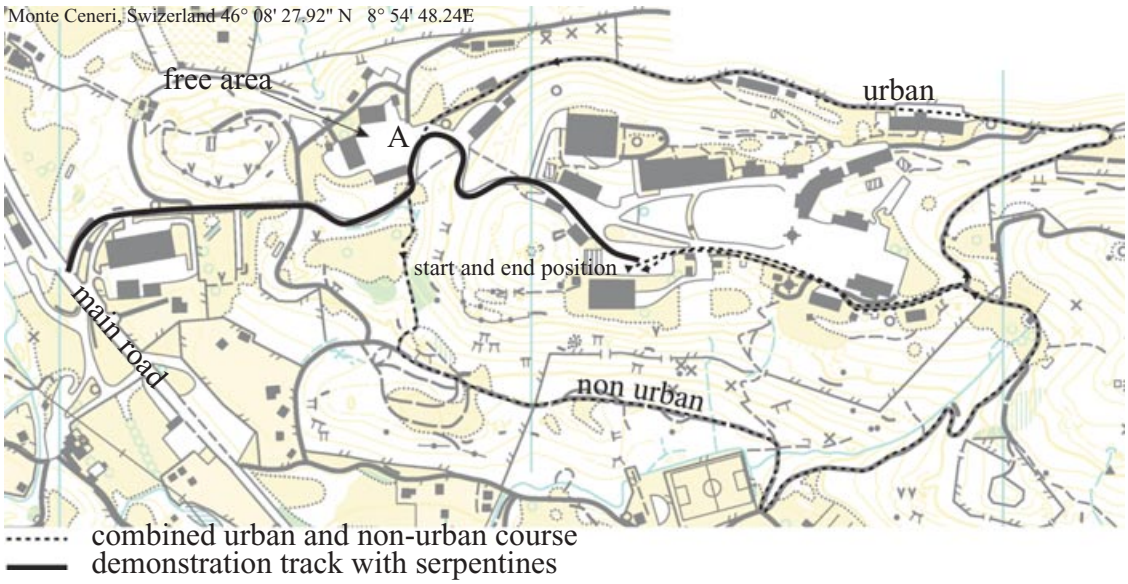


Figure 26. The solid line shows the serpentines from the base camp at Monte Ceneri, Swizerland, down to the main road our vehicle was able to drive fully autonomously without using any GPS information at the C-Elrob 2007. Occasionally, the vehicle entered the free area A. The dashed line shows the combined urban and nonurban course our vehicle drove in record time (manual intervention at some crossings).

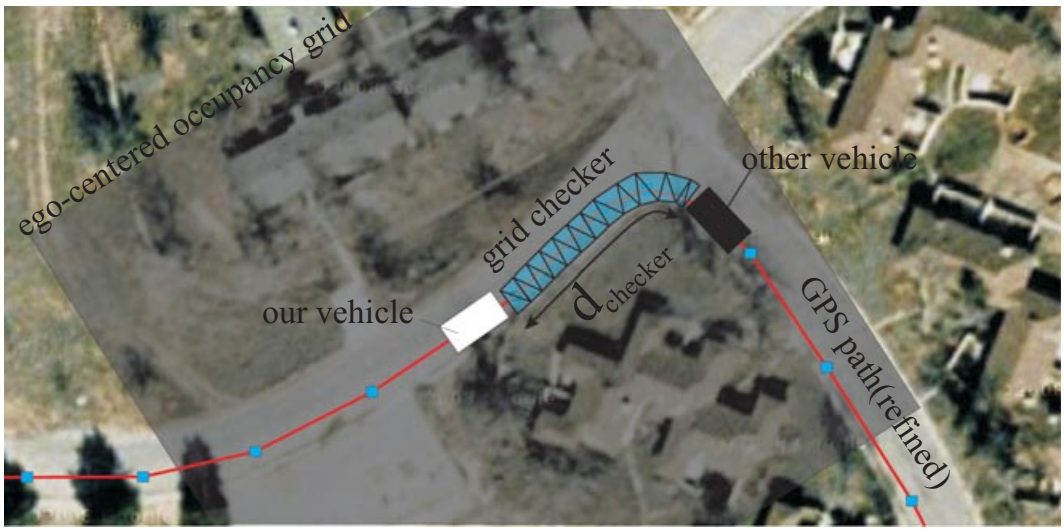


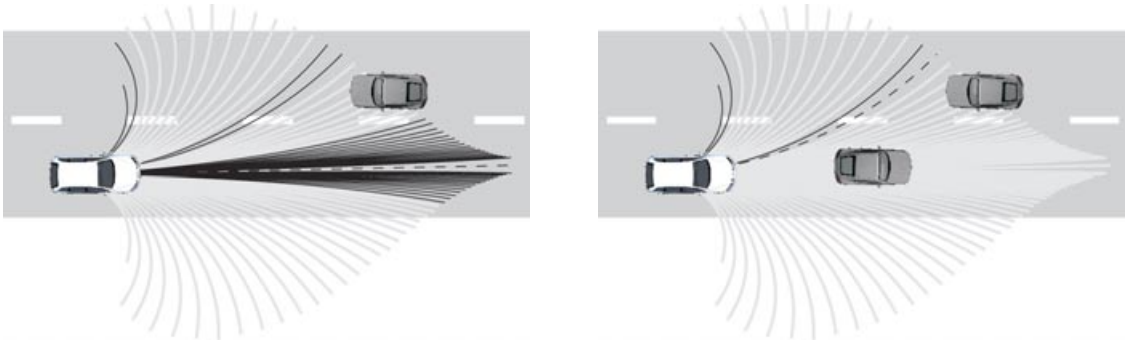
Figure 27. The grid checker was a triangular mesh that was defined along the current section of the RNDF in front of the vehicle. The triangles were projected into the ego-centered occupancy grid to see whether the respective area was free of obstacles. Tentacles were switched on only if the area was not clear. To avoid the tentacles reacting on other cars that were previously seen by the separate obstacle tracker (Kammel et al., 2008), the speed-dependent length d_{checker} was cropped by the distance to the first obstacle.

to follow the GPS trajectory if possible (because of the trajectory value). For the Urban Challenge setup we ran the tentacle approach with the parameters of Eq. (16) being $a_0 = 1$, $a_1 = 0$, and $a_2 = 0.5$ such that the vehicle had a preference to follow the given GPS trajectory but avoided obstacles at the same time. The choice $a_0 = 0$ means that flatness was actually ignored for the Urban Challenge. The rationale for the Urban Challenge was that as long as a tentacle is drivable, we would prefer the one that follows the GPS path instead of flat tentacles in the extreme. In general, the driving speed was determined by the behaviors of the overall system. However, the tentacle method was allowed to undercut this speed. During the UC07 the speed was cut down to 2.0 m/s when tentacles became active (if no lower speed was initially selected). The reason for this low speed was that if the tentacles became active, one could conclude that something unpredicted had happened, e.g., due to a GPS drift or an obstacle not detected by the obstacle detection/tracking module.

7.3. Experiments in Preparation for the Urban Challenge

In preparation for the Urban Challenge our approach was tested excessively at both a parking lot with manually placed obstacles and vehicles and a residential area (under safe conditions, security driver). The most difficult tests were performed in the residential area using a road network definition file as shown in Figure 28(a): Here, the GPS trajectory was intentionally defined directly through a traffic circle, such that if the GPS trajectory was executed blindly, the vehicle would collide with the circular area in the center [see Figure 28(b)]. In dozens of repetitions our approach was confirmed to safely drive around the traffic circle not strictly following the GPS trajectory. In a similar

Figure 28. This figure shows the test area and road network we used in preparation for the Urban Challenge 07. In excessive tests, our approach was proven to be able to safely drive in this conventional residential area, avoiding pavement edges, parking cars, and other obstacles. No traffic was present and a security driver was onboard for safety reasons. In particular, our approach mastered the following two difficult cases: (b) The GPS trajectory was intentionally defined to lead straight through a traffic circle. Our approach was repeatedly shown to drive around the traffic circle, not following the collision course of the GPS trajectory directly. (c) The GPS trajectory was defined to shortcut a road crossing colliding with neighboring houses. However, using our approach the vehicle followed the road to the next crossing and turned right correctly, not strictly following the GPS trajectory.



(a) (b)

Figure 29. The situation shown in panel a is handled well by our method. Some tentacles are blocked by the oncoming traffic (light gray) and the pavement edges of the road, leaving three groups of tentacles (black). It would be dangerous to select a tentacle leading to the left lane. However, one of the straight tentacles is selected (dashed black) because of the hysteresis step (see Section 4.6), preferring tentacles that are more similar to the last selected tentacle in ambiguous situations. The situation shown in panel b is dangerous because the vehicle would try to overtake the car in front, notwithstanding the oncoming traffic. Hence, our approach should be combined with a separate object tracker in traffic environments (e.g., predicting future obstacle positions and explicitly drawing them into the grid).

experiment, we intentionally misplaced the GPS trajectory at a straight road by an offset of about 10 m, such that the trajectory passed through the front gardens of the neighboring houses. Despite this severe offset, our approach lets the vehicle drive along the road, avoiding pavement edges, parking cars, trees, and other obstacles. Occasionally the vehicle tried entering wide doorways at the side corresponding to the GPS offset but stopped safely at dead ends. In the scenario shown in Figure 28(c) we intentionally misdefined the GPS trajectory short, cutting the road crossing by about 50 m through nearby buildings. Using our approach the vehicle did not follow the collision course but continued driving along the straight road, taking the next road correctly to the right.

Care must be taken when dealing with moving objects. Oncoming traffic as shown in Figure 29(a) is handled well by our algorithm, and we did not experience a single dangerous behavior in all our experiments. However, the case in Figure 29(b) is dangerous (for a human driver, too), because the vehicle would try overtaking the car in the right lane, although another car is approaching in the left lane. Hence, our approach should not be used in a standalone manner, in case of traffic scenarios. Instead a separate tracker for moving objects should be run in parallel (as we did for the 2007 Urban Challenge). One option for combining our method with a separate obstacle tracker is to predict future obstacle positions and explicitly draw them into the grid, blocking the corresponding tentacles. Another option (we applied at the 2007 Urban Challenge) is to evaluate tentacles just up to the first tracked object, such that a car driving ahead is not sensed by the tentacles. Speed control then has to be handled in such a way that the robot follows the car. In this way, our approach can be restricted to avoid only those obstacles that were not detected by the object tracker.

7.4. Performance at the Urban Challenge

At the 2007 Urban Challenge our method was active in all preliminary tests, tests A, B, and C, and also at the final race. Unfortunately, tentacle logging was off, and no quantitative data are available on how often or when the tentacles were active. The only certainty about them having been active was the slow speed of the AnnieWAY vehicle observable when driving along narrow passages. This can also be seen in videos we recorded. A typical scene where tentacles were active is depicted in Figure 30, showing narrow passages where even small GPS offsets could be dangerous. When tentacles were active the velocity was cropped to at most 2 m/s. In one test, our car slightly hit a wall. The reason was an error in the tentacle code but evidently a software bug. With this bug (which happened by switching from

Figure 30. At the 2007 Urban Challenge our approach was important at narrow passages where small GPS offsets could cause a collision.

debug to release mode), the vehicle had computed NaN-floats as curvatures and transmitted them to the low-level controller. In the final race, the AnnieWAY car stopped at the entry of a sparse zone. This was also due to a software bug, known to the team and not related to the tentacle method. Overall, we believe that our stop in the final was not because of a conceptual problem, but because of insufficient time for testing and bug fixing.

7.5. Lessons Learned

The major lesson learned at the DARPA Urban Challenge is the inevitability of having to handle problems arising due to GPS inaccuracies. Many teams participating at the Challenge reported that one of the key enabling techniques was to constantly keep track of the offset between GPS measurements and the DARPA-provided GPS way points. In this spirit, the approach presented in this paper allows following a given GPS trajectory while avoiding obstacles at the same time. Using feature-based references for estimating the GPS offset is particularly hard to accomplish, as free space areas are often lacking such features as lane markings. For example, the path planner that was active at zones had to cope with the problem that the GPS destination point was sometimes believed to be occluded by an obstacle due to GPS drifts.

We believe that this lesson points toward a new direction for how plans for navigation should be represented. To open this perspective, we briefly consider the merits of reactive schemes, such as the one this paper describes, and path planning schemes. The most obvious difference is the temporal behavior of both: whereas reactive schemes are fast and react immediately, trajectory planning methods are comparatively slow. Often, replanning is done at a rate too low to guarantee that no dynamic obstacle has moved into the planned track. On the other hand, path planning allows for anticipation and avoid traps a reactive scheme might just run into. However, the fundamental difference we would like to emphasize is the way both methods relate to the world as perceived. In contrast to path planning methods, in which planning takes place in abstract models of the world (e.g., SLAM and Cartesian map approaches in general), reactive schemes directly couple perception with action. Consider, for instance, a reactive navigation approach to driving along a road, guided by the principle to drive on flat ground. Then, the road essentially corresponds to what a planned path is in a path planning method.

We do not argue for reactive approaches in general but for their inherent property of direct reference to the world. Thus, the critical question is whether this property can somehow be transferred to planning methods. We believe that a change of the planning domain from Cartesian coordinate systems fixed to the ground to what one might call perceptual references is a promising direction. With perceptual ref-

erences, plans would be described similar to how humans communicate plans: "Overtake the obstacle to the left, pass the next one at the right side." We believe that successful future navigation systems will follow this direction instead of doing what most researchers do now: implementing "GPS-trains" following "GPS trails."

8. CONCLUSIONS AND FUTURE WORK

In this paper we proposed a simple, very efficient, and fast-to-implement reactive navigation method intended for use with multibeam LIDARs. The approach can be used for two purposes: It allows an autonomous robot to safely navigate in previously unknown static terrain. As a second option it can be used as protective shelter for systems that are based on following GPS way points, such that the method follows the GPS trajectory in case it is safe but avoids obstacles aside the track in case of GPS drift. The method is not intended to deal with moving obstacles, but the paper shows how the method can be combined with a separate dynamic obstacle tracker. Both options have been demonstrated at robotic competitions, the exploration of unknown terrain at the C-Elrob 2007, where our vehicle drove a difficult combined urban and nonurban course 90% autonomously without using any GPS (manual intervention at crossings to make the vehicle drive the correct course), and the protective shelter option implemented within the system of the DARPA Urban Challenge finalist Team AnnieWAY. At the core of the system is a set of motion primitives (called tentacles) that are used for both perception and motion execution. In contrast to other methods, our algorithm uses an ego-centered occupancy grid. In this way the geometric relation between the tentacles and the grid is static, allowing precomputation of grid addresses. This static relationship is the main reason for the efficiency of the method. However, the efficiency comes at the cost of violating vehicle dynamics in the sense that the geometric shape of the tentacles cannot be precisely executed. However, the interesting aspect is—and we despite a theoretic analysis of this aspect—that despite this violation the algorithm can ensure that the deviation of the resulting path to the tentacle is bound to be in an area that is ensured to be free of obstacles. The second reason for the efficiency of the method is that it evaluates only a small set ($n = 81$) of tentacles at a time, all having the shape of circular arcs. Despite this seeming geometric restriction, the method achieves a huge space of possible trajectories by using a speed-varying evaluation mechanism of the tentacles and chaining only small fragments of the circular arcs. The reactive method does not accumulate data and is completely data driven, thereby avoiding the SLAM problem. No global position of the vehicle or obstacles has to be estimated.

Several interesting extensions of our method are possible. The first improvement would be to consider a set of tentacles for a large number of sampled points in the state space of the vehicle dynamics. In this way, the dynamically correct shapes could be picked up in every iteration. This improvement would be at the cost only of memory, not of computational load. Further, it would be interesting to learn the tentacles of each state by observing and sampling the states and trajectories recorded while a human drives. Another interesting idea arises by observing that the classification procedure of the tentacles determines not only whether a tentacle is drivable but also the distance at which the first obstacle occurs. When driving along a conventional road with pavement edges, one can observe that the set of tentacles finds these edges well. The idea is to see the tentacles as a curb detector and try to group those obstacle positions, e.g., by verifying whether they constitute a smooth sequence of positions, in this case detecting the border of the road. Then, navigation could be done relative to the detected and tracked road boundary. In case no detection is possible, the method could fall back on the pure reactive navigation mode. In this way, reactive and cognitive navigation in relation to perceived road boundaries could be combined nicely.

ACKNOWLEDGMENTS

This research was supported in part by the German Excellence Cluster CoTeSys "Cognition for Technical Systems."

REFERENCES

Besl, P. J., & McKay, N. D. (1992). A method for registration of 3-D shapes. IEEE Transactions on Pattern Analysis and Machine Intelligence, 14(2), 239–256.

Braitenberg, V. (1984). Vehicles: Experiments in synthetic psychology. Cambridge, MA: MIT Press.

Coombs, D., Murphy, K., Lacaze, A., and Legowik, S., (2000). Driving autonomously off-road up to 35 km/h. In Proceedings of IEEE Intelligent Vehicles Symposium 2000, Detroit, MI (pp. 186–191).

Coulter, R. C. (1992). Implementation of the pure pursuit path tracking algorithm (Tech. Rep. CMU-RI-TR-92-

01). Pittsburgh, PA: Robotics Institute, Carnegie Mellon University.

Dickmanns, E. D. (1994). The 4D-approach to dynamic machine vision. In Proceedings of the 33rd IEEE Conference on Decision and Control, Lake Buena Vista, FL (volume 4, pp. 3770–3775).

Dickmanns, E. D. (2007). Dynamic vision for perception and control of motion. Berlin: Springer-Verlag.

Dissanayake, M., Newman, P., Clark, S., Durrant-Whyte, H., & Csorba, M. (2001). A solution to the simultaneous localization and map building (SLAM) problem. IEEE Transactions on Robotics and Automation, 17(3), 229–241.

Goldberg, S., Maimone, M., & Matthies, L. (2002). Stereo vision and rover navigation software for planetary exploration. In Proceedings of the IEEE Aerospace Conference, Big Sky, MT (volume 5, pp. 2025–2036).

Julier, S. J., & Uhlmann, J. K. (2001). A counter example to the theory of simultaneous localization and map building. In Proceedings of the IEEE International Conference on Robotics and Automation (ICRA) (volume 4, pp. 4238–4243).

Kammel, S. (2007). DARPA Urban Challenge 2007 Team AnnieWAY, team homepage, http://annieway.mrt.uni-karlsruhe.de.

Kammel, S., Ziegler, J., Pitzer, B., Werling, M., Gindele, T., Jagzent, D., Schröder, J., Thuy, M., Goebl, M., von Hundelshausen, F., Pink, O., Frese, C., & Stiller, C. (2008). Team AnnieWAY's autonomous system for the DARPA 2007 Urban Challenge. Journal of Field Robotics, 25(9),615–639.

Kelly, A., & Stentz, A. T. (1998). Rough terrain autonomous mobility—part 2: An active vision, predictive control approach. Autonomous Robots, 5, 163–198.

Lamon, P., Kolski, S., Triebel, R., Siegwart, R., & Burgard, W. (2006). The SmartTer for ELROB 2006—A vehicle for fully autonomous navigation and mapping in outdoor environments (Tech. Rep.). Autonomous Systems Lab., Ecole polytechnique Fédérale de Lausanne, Switzerland, Autonomous Intelligent Systems, Albert-Ludwigs-University of Freiburg, Germany.

Landau, Y. D. (1979). Adaptive control: The model reference approach. New York: Marcel Dekker.

Mitschke, M., & Wallentowitz, H. (2004). Dynamik der Kraftfahrzeuge. Berlin: Springer-Verlag.

Nilsson, N. J. (1984). Shakey the robot (Tech. Rep. 323). Menlo Park, CA: Artificial Intelligence Center, SRI International.

Preparata, F. P., & Shamos, M. I. (1985). Computational geometry: An introduction (Monographs in Computer Science). Berlin: Springer. (Corrected 5th printing 1993.)

Rosenblatt, J. (1997). DAMN: A distributed architecture for mobile navigation. Ph.D. thesis, Robotics Institute, Carnegie Mellon University, Pittsburgh, PA.

Sariff, N., & Buniyamin, N. (June 2006). An overview of autonomous mobile robot path planning algorithms. In 4th Student Conference on Research and Development, 2006. SCOReD 2006, Shah Alam, Malaysia (pp. 183–188).

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., van 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.