6

Intelligent Off-Road Navigation Algorithms and Strategies of Team Desert Buckeyes in the DARPA Grand Challenge '05

Qi Chen and Ümit Özgüner

Department of Electrical and Computer Engineering, The Ohio State University,
Columbus, Ohio 43210, USA

Summary. This paper describes one aspect of our approach in developing an intelligent off-road autonomous vehicle, the Intelligent Off-road Navigator (ION), as team Desert Buckeyes from the Ohio State University for the DARPA Grand Challenge 2005. The real-time navigation is one of the critical components in an autonomous ground vehicle system. In this paper, we focus on the navigation module, whose main responsibility is to generate smooth and obstacle-free local paths with appropriate speed setpoints. For the smooth path generation, we introduce a polynomial interpolation method. To generate obstacle-free paths, a steering controller utilizing a fuzzy obstacle avoidance algorithm is presented. A speed fuzzy controller is introduced to generate the speed setpoints. These two fuzzy controllers collaborate with each other to guide our vehicle ION to the goal safely. The obstacle avoidance algorithm proposed in this paper was also tested in simulations and on small robots successfully. Other issues related to the navigation module are discussed in the paper as well, such as the vehicle's system structure and its finite state machine. As a result, ION achieved great performance in the National Qualification Event (NQE), covered about 30miles in the Nevada Desert with complete autonomous operations, and finished 10th in the Grand Challenge 2005.

Keywords: autonomous ground vehicle, Grand Challenge, real-time, navigation, finite state machine, obstacle avoidance, fuzzy controller.

6.1 Introduction

This paper describes our approach in developing the navigation module of the vehicle ION, the Intelligent Off-road Navigator, as team Desert Buckeyes from the Ohio State University for the DARPA Grand Challenge 2005 (GC05).

GC05 was a competition for off-road autonomous ground vehicles (AGVs). About three thousands waypoints (GPS coordinates) were provided shortly before the race, the AGVs were required to follow the waypoints one by one safely, continuously, smoothly and fast across natural terrain en route to the finish line without any human interference. The challenge was indeed "grand" because the AGVs had to respond to the dynamically changing environment in a timely way

M. Buehler, K. Iagnemma, and S. Singh (Eds.): DARPA'05, STAR 36, pp. 183–203, 2007.
springerlink.com

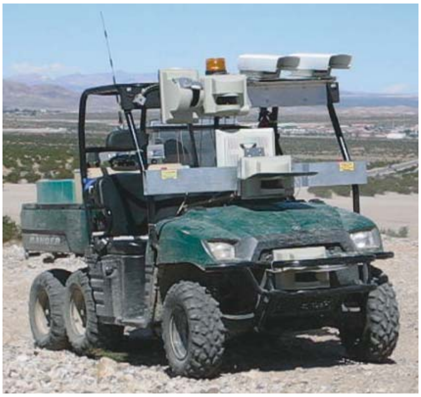


Fig. 6.1. Vehicle ION: the Intelligent Off-road Navigator. The vehicle is a 2005 Polaris Ranger 6x6. It is 120 inches long and 60 inches wide and its height is 78 inches. Drive by wire capability has been added to the vehicle so that computer control is possible for throttle, brake, steering control, and transmission gear switching.

and the data acquired was both complex and full of erroneous information. The total distance in GC05 was about 132miles.

Developing an autonomous vehicle to traverse the desert means solving a series of problems and developing a series of technologies, such as sensor fusion, navigation, artificial intelligence, vehicle control, signal processing, drive-by-wire technology, reliable software and mapping (Toth et al., 2006), etc. ION was developed in partnership with University of Karlsruhe, which developed the image processing system (Hummel et al., 2006). ION is a 6 wheeled vehicle with full drive-by-wire capability. A set of sensors (LIDARs, radars, cameras and ultrasonic transducers) and a GPS and IMU, the internal measurement unit, provide extensive sensing capability, shown in Figure 6.1. A sophisticated sensor fusion system (Redmill, Martin, & Özgüner, 2006) was developed and used with a complex intelligent analysis, decision and control configuration (ION, 2005). The overall design was quite similar to the one we developed for TerraMax in GC04 (Chen, Özgüner, & Redmill, 2004).

In this paper, we focus on the navigation module whose task is to guide ION toward the goal without colliding with obstacles. Our strategy is to develop a finite state machine that decides the state of ION so that the proper drive mode is selected in different situations. By perceiving both nearby environment and ION's status, the navigation module generates commands for the low-level controller. The command can be, for example, a set of GPS route points called *pathpoints* and the *speed setpoint*, or a sequence of motion specifications called *robotic unit operations*, or some *direct commands* such as "stop-and-wait". The finite state machine design and the obstacle avoidance algorithm utilizing fuzzy logic are introduced.

As the Desert Buckeyes was the team that developed the sensing and intelligence for the 2004 TerraMax, a number of aspects of ION were descendants of technology and approaches used in the 2004 Grand Challenge (Chen et al., 2004), (Yu, Chen, & Özgüner, 2004), (Özgüner, Redmill & Broggi, 2004), (Chen & Özgüner, 2005). We also have developed autonomous vehicles for highway driving, which have shown great performance in the structured environments. (Özgüner, Hatipoglu, & Redmill, 1997), (Redmill & Özgüner, 1998), (Hatipoglu, Özgüner, & Redmill, 2003). As far as off-road is concerned, the environment being unstructured, the navigation module and the obstacle avoidance algorithm proposed in this paper is desired to deal with more complicated situations.

The remainder of this paper is organized as follows. Section 6.2 briefly describes our system structure and general bases. The design of the finite state machine is described in Section 6.3. Section 6.4 introduces navigation module with the path planning algorithm that generates collision-free paths and appropriate speed set-points for the vehicle. ION's performances are presented in Section 6.5. Section 6.6 concludes the paper.

6.2 System Structure

The system structure of ION is illustrated in Figure 6.2. Environment sensor fusion, vehicle ego-state sensor fusion, high-level controller and low-level controller are ION's four major modules. ION is equipped with several cameras, LIDARs, ultrasonic Sonars and a radar. A sophisticated sensor fusion system generates a local grid map that moves with the vehicle. The local map contains the sizes and positions of nearby obstacles. ION's ego states are fused from a GPS and an IMU. The navigation module, also called the high-level controller, obtains information from the sensor fusion modules and generates a series of pathpoints and a speed set point for the low level controller. In some situations, the navigation module sends direct control command to the low level controller, such as "Stop" and specific "robotic unit operations". Obtaining the commands from the high-level controller, the low-level controller enables ION to follow the given route defined by a series of pathpoints at the given speed, as has been done in (Redmill, Kitajima, & Özgüner, 2001).

6.3 Finite State Machine

Since ION was expected to encounter very complicated situations, we developed a finite state machine (FSM) in ION's navigation module to deal with difference situations. Based on the information collected from the sensor fusion modules and the inner state monitoring, the navigator module determines the right machine state for ION to activate the corresponding subsystem/algorithm so that the proper command is generated.

The FSM we implemented in ION is represented conceptually in Figure 6.3 . There are two major states, the "*Path-Point Keeping*" state and the "*Obstacle*

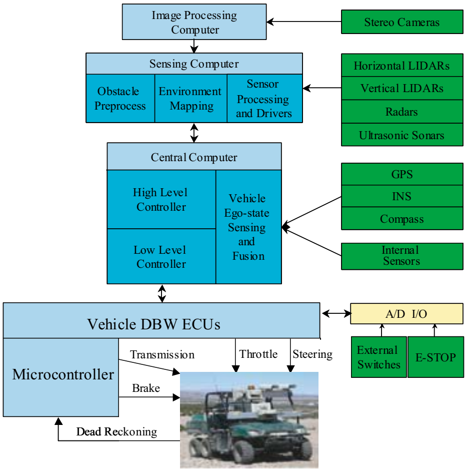


Fig. 6.2. The System Structure of ION

Avoidance" state. ION stays in these two states at the most of racing time, when there are no special events, such as tunnel, sensor failure, and narrow path, etc, detected. When ION is in one of these two states, the pathpoints are generated by the path planning algorithms as described in the following section. The smooth path generation algorithm is for the "*Path-Point Keeping*" state, while the obstacle avoidance algorithm is used in the "*Obstacle Avoidance*" state.

In the FSM, the other states are also very essential for ION to deal with special events that the vehicle may encounter. The "*Tunnel Mode*" state is designed specifically for the situation when ION is in a tunnel where GPS signals are lost temporarily. We assume ION encounters no obstacles in the tunnels and the FSM switches only to the "*Path-Point Keeping*" state from this state. In the "*Tunnel Mode*" state, the distances to both sides of the vehicle are measured by the ultrasonic transducers mounted on ION so as to keep the vehicle in the center of the tunnel. The FSM gets into the "*Robotic Operations*" state when ION needs to adjust its heading or position or both, for example, when narrow paths and very sharp turns, i.e. direct forward path would contact obstacles, are encountered. In this state, a sequence of back-and-forth operations are executed to adjust the ION's status. When there are sensor failures, the FSM transits into the "*Alarm*" state to deal with the malfunction. In this state, the navigation module executes the sensor resetting command to recover the failed sensor or sensors. In the case that the sensor failure is unrecoverable, a flag is received from the sensor fusion module and the FSM switches back to the "*Path-Point Keeping*" state. Carrying these flags, the navigation module reduces the maximum allowed speed to reduce the risk of collision, and the sensor fusion module adjust the coefficients correspondingly as well. The "*Road Following*" state is designed for the situation when the image processing module detects a road. For the situation when ION

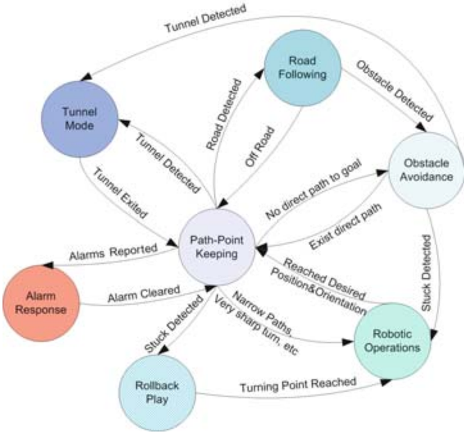


Fig. 6.3. ION's finite state machine of the navigation module

is stuck on the road, the FSM switches to the "*Rollback*" state. In this state, ION drives back along the trajectory it records till the point where another path can be initiated.

We also introduced a watch-dog to prevent the vehicle from being stuck forever. The vehicle might be stuck because the robotic operations couldn't adjust vehicle status amidst obstacles in six tries, or false obstacles block the path totally, or because of some other unexpected events. Anyway, the watch-dog was not designed to deal with normal situations. When ION's position does not change or the FSM rests in a state other than the "*Path-Point Keeping*" or "*Obstacle Avoidance*" state for a certain period of time, the FSM automatically resets to the "*Path-Point Keeping*" state and stays in this state regardless of the obstacles and other events until ION reaches a certain distance. During the FSM resetting process, a high throttle value is sent to the lower level controller so that ION can possibly get out of holes or run over some obstacles like small bushes. With the watch-dog design, ION might be able to get out of stuck in many cases and gain chances to continue.

6.4 Navigation Module

ION's navigation module, also called the high level controller, plans or replans the local path when the machine state is at the "*Path-Point Keeping*" or "*Obstacle Avoidance*" state. The navigation module checks the status of the local path, which consists of 10 pathpoints, at 10Hz. If the path comes across obstacles in the local map, or ION has reached the 5th point of the local path, or ION is indicated passing a waypoint, the navigation module then generates a new local path. Figure 6.4 shows the procedure to generate the local path. There are two path generation algorithms implemented in the navigation module: the smooth path generation algorithm and the obstacle avoidance algorithm.

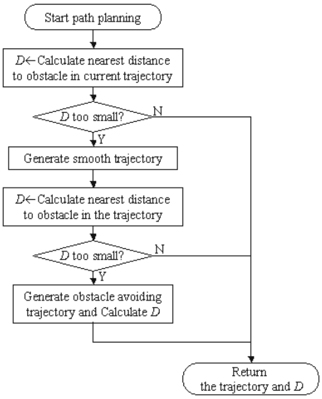

Start path planning
D← Calculate nearest distance to obstacle in current trajectory
D too small?
N
Y
Generate smooth trajectory
D← Calculate nearest distance to obstacle in the trajectory
D too small?
N
Y
Generate obstacle avoiding trajectory and Calculate D
Return the trajectory and D

Fig. 6.4. The flow chart for path planning

6.4.1 Smooth Path Generation

A smooth path, whose curvature is continuous, is generated to connect the waypoints. Let $\{P_i,\ i = 0, 1, 2, \cdots\}$ denote the coordinates of the waypoints, where $P_i := (x_i,\ y_i)^T \in \mathbf{R}^2$. To generate a path, $\{P_i(s),\ 0 \le s \le 1\} \subset \mathbf{R}^2$, connecting P_i and P_{i+1}, a polynomial interpolation is applied:

$$P_i(s) = A_0(s)P_{i-1} + A_1(s)P_i + A_2(s)P_{i+1} + A_3(s)P_{i+2}, \quad 0 \le s \le 1 \qquad (6.1)$$

where $A_i(\cdot)$'s are scalar polynomial coefficient functions. According to the theory of planar curves (Hsiung, 1998), for a certain $s \in [0,\ 1]$, the vector tangent to the curve and associated at the point $P_i(s)$ is defined by

$$\begin{aligned} T_i(s) =& \lim_{h\to 0} \frac{1}{h}(P_i(s+h) - P_i(s)) \\ =& A_0'(s)P_{i-1} + A_1'(s)P_i + A_2'(s)P_{i+1} + A_3'(s)P_{i+2} \end{aligned} \qquad (6.2)$$

where $A_i'(\cdot)$ is the derivative of function $A_i(\cdot)$. Define the unit tangent vector as $\mathbf{u}_i(s) := T_i(s)/\|T_i(s)\|$. $A_i(\cdot)$'s are selected by satisfying

$$\begin{cases} P_i(0) = P_i \\ P_i(1) = P_{i+1} \\ \mathbf{u}_{i-1}(1) = \mathbf{u}_i(0) \end{cases} \qquad (6.3)$$

According to (6.1), $\{P_i(s),\ 0 \le s \le 1\}$ is a planar curve determined by four waypoints $(P_{i-1},\ P_i,\ P_{i+1},\ P_{i+2})$. For curve $\{P_{i-1}(s),\ 0 \le s \le 1\}$, it is generated by $(P_{i-2},\ P_{i-1},\ P_i,\ P_{i+1})$ and connects P_{i-1} and P_i. These two curves have one common point, P_i. By satisfying $\mathbf{u}_{i-1}(1) = \mathbf{u}_i(0)$ in (6.3), the unit tangent vector

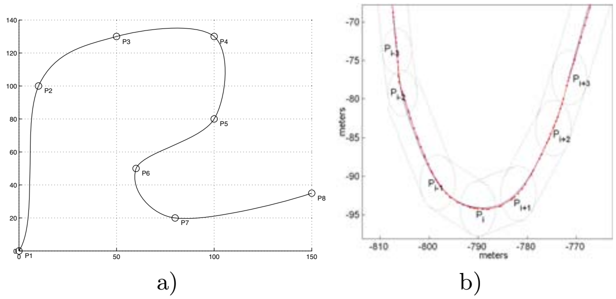


Fig. 6.5. Smooth path generation examples: a) A smooth path generation example: the curve is generated by equation (6.1) with the coefficient polynomials defined in equation (6.4); b) An example of smooth path generation: The solid line with dots (red) is the the planned path that ION generated in the NQE, GC05. The path smoothly connects the waypoints, which are located at the center of each circle. The solid curve (blue) is the trajectory that ION traveled by following the planned path. These two curves almost overlap, which shows ION followed the planned path very well.

of the curve $\{P_{i-1}(s)\}$ is the same as that of the curve $\{P_i(s)\}$ at the point P_i. We say these two curves are smoothly connected.

In ION, we select $A_i(\cdot)$'s in (6.1) as follows:

$$\begin{cases} A_0(s) = (-s + 2s^2 - s^3)/2 \\ A_1(s) = (2 - 5s^2 + 3s^3)/2 \\ A_2(s) = (s + 4s^2 - 3s^3)/2 \\ A_3(s) = (-s^2 + s^3)/2 \end{cases} \tag{6.4}$$

Applying the coefficient polynomials (6.4) to (6.1), the result satisfies the constraints in (6.3) so that the path generated for ION is smooth. For example, given eight points, $\{P_1, \cdots, P_8\}$, arbitrarily in a map, the above method generates a smooth path connecting them. The result is shown in Figure 6.5(a). In ION, we use cubic polynomials for $A_i(\cdot)$'s. The selection of the order of the polynomials is somewhat arbitrary and a third order polynomial satisfies the given constraints. With higher order polynomials that satisfy (6.3), we can obtain similar results. Figure 6.5(b) shows part of the trajectory that ION generated in the NQE and it followed the smooth path very well.

6.4.2 Obstacle Avoidance

In the case that the smooth path comes across some obstacles, the FSM stays in the "*Obstacle Avoidance*" state and a collision-free path is generated by the obstacle avoidance algorithm.

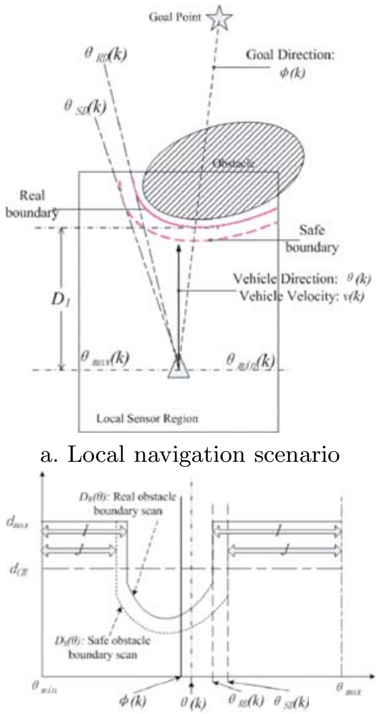
Since the late 1970s, extensive effort has been exerted to develop obstacle avoidance algorithms, see (Latombe, 1991) and references therein. The research can be classified into two major areas: the global path planning (Kambhampati & Davis, 1986), (Hwang & Ahuja, 1988), (Warren, 1989) and the real-time local motion planning (Khatib, 1985), (Borenstein & Koren, 1989), (Adams & Probert, 1990), (Stentz, 1994). In GC04 and GC05, the race routes were described with road definition data that specified the GPS coordinates, width and speed limit for each section. The provided data was quite dense and the global path planning methods were not needed in those events. On the other hand, the local motion planning methods dynamically guide the AGV according to the locally sensed obstacles, which requires less prior knowledge about the environment. The fuzzy controller described in this paper is a real-time local motion planning method, which is more suitable and practical for ION in the Grand Challenge events, since ION senses only the nearby obstacles.

Fuzzy logic navigation methods have been studied and implemented in small robots, rovers and AGVs with small turning radius, see (Saffiotti, 1997), (Hodge & Trabia, 1999), (Seraji, 2000), (Lee, Lai, & Wu, 2005) and references therein. As noted by Saffiotti (Saffiotti, 1997), fuzzy logic has the feature to make it a useful tool to cope with the large amount of uncertainty that is inherent in natural environments. Most of these approaches select the direction by weighting the effort of *target-approach* and the need of *obstacle-avoidance*. However, few fuzzy logic approaches concern the dynamic and kinematic constraints of a big AGV like ION. For ION, if the two consecutive steering decisions are totally opposite, such as left turn and right turn, the steering decisions would neutralize with each other. Therefore, frequent jumps between two consecutive steering decisions should be avoided. Also, since the turning radius of ION is considerably large, it is preferable to respond to obstacles before being too close to them.

In the remainder of this subsection, we introduce two heterogeneous fuzzy controllers for ION's obstacle avoidance path planning system. The steering controller emphasizes goal reaching and plays an important role in obstacle avoidance simultaneously. Fuzzy rules are proposed for ION to approach the goal and avoid obstacles at the same time. The speed controller prevents collisions with obstacles. The rules in each fuzzy controller are defined for reacting in different situations and each rule represents a certain behavior character based on human drivers' knowledge. These two fuzzy controllers collaborate to direct ION to the goal without collision with obstacles.

6.4.2.1 Steering Fuzzy Controller Design

Figure 6.6 shows one example scenario that explains the basic steering rules. Based on the local sensor ability, only the boundary of the obstacle in the local sensor region is detectable. The shape and size of the whole obstacle outside of the sensing region are not known. Practically, we expand the boundary of the obstacle by half the size of the vehicle so that we can consider the vehicle as a mass point. The expanded obstacle boundary is called the "real boundary", henceforth. Furthermore, to ensure the safety of ION, the obstacles is further



a. Local navigation scenario

b. The scanning distance to the obstacle's boundaries at time k

Fig. 6.6. An example scenario. In the figures, $[\theta_{min},\ \theta_{max}]$ is the scanning angle range of interest; θ_{RD} and θ_{SD} are the decision angle based on the real boundary and safe boundary, respectively. D_1 is the distance to the obstacle in the current heading direction. d_{CR} is the criterion distance. d_{max} is the maximum scanning distance.

enlarged to create a "safe boundary", so that the real boundaries are enveloped with the safe boundaries. By the two-step obstacle extension, each obstacle has two different boundaries, the "real boundary" and the "safe boundary". Figure 6.6 illustrates the boundaries and the scanning distances.

The obstacle avoidance algorithm to be presented in this subsection originates from a heuristic non-fuzzy algorithm which selects the direction based on the distance to obstacles and the difference to the goal direction. The original algorithm works well when there are few obstacles in the environment and the obstacles are all convex-shaped. However, it has deficiencies in some special situations, such as candidate set being empty, or selecting wide left or right instead of finding a path in the front, or being stuck at some obstacle compositions, or the vehicle swinging toward an obstacle.

In order to overcome these shortcomings, four fuzzy rules are developed one by one and together they work very well. These fuzzy rules are defined to reshape the scanning distances curve and then the steering strategy is to select the direction with the smallest angle difference to the goal direction and enough distance to the obstacles. The formulas are expressed as follows:

$$\begin{aligned}
\mathcal{I}(k) &= \{\theta | D_R(\theta) - \varDelta(\theta) > d_{CR,R}(k)\} \\
\mathcal{J}(k) &= \{\theta | D_S(\theta) - \varDelta(\theta) > d_{CR,S}(k)\} \\
d_{CR,R}(k) &= \min\{d^0_{CR,R}, \max_\theta\{D_R(\theta) - \varDelta(\theta)\} - \epsilon\} \\
d_{CR,S}(k) &= \min\{d^0_{CR,S}, \max_\theta\{D_S(\theta) - \varDelta(\theta)\} - \epsilon\}
\end{aligned} \tag{6.5}$$

and

$$\begin{aligned}
\theta_{RD}(k) &= \arg\min_{\theta \in \mathcal{I}}\{|\theta - \phi(k)|\} \\
\theta_{SD}(k) &= \arg\min_{\theta \in \mathcal{J}}\{|\theta - \phi(k)|\} \\
\theta_D(k) &= Select(\theta_{RD}(k), \theta_{SD}(k), \phi(k))
\end{aligned} \tag{6.6}$$

where $\mathcal{I}$ and $\mathcal{J}$ are the candidate sets of angles at which the distance to the obstacle is longer than the criterion distance, $d_{CR,R}(k)$ and $d_{CR,S}(k)$, respectively. $\varDelta(\cdot)$ reshapes the obstacle scanning distances curves, $D_R(\cdot)$ and $D_S(\cdot)$, so that different directions have different priorities to be selected from. The rule that decides $\varDelta(\cdot)$, called the the "Focus" rule, is described later. $d^0_{CR,R}$ and $d^0_{CR,S}$ are design parameters to prevent the algorithm from being sensitive to obstacles very far away, known as the far-sighted situation, where there may be many false obstacles. ϵ is a small positive constant so that the candidate sets are both nonempty.

Equations (6.5) and (6.6) give the method to find two directions, $\theta_{RD}(k)$ and $\theta_{SD}(k)$, as decisions based on the real boundary and the safe boundary scanning distances, respectively. Both decisions attempt to reach the goal point and keep the obstacle away for certain distances, $d_{CR,R}(\theta, k)$ and $d_{CR,S}(\theta, k)$, respectively. When ION is far away from the obstacles, $d_{CR,R}(k) = d^0_{CR,R}$. Otherwise, when ION is close to the obstacles, the steering controller chooses the direction with the longest distance to the obstacles. In other words, the direction toward open space is selected.

A. The Selection Rule

Let $\theta_D(k)$ be the final decision direction at time k. It is selected from either $\theta_{SD}(k)$ or $\theta_{RD}(k)$ by calculating $|\theta_{SD}(k) - \theta_{RD}(k)|$ and $|\theta_{SD}(k) - \phi(k)|$, where $\phi(k)$ is the goal direction. Thus, the steering strategy reaches the goal and avoids obstacles at the same time. Table 6.1 shows the selection rule noted by "*Select*" in (6.6).

The "*Small*", "*Medium*" and "*Large*" in Table 6.1 are fuzzy descriptions. The defuzzification function picks either $\theta_{SD}(k)$ or $\theta_{RD}(k)$, as the result of $\theta_D(k)$. This ensures that the result would not be some value between $\theta_{SD}(k)$ and $\theta_{RD}(k)$.

Remark 1. *If $\theta_{SD}(k)$ is chosen to be the steering decision, then there is an open space in the direction so that vehicle is allowed to pick up a relatively high*

Table 6.1. The selection rule

		$\lvert\theta_{SD}(k) - \theta_{RD}(k)\rvert$		
	$\theta_D(k) =$	Small	Medium	Large
$\lvert\theta_{SD}(k) - \phi(k)\rvert$	Small	$\theta_{SD}(k)$	$\theta_{SD}(k)$	$\theta_{RD}(k)$
	Medium	$\theta_{SD}(k)$	$\theta_{RD}(k)$	$\theta_{RD}(k)$
	Large	$\theta_{SD}(k)$	$\theta_{RD}(k)$	$\theta_{RD}(k)$

speed safely. On the other hand, if $\theta_{RD}(k)$ is chosen, it can be concluded that $|\theta_{SD}(k) - \theta_{RD}(k)|$ is large, i.e. the $\theta_{SD}(k)$, presenting the open free space, is in the other direction, and the chosen direction must be a narrow path. The speed controller uses this information and sets a relatively low speed setpoint.

B. The Focus Rule

In order to prevent ION from being distracted by side directions and going off the road, we introduce the "Focus" rules in the steering controller. In the focus rules, we consider the directions pointing forward and leading to the goal point to have higher priorities than others.

In (6.5), $\varDelta(\cdot)$ is introduced to reshape the scanning distances curves, $D_R(\cdot)$ and $D_S(\cdot)$. When a direction θ has low priority, the $\varDelta(\theta)$ is then set to a large value so that the direction θ is less likely to be in the candidate sets, $\mathcal{I}$ and $\mathcal{J}$. By doing so, the steering controller tends to find the path direction with high priority.

Let $\varDelta 1(\theta)$ and $\varDelta 2(\theta)$ represent the $|\theta - \theta(k)|$ and $|\theta - \phi(k)|$, respectively, where $\theta(k)$ is the vehicle heading at time k. So, a direction with high priority has a small values of $\varDelta 1$ and $\varDelta 2$. Let (S, MS, M, ML, L) represent (*Small, Medium Small, Medium, Medium Large* and *Large*), respectively. The fuzzy "Focus" rules that work on the $\varDelta(\theta)$ in the equation (6.5) are as follows:

- If ($\varDelta 1$ is S) and ($\varDelta 2$ is S), then ($\varDelta$ is S): if θ points forward and leads to the goal direction, then θ has the highest priority to be chosen, i.e. $D_R(\theta)$ or $D_S(\theta)$ has the *smallest* deduction.
- If (($\varDelta 1$ is S) and ($\varDelta 2$ is M)) or (($\varDelta 1$ is M) and ($\varDelta 2$ is S)) or (($\varDelta 1$ is M) and ($\varDelta 2$ is M)), then ($\varDelta$ is MS).
- If (($\varDelta 1$ is S) and ($\varDelta 2$ is L)) or (($\varDelta 1$ is L) and ($\varDelta 2$ is S)), then ($\varDelta$ is M).
- If (($\varDelta 1$ is L) and ($\varDelta 2$ is M)) or (($\varDelta 1$ is M) and ($\varDelta 2$ is L)), then ($\varDelta$ is ML).
- If ($\varDelta 1$ is L) and ($\varDelta 2$ is L), then ($\varDelta$ is L): if the angle θ neither points forward nor leads to the goal direction, it has low priority and the deduction is *large*.

Figure 6.7 is an example of how the "Focus" rules work. The $-\varDelta(\theta)$ forms a $\varLambda$-shaped curve when $\theta(k)$ is close to $\phi(k)$. Note that the curve can be an M-shaped curve when $\theta(k)$ is separated away from $\phi(k)$. For those θs that $\varDelta(\theta) > D_R(\theta)$, we have $\theta \notin \mathcal{I} \cup \mathcal{J}$ according to the focus rules. Therefore, these directions are

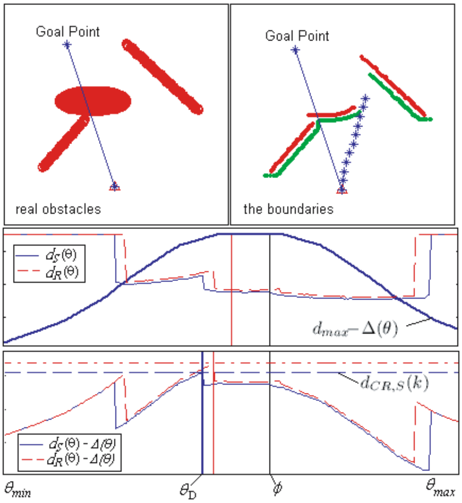


Fig. 6.7. The example of the focus rules: The upper left figure is the "real world" and the upper right figure displays the obstacle boundaries, both real and safe boundaries. The scanning distances to the boundaries are shown in the middle figure. The $d_{max} - \Delta(\theta)$ is a Λ-shaped curve shown in the figure. The adjusted real distance and safe distance curves are plotted in the bottom figure. By selecting the direction θ_D as the steering decision, the steering controller generates a curve of path points shown in the upper right figure, following which the AGV avoids the obstacles and approaches the goal point.

excluded from being selected as the decision direction. By doing so, the steering controller opens a *priority window* for those directions that point forward and lead to the goal point. Thus, the steering controller ignores the distractions from low priority directions. A direction with low priority is selected only when the directions with higher priority are all blocked by obstacles.

In this fuzzy method, triangle membership functions are used to define the fuzzy sets for the $\Delta 1$ and $\Delta 2$, as shown in Figure 6.8. The values assigned to (S, MS, M, ML, L) determine the height of the hump. The larger the values are, the higher the hump is. The values of (a, b, c, d) determine the membership function, consequently determine the width of the hump in the curve of $-\Delta(\theta)$. if the values are small, $-\Delta(\theta)$ is not reduced only in a narrow range of directions. Thus, the "Focus" effort is enhanced. On the other hand, if the values are large, then the "Focus" effort is neutralized and the navigator has a wide range of direction to select from.

C. The Focus vs. Search Rules

The focus rules prevent the distraction by side directions quite well. Nevertheless, the focus rules trade the ability in searching the feasible directions for the

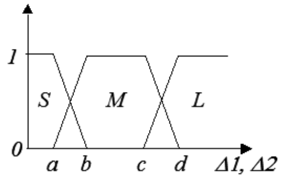


Fig. 6.8. The membership functions for $\Delta 1$ and $\Delta 2$

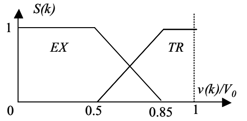


Fig. 6.9. The membership functions for $\Delta 1$ and $\Delta 2$

distraction prevention. It works well when there are few obstacles in the environment. When there are many obstacles in the environment, however, the focus rules impair the search capability of the navigator. Furthermore, as stated previously, the values of (a, b, c, d) in the membership function of the fuzzy sets of $\Delta 1$ and $\Delta 2$ are important parameters of the focus rules. The smaller values of (a, b, c, d) correspond to stronger "Focus" and vice versa.

Motivated by this, the "Focus" rules are improved by considering the fuzzy evaluation of the AGV status. The $v(k)$, the vehicle's speed, is regarded as the index of vehicle's situation: the lower the speed is, the more search capability is required. Therefore, the equations are:

$$\begin{aligned} S(k) &= F_S(v(k), V_0) \\ [a(k)\ b(k)\ c(k)\ d(k)]^T &= F(S(k)) \end{aligned} \tag{6.7}$$

where $S(k)$ is the situation classification, a membership function value of the fuzzy sets: *exploring* and *travelling*.

The membership function is shown in Figure 6.9. The rule $F_S(\cdot)$ states: When the $v(k)/V_0$ is small, $S(k)$ is "*exploring*" (EX). On the other hand, when $v(k)/V_0$ is large, $S(k)$ is "*travelling*" (TR). The AGV speed, $v(k)$, is classified by comparing rather to the base speed than to the absolute values. The base speed, V_0, is provided to the AGV as the current speed limit. In the "Focus vs. Search" rules, the values of (a, b, c, d) are no longer fixed. The fuzzy rules in (6.7) are as follows:

- If $S(k)$ is EX, then the values of (a, b, c, d) increases: When the AGV is nearing obstacles, we assume that the speed controller slows down the AGV. In this case, the $S(k)$ becomes more "EX", and the focus rules are neutralized. In other words, the "Search" capability is enhanced.
- If $S(k)$ is TR, then the values of (a, b, c, d) reduces: vice versa, the focus rules are enhanced in this case.

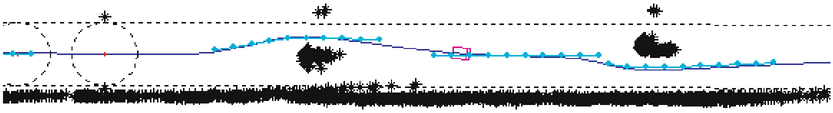


Fig. 6.10. The example of ION's path planning in the two-car-passing section at NQE, GC05. The black stars are the obstacles recovered from ION's sensor logs. Two cars were placed in a corridor of 15ft half-width, as shown in the figure. The distance between the two cars was about 100ft. ION passed the two cars, from left to right in the figure, at the speed of 10 mph, the highest speed permitted in the section.

By applying the rules in (6.7), the AGV is much less likely to be stuck in front of the concave-shaped obstacles. When the AGV approaches the obstacles, the speed is reduced so that the "priority window" is opened up for search and the "focus" effect is neutralized. The "Search" rules help to prevent the AGV from being stuck.

D. The Persistence Rules

Because of the existence of false obstacles, some obstacles may popup or disappear suddenly in the local sensor map. This brings a problem that the *discontinuous* outputs from the steering controller causes the oscillation of the vehicle's heading. In the worst situation, the two consecutive steering decisions may neutralize each other.

To solve this problem, the value of $\theta_D(k-1)$ is introduced to adjust $\varDelta(\theta)$ in the focus fuzzy rules. By doing so, the steering controller adjusts the searching window and raise the priority of directions near to the former decision, $\theta_D(k-1)$. Therefore, the steering controller maintains the persistence. The "Persistence" rules weigh on the last decision and give the controller a characteristic of persistence so that the zigzag performance of the AGV is prevented.

When the decision direction is selected, a sequence of pathpoints is generated for the low-level controller. Figure 6.10 shows the path planning result of ION in the two-car-passing section in the NQE, GC05. The dotted lines are the section boundary, the solid line is the trajectory that ION runs and the dark stars are the detected obstacles. Three sections of the 10-dot-curve are the planned path, which avoids the obstacles and stays within the section boundaries. Most planned paths are not shown in order to make the figure clear. ION's trajectory and planned paths almost overlap, which shows that our low level controller follows the planned path very well.

6.4.2.2 Speed Fuzzy Controller Design

The speed controller's main aim is to avoid the collision into the obstacles. Moreover, due to the physical constraint of the vehicle, sharp turning at high

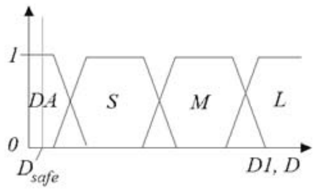


Fig. 6.11. The membership functions for $D1$ and D

speed should be avoided to prevent the AGV from rolling over. The velocity fuzzy controller in ION consists of two sets of rules: the *Anti-collision rules* and the *Safe-steering rules*, shown in the following:

$$\begin{aligned} V_s(k) &= \min[V_0, A(D1, D), T(\varDelta\theta)] \\ \varDelta\theta &= |\theta_D(k) - \theta(k)| \end{aligned} \tag{6.8}$$

Where $D1$ represents the distance to obstacles in the front of vehicle and D is the value returned from path-planning procedure as in Figure 6.4. $\varDelta\theta$ is the angle difference between the decision angle and current heading. V_0 is the speed limit allowed for ION. The final speed set point value is $V_s(k)$.

A. The Anti-collision Rules

The main purpose of the collision rules is to prevent the collision. Let A represent the anti-collision rules. The fuzzy rules, A, have two inputs, $D1$ and D. Let (S, MS, M, ML, L) have the same meaning as stated above, and let (DA) denote the fuzzy set of dangerous distance. The rules are stated as follows:

- If (one of $(D1, D)$ is DA), then A is $STOP$: the vehicle should stop when the distance to the obstacles is too close and it is dangerous to move on.
- If ($D1$ is S), then A is S: the vehicle speed should be reduced when it is close to the obstacles.
- If ($D1$ is M and D is S), then A is MS.
- If ($D1$ is M and D is M), then A is M.
- If (one of $D1$ $(D1, D)$ is L) and (the other is M), then A is ML.
- If (both of $(D1, D)$ are L), then A is L: no obstacle is nearby, the speed is set to a high value.

The boundaries of the fuzzy set is selected according to the dynamic performance of the vehicle and how much risk you want to take. The membership functions used in our system are shown in Figure 6.11. The membership functions for $D1$ and D are the same in our system, although they could be different in the boundaries of the fuzzy sets. Note that the D_{safe}, the minimum safe distance, is marked in the figure, which is totally within the fuzzy set DA. Therefore, the AGV stops before reaching the distance D_{safe} so that collisions are prevented. Should the vehicle be stopped by the rules, for example it is dangerously close to an obstacle, the machine state would switch to the "Rollback" state and wait to be recovered by a sequence of robotic operations. In that case, the vehicle's heading is adjusted for the next try.

B. The Safe-Steering Rules

The other issue is the safety in making a turn. If a high speed set point and the sharp-steering command are sent to the AGV at the same time, it may cause the AGV to roll over. More importantly, it has been observed in our experiments that the low level controller would have some overshoot in path following during sharp turns, thus may cause the collisions with obstacles even if the planned paths are obstacle-free.

To prevent the danger and reduce the overshoot in path following, the safe-steering rules, denoted as T, are introduced. We adopt the value of $|\theta_D(k) - \theta(k)|$, say $\Delta\theta$, to represent the steering command. The safe-steering rule T says: If ($\Delta\theta$ is S) then T is L; If ($\Delta\theta$ is M) then T is M; If ($\Delta\theta$ is L) then S is L. By doing so, the speed set point of the AGV is reduced when the sharp steering command occurs.

6.5 Performance of the Navigation Module

A simulator has been built to test the navigation algorithm designed in this paper. In the simulator environment, the obstacles are already expanded, and the vehicle is regarded as a point mass. The vehicle model in this simulator is commonly known as the Dubins' car model with a minimum turning radius. The kinematic equations are written as:

$$\begin{aligned} \dot{x} &= u \cos\theta \\ \dot{y} &= u \sin\theta \\ \dot{u} &= a \\ \dot{\theta} &= \omega \end{aligned} \tag{6.9}$$

where x and y are the position coordinates. θ is the yaw angle. u stands for the linear velocity, which is assumed to be positive. a is the acceleration. The ω is the angular velocity, a control variable. The model is subject to the constraint:

$$\left|\frac{\omega}{u}\right| \leq \frac{1}{R} \tag{6.10}$$

so that a minimum turning radius R is imposed. In this simulator, we can randomly generate obstacles over a terrain and arbitrarily select a series of check points. Figure 6.12 shows the AGV trajectory over such a terrain. The trajectory of the AGV, the smooth (blue) line in the figure, indicates that the AGV followed the check points, avoided all obstacles and reached the final goal smoothly and safely in the end. The navigation module has been tested with different obstacle densities over 100 times and failed in less than five cases and two of which didn't have feasible path since the obstacles are generated randomly. In the other three

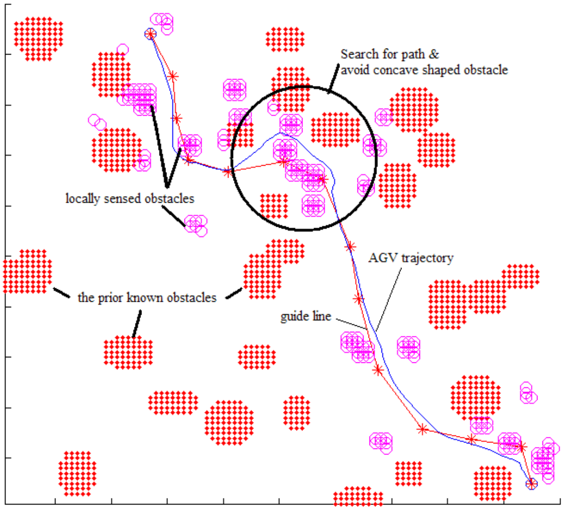


Fig. 6.12. The AGV trajectory over the obstacle-occupied terrain. The AGV is shrunk to a point. The "locally sensed obstacles" are detected by the vehicle and plotted on the map only when they are in the vehicle's sensing range. As a consequence, only those previously unknown obstacles that are near the vehicle trajectory are displayed in the result map as the "locally sensed obstacles".

failed cases, the AGV was stuck because the check points misled the AGV into a wrong fork, which shows that the obstacle avoidance algorithm in this paper is not sufficient in finding a path through a very complicated environment. Nevertheless, the algorithm is sufficient for the GC events.

Figure 6.13 shows the test on small robots. In the indoor robot test, we put an obstacle just behind a gate formed by two obstacles. In this case, even if the obstacles were close to each other, the navigation algorithm was still able to find a path and navigate the robot along the corridor without touching any obstacles. We tested the navigation module for over 20 times on small robots with different obstacles positions and configurations. The small robot failed twice to get to the goal because of the extreme closeness of certain obstacles.

The performance of the navigator designed in this paper was demonstrated by ION in the DARPA GC05 event. ION completely traversed the NQE[1] course successfully four times[2], which fully exhibited the obstacle-avoidance and goal-approaching capabilities of the navigation module and our AGV system. Figure 6.14 shows ION successful passing a car in the NQE.

[1] The National Qualification Event (NQE) in the 2005 GC was the 2.7 mile test track DARPA used as the semifinals. The track provided a series of obstacles and the AGVs are required to go through 51 "gates". It also had a 100ft metal "tunnel" where GPS was lost.

[2] ION tried the NQE course five times in total and only failed once due to a mechanical problem.

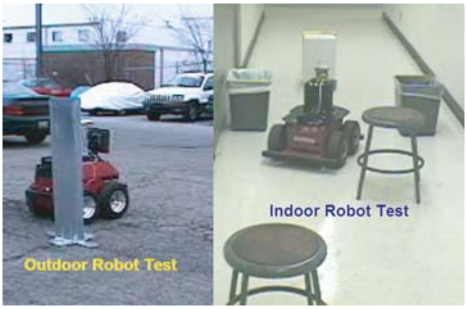


Fig. 6.13. Test obstacle avoidance algorithm on small robots. We used an all-terrain-robot-vehicle (ATRV) for outdoor tests and a PIONEER P3-AT robot for indoor tests.

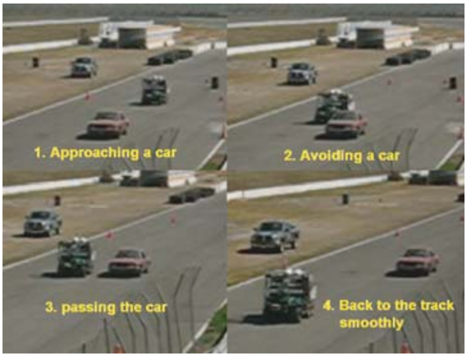


Fig. 6.14. An example: ION's passing a car in NQE, GC05

In GC05, the race route was described by a road definition file which consists of 2935 waypoints in total. Each waypoint was specified by GPS coordinates, the width and the maximum allowed speed of section. The total route distance was about 132 mile. Running with complete autonomous operations, ION covered about 30miles in the Nevada Desert and was terminated at the 576th waypoint from the start. The maximum speed ION reached in some sections was about 25 mph. During the race, ION experienced twice LIDAR failures and both were recovered in two minutes. Without the recovery design, ION could not have reached that far. Also, ION got into the *"robotic operation"* state twice. In the first set of robotic operations, ION tried five times back-and-forth operations before it overcame an obstacle. In the second set of robotic operations, ION was terminated. However, neither DARPA report nor ION's race log indicated that ION had had any collisions or gone off the road in the GC05 race. Also, ION was still totally driveable and stayed in the middle of the road when it was terminated. We guess it was because of the slowness in ION's gear shifting[3]. Since

[3] We were not provided with official reasons of the termination.

it took up to one minute for ION to shift its gear position in some situations, which might be intolerably slow, the second set of robotic operations might present the illusion that ION came to a halt and thus caused the termination. With the watchdog design in the FSM described in Section 6.3, given more time, we expect ION could have reached further in GC05 and even had the potential to finish the whole course.

6.6 Conclusion

ION, the working off-road AGV from team Desert Buckeyes, is a successful integration of a series of technologies, such as sensor fusion, navigation, vehicle control, signal processing, drive-by-wire technology, reliable software and mapping, etc. According to ION's performance in GC05, we realize that the robustness design is of great importance since some modules may not work properly in some situations while being off-road. We have invested extensive effort to improve the robustness of ION, for example, introducing the "Robotic Operation" and "Alarm" states in the FSM design, realizing the recovery design of sensor failures, and implementing the watchdog design for the FSM. Without these, ION could not have reached that far. Nevertheless, ION didn't finish the race because of some unexpected events. The second lesson we learn is that practicing in desert is also important for the race. We have had a lot of off-road practice yet we tested ION in desert for less than one week. Although this one-week testing improves the ION's desert performance greatly, it is still not enough. Practicing in the similar environment to the race can reduce the uncertainties and unexpected events.

This paper briefly introduces the system structure of ION and focuses on ION's navigation module and the obstacle avoidance algorithm utilizing fuzzy logic. The steering fuzzy controller, derived from a basic steering strategy, utilizes several fuzzy rules to deal with complicated situations so that the AGV can reach the goal point and avoid obstacles. The steering controller deals with the goal approaching and obstacle avoidance at the same time, which helps the AGV approach the goal smoothly. The speed controller utilizes fuzzy rules to prevent the AGV from colliding with obstacles and enhance the vehicle path following performance. ION's performance in GC05, both in NQE and the race, justifies the design of the real-time navigation module described in this paper.

Acknowledgement

A large group of researchers and students participated in ION's development. We acknowledge Dr. Keith Redmill, John Martin, Dr. Charles Toth, Matthew Knollman, George McWilliams and Rong Xu for their insightful discussions, valuable suggestions and contributions. We thank the Ohio State University for all their support and assistance. We also acknowledge the contribution of our partners, the team from University of Karlsruhe, which developed the image processing system.

References

Adams, M., & Probert, P. (1990, Jul.). Towards a Real-Tme Navigation Strategy for a Mobile Robot. Paper presented at the IEEE International Workshop on Intelligent Robots and Systems (IROS '90), Tsuchiura, Japan.

Borenstein, J., & Koren, Y. (1989). Real-time Obstacle Avoidance for Fast Mobile Robots. IEEE Transactions on Systems, Man and Cybernetics, 19(5), 1179-1187.

Chen, Q., Özgüner, Ü., & Redmill, K. (2004). The Ohio State University Team at the Grand Challenge 2004: Developing A Completely Autonomous Vehicle. IEEE Intelligent Systems, 19(5), 8-11.

Chen, Q., & Özgüner, Ü. (2005, Jun.). Real-time navigation for autonomous vehicles: a fuzzy obstacle avoidance and goal approach algorithm. Paper presented at the American Control Conference (ACC '05), Portland, OR.

Hatipoglu, C., Özgüner, Ü., & Redmill, K. (2003). Automated lane change controller design. IEEE Transactions on Intelligent Transportation Systems, 4(1), 13-22.

Hodge, N., & Trabia, M. (1999, May). Steering Fuzzy Logic Controller for an Autonomous Vehicle. Paper presented at the IEEE International Conference on Robotics and Automation, Detroit, MI.

Hsiung, C. (1998). First Course in Direrential Geometry. Springer, New York (USA), 1998.

Hummel, B., Kammel, S., Dang, T., Duchow, C., & Stiller, C. (2006, Jun.). Vision-based Path Planning in Unstructured Environments. Paper presented at the IEEE Intelligent Vehicle Symposium (IV '06), Tokyo, Japan.

Hwang, Y., & Ahuja, N. (1988, Apr.). Path planning using a potential field representation. Paper presented at the IEEE International Conference on Robotics and Automation, Philadelphia, PA.

Team Desert Buckeyes. (2005). Team Desert Buckeyes Technical Paper. http://www.darpa.mil/grandchallenge/TechPapers/DesertBuckeyes.pdf.

Khatib, O. (1985, Mar.). Real-Time Obstacle Avoidance for Manipulators and Mobile Robots. Paper presented at the IEEE International Conference on Robotics and Automation, St. Louis, MO.

Kambhampati, S., & Davis, L. (1986). Multiresolution path planning for mobile robots. IEEE Journal on Robotics and Automation, 2(3), 135-145.

Latombe, J.-C. (1991). Robot motion planning. Boston MA: Kluwer Academic Publishers.

Lee, T., Lai, L., & Wu, C. (2005, May). A fuzzy algorithm for navigation of mobile robots in unknown environments. Paper presented at the IEEE International Symposium on Circuits and Systems 2005 (ISCAS '05), Kobe, Japan.

Özgüner, Ü., Hatipoglu, C., & Redmill, K. (1997, Nov.). Autonomy In A Restricted World. Paper presented at the IEEE Conference on Intelligent Transportation System (ITSC '97), Boston, MA.

Özgüner, Ü., Redmill, K., & Broggi, A. (2004, Jun.). Team TerraMax and the DARPA Grand Challenge: A General Overview. Paper presented at the IEEE Intelligent Vehicles Symposium (IVS '04), Parma, Italy.

Redmill, K., Kitajima, K., & Özgüner, Ü. (2001, Aug.). DGPS/INS integrated positioning for control of automated vehicle. Paper presented at the IEEE Conference on Intelligent Transportation System (ITSC '01), Oakland, CA.

Redmill, K., & Özgüner, Ü. (1998, Feb.). The Ohio State University Automated Highway System Demonstration Vehicle. Paper presented at the 1998SAE International Congress and Exposition (SAE Paper 980855), Detroit, MI.

Redmill, K., Martin, J., & Özgüner, Ü. (2006, Jun.). Sensing and Sensor Fusion for the 2005 Desert Buckeyes DARPA Grand Challenge Offroad Autonomous Vehicle. Paper presented at the IEEE Intelligent Vehicle Symposium (IV '06), Tokyo, Japan.

Saffiotti, A. (1997). The uses of fuzzy logic in autonomous robot navigation. Journal of Soft Computing, 1(4), 180-197.

Seraji, H. (2000). Fuzzy Traversability Index: A new concept for terrain-based navigation. Journal of Robotic Systems, 17(2), 75-91.

Stentz, A. (1994, May). Optimal and Efficient Path Planning for Partially-Known Environments. Paper presented at the IEEE International Conference on Robotics and Automation (ICRA'94).

Toth, C., Paska, E., Chen, Q., Zhu, Y., Redmill, K., & Özgüner, Ü. (2006) Mapping Support for the OSU DARPA Grand Challenge Vehicle. Paper submitted to the IEEE Conference on Intelligent Transportation System (ITSC '06), Toronto, Canada.

Warren, C. (1989, May). Global Path Planning Using Artificial Potential Fields. Paper presented at the IEEE International Conference on Robotics and Automation, Scottsdale, AZ.

Yu, H., Chen, Q., & Özgüner, Ü. (2004, Jul.). Control System Architecture for TerraMax - The off-road intelligent navigator. Paper presented at the 5th IFAC/EURON Symposium on Intelligent Autonomous Vehicles, Lisboa, Portugal.