

USING FUZZY LOGIC IN AUTOMATED VEHICLE CONTROL

Navdeep Kaur¹, Ms. Sana Tak²

¹Enrollment No.:- C2IMT(CS)100009, ²Assistant Professor in CSE Deptt.
Kalinga University, Raipur (C.G.)

ABSTRACT: Automated versions of a heavily-produced vehicle uses fuzzy logic techniques to both address common challenges and incorporate human procedural knowledge into the vehicle managing algorithms. In-vehicle computing has been largely relegated to auxiliary tasks such as regulating cabin temperature, opening doors, and monitoring fuel, oil, and battery-charge levels. However, computers are increasingly assuming driving-related tasks in some commercial models. Among those tasks are: maintaining a reference velocity or keeping a safe distance from other vehicles; improving night vision with infrared cameras; and building maps and providing alternative routes.

I. INTRODUCTION

Still, many traffic situations remain complex and difficult to manage, particularly in urban settings. The driving task belongs to a class of problems that depend on underlying systems for logical reasoning and dealing with uncertainty. So, to move vehicle computers beyond monitoring and into tasks related to environment perception or driving, we must integrate aspects of human intelligence and behaviors so that vehicles manage driving actuators in a way similar to humans. This is the motivation behind the AUTOPIA program, a set of national research projects in Spain. AUTOPIA has two primary objectives: First, we want to implement automatic driving using real, mass-produced vehicles tested on real roads. Although this objective might be called “utopian” at the moment, it’s a great starting point for exploring the future. Our second aim is to develop our automated system using modular components that can be immediately applied in the automotive industry. AUTOPIA builds in the Instituto de Automatica Industrail’s extensive experience developing autonomous robots and fuzzy control systems and the Universidad de Alcala de Henares’s knowledge of artificial vision.

Automated-Vehicle Equipment:-

Fig 1 shows two mass-produced electric Citroen Berlingo vans, which we’ve automated using an embedded fuzzy-logic-based control system to control their speed and steering. The system’s main sensor inputs are a CCD(Charged Couple Device) color camera and a high-precision global positioning system. Through these, the system controls the vehicle-driving actuators-that is, the steering, throttle, and brake pedals. Both vehicles include an onboard PC-based computer, a centimetric, real-time kinematic differential GPS(RTK DGPS); Wireless LAN support; two servomotors; and an analog/digital I/O card. We added a vision system in another computer connected to the control computer. Fig 2 shows the control system that we developed to handle all these devices.



Figure 1. The AUTOPIA testbed vehicles. An embedded fuzzy-logic-based control system controls both speed and steering in each Citroën Berlingo.

The computer drives the vans using two fuzzy-logic-based controllers: the steering(lateral) control and the speed(longitudinal) control. To automate the steering, we installed a DC servomotor in the steering wheel column. The Berlingo has an electronic throttle control, so we shortened the electronic circuit to actuate the throttle using an analog output card. The brale pedal is fully mechanical; we automated it using a pulley and a DC servomotor. We equipped the transmission with an electronic gearbox with forward and reverse selection. We automated this using a digital I/O card that sends the correct gear to the internal vehicle computer. We designed our driving area to emulate an urban environment because automatic urban driving is one of ITS’s less researched topics.

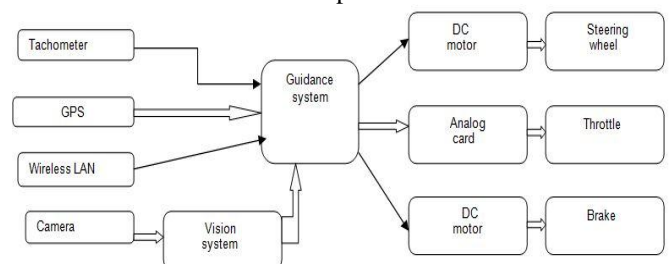


Figure 2. The AUTOPIA system control structure. The sensorial equipment supplies the necessary data to the fuzzy-logic-based guidance system, which decides the optimal control signals to manage the vehicle actuators (steering wheel, throttle and brake).

II. GUIDANCE SYSTEM

We modeled the guidance system using fuzzy variables and rules. In addition to the steering wheel and vehicle velocity functionalities, we also consider variables that the system can use in adaptive cruise control(ACC) and overtaking capabilities. Among these variables are the distance to the

next bend and the distance to the lead vehicle(that is, any vehicle driving directly in front of the automated vehicle).

Car driving is a special control problem because mathematical models are highly complex and can't be accurately linearized. We use fuzzy logic because it's a well-tested method for dealing with this kind of system, provides good results, and can incorporate human procedural knowledge into control algorithms.

Steering Control:-

The steering control system's objective is to track a trajectory. To model lateral and angular tracking deviations perceived by a human driver, we use two fuzzy variables : Lateral_Error and Angular_Error. These variables represent the difference between the vehicle's current and correct position and its orientation to a reference trajectory.

Both variables can take left or right linguistic values. Angular_Error represents the angle between the orientation and vehicle velocity vectors. If this angle is counterclockwise, the Angular_Error value is left. If the angle is clockwise, the Angular_Error value is right. Lateral_Error represents the distance from the vehicle to the reference trajectory. If the vehicle is positioned on the trajectory's left, the Lateral_Error value is left; it's right if the vehicle is on the right.

We compute the variables' instantaneous value using the DGPS data and a digital environment map. The fuzzy output variable is Steering_Wheel and indicates which direction the system must turn the steering wheel to correct the input errors. Again, the variable also has left and right linguistic values. The value is left if the steering wheel must turn counterclockwise, and right if it must turn clockwise. We define the fuzzy sets that define the left and right values in an interval of -540 degrees and 540 degrees.

As with human behavior, our guidance system works differently for tracking lanes or turning on sharp bends. When travelling along a straight road, people drive at relatively high speeds while gently turning the steering wheel. In contrast, on sharp bends, they rapidly reduce speed and quickly turn the steering wheel. We emulate such behavior by changing the membership function parameters of the Lateral_Deviation, Angular_Deviation, and Steering_Wheel linguistic variables. To represent the human procedural knowledge in the driving task, we need only two fuzzy rules. These rules tell the fuzzy inference motor how to relate the fuzzy input and output variables:

If Angular_Error left OR Lateral_Error left THEN Steering_Wheel right
 IF Angular_Error right OR Lateral_Error right THEN Steering_Wheel left

Although these rules are simple, they generate results that are close to human driving. The rules are the same for all situations, but the definition of the fuzzy variables' linguistic values change. Fig 3 shows this feature in the membership function definition for Lateral_Error and Angular_Error. Figs 3a and 3b show the degree of truth for the input error values in straight-path tracking situations. This definition lets the

system act quickly when trajectory deviations occur-again in keeping with human behavior.

To prevent accidents, we must limit the maximum turning angle for straight-lane driving. This limitation is also similar to human behavior; we achieve it by defining the output variable membership function as a singleton, confining this turning to 2.5 percent of the total. This makes the driving system less reactive when tracking a straight trajectory and assures that they'll adapt to the route smoothly. We can also represent the output using a singleton without turning limitations. We fine-tuned the membership functions experimentally, comparing their behavior with human operations and correcting it accordingly until the system performed acceptably. So, the driving system selects a fuzzy membership function set depending on the situation, which leads to different reactions for each route segment.

Speed Control:-

To control speed, we use two fuzzy input variables: Speed_Error and Acceleration. To control the accelerator and the brake, we use two fuzzy output variables: Throttle and Brake. The Speed_Error crisp value is the difference between the vehicle's real speed and the user-defined

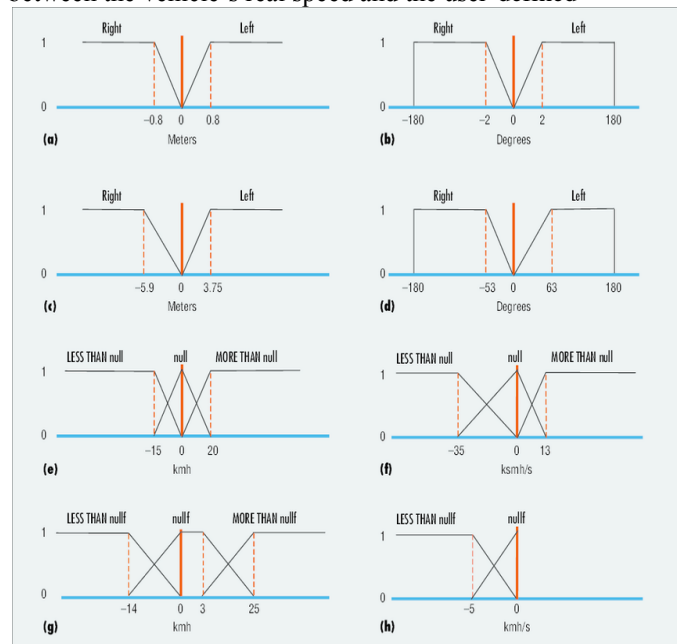


Figure 3. The membership function definition for fuzzy variables: (a) Lateral_Error straight, (b) Angular_Error straight, (c) Lateral_Error curves, (d) Angular_Error curves, (e) Speed_Error throttle, (f) Acceleration throttle, (g) Speed_Error brake, and (h) Acceleration brake.

target speed, and the Acceleration crisp value is the speed's variation during a time interval. The throttle pressure range is 2-4 volts, and the brake pedal range is 0-240 degrees of the actuation motor.

The fuzzy rules containing procedural knowledge for throttle control are:-

IF Speed_Error MORE THAN null THEN Throttle up
 IF Speed_Error LESS THAN null THEN Throttle down

IF Acceleration MORE THAN null THEN Throttle up
IF Acceleration LESS THAN null THEN Throttle down

The rules for brake control are:-

IF Speed_Error MORE THAN nullf THEN Brake down
IF Speed_Error LESS THAN nullf THEN Brake up
IF Acceleration LESS THAN nullf THEN Brake up

Where brake/throttle down means depress the brake and throttle, and brake/throttle up means release the brake and throttle. The associated membership functions of the fuzzy linguistic labels null and nullf define the degree of nearness to 0 of Acceleration and Speed_Error, respectively.

Figures 3e through 3h show the membership functions of null(for the throttle controller) and nullf (for the brake controller) for Speed_Error and Acceleration, respectively. An asymmetry exists in the two variable definitions for two reasons:

- To account for the difference in how accelerating and braking vehicles behave, and
- To coordinate both pedals' actuation to emulate human driving.

Throttle and brake controllers are independent, but they must work cooperatively. Activating the two pedals produces similar outcomes and can

- Increase the target speed (stepping on the throttle or stepping off the brake on down-hill roads),
- Maintain speed (stepping on or off either pedal when necessary), and
- Reduce the vehicle's speed (downshifting the throttle or stepping on the brake).

ACC+Stop&Go

With ACC, the system can change the vehicle's speed to keep a safe distance from the lead vehicle. As an extreme example, the lead vehicle might come to a complete stop owing to a traffic jam. In this case, the ACC must stop the vehicle using a stop- and-go maneuver; when the road is clear, the ACC reaccelerates the vehicle until it reaches the target speed. Combining ACC with stop-and-go maneuvers increases driving comfort, regulates traffic speed, and breaks up bottlenecks more quickly. Many rear-end collisions happen in stop-and-go situations because of driver distractions.

ACC systems have been on the market since 1995, when Mitsubishi offered the Preview Distance Control system in its Diamante model. Several sensors can provide the vehicle with ACC capability; radar, laser vision, or a combination thereof. Almost all car manufacturers now offer ACC systems for their vehicles, but they all have two clear drawbacks. First, the ACC systems don't work at speeds lower than 40kmh, so they can't offer stop-and-go maneuvers. Second, the systems manage only the throttle automatically; consequently, the speed adaptation range is limited.

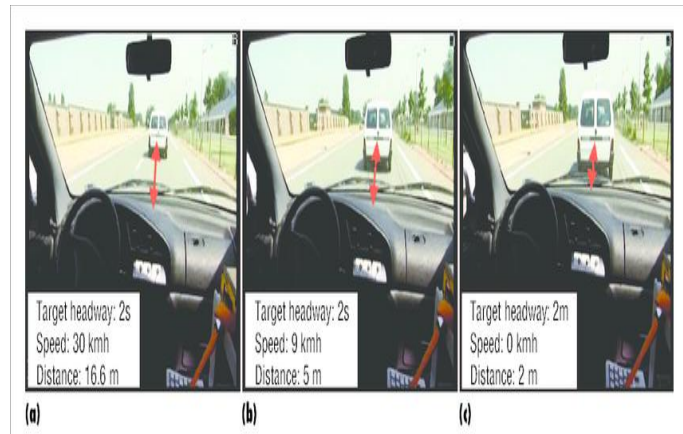


Figure 4. The ACC+Stop&Go controller's performance in an automated vehicle. Keeping a safe distance (a) at 30 kmh, (b) during speed reduction, and (c) in stop-and-go situations.

Our system overcomes these limitations by automating the throttle and brake, which lets the system act across the vehicle's entire speed range. In our case, we selected GPS as the safety-distance sensor. We installed GPS in both the vehicles, and they communicate their position to one another via WLAN.

Keeping a user-defined safety distance from the next vehicle is a speed-dependent function: the higher the speed, the larger the required intervehicle gap. This is the time-headway concept- a time-dependent safety distance maintained between two vehicles. If we set a safety time gap of two seconds, for example, the space gap is 22.2 meters for a vehicle moving at 40 kmh but approximately 55.5 meters for 100 kmh. The time gap setting depends on the vehicle's braking power, the weather, the maximum speed, and so on.

Figure 4 shows our ACC+Stop&Go controller's performance in one of our automated vehicles. At the experiment's beginning, the trailing vehicle starts moving, speeds up, and eventually stops because the lead vehicle is blocking the way. The lead vehicle then starts moving, gains speed, and brakes again, emulating a congested traffic situation. A few seconds later, the trailing vehicle starts up again, eventually stopping behind the lead vehicle.

Overtaking:-

The system can also manage obstacles or other vehicles in the vehicle's path by calculating when the vehicle should change lanes to overtake the obstacles. First,

- The vehicle must be in the straight-lane driving mode,
- The left lane must be free, and
- There must be room for the overtaking.

Given this, overtaking occurs as follows:-

- Initially, the vehicle is in straight-lane mode.
- The driving mode changes to lane-change mode, and the vehicle moves into the left lane.
- The driving mode changes to straight-lane mode until the vehicle has passed the obstacle or vehicle.

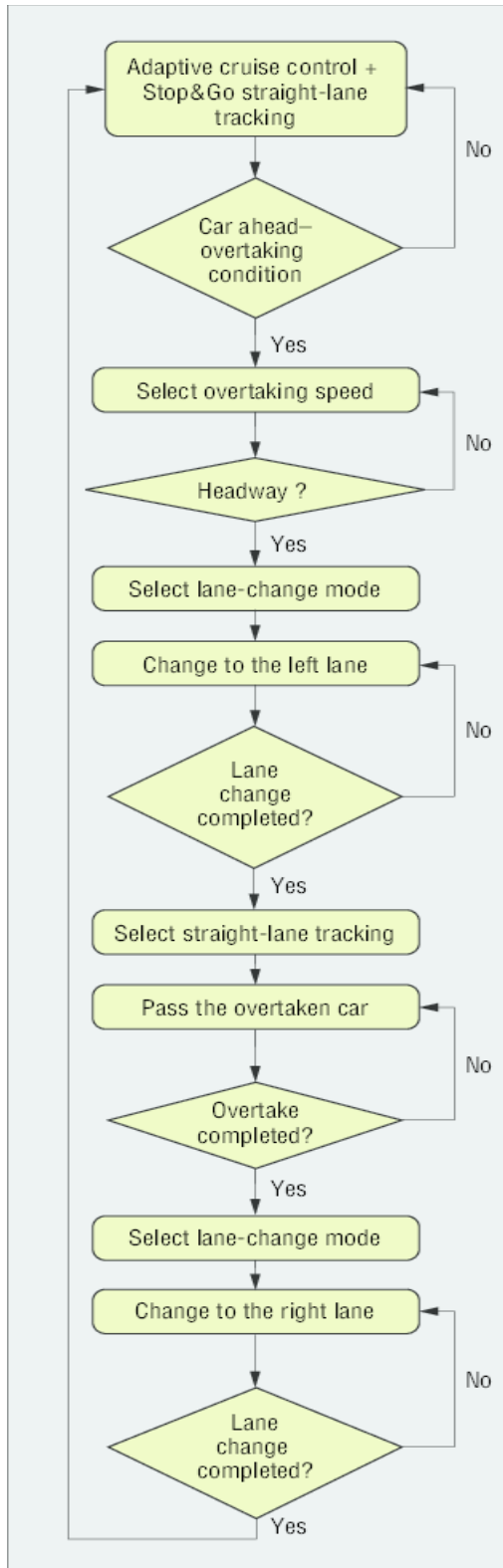


Figure 5. A flow chart of the overtaking algorithm.

The driving mode again changes to lane-change mode, and the vehicle returns to the right lane.

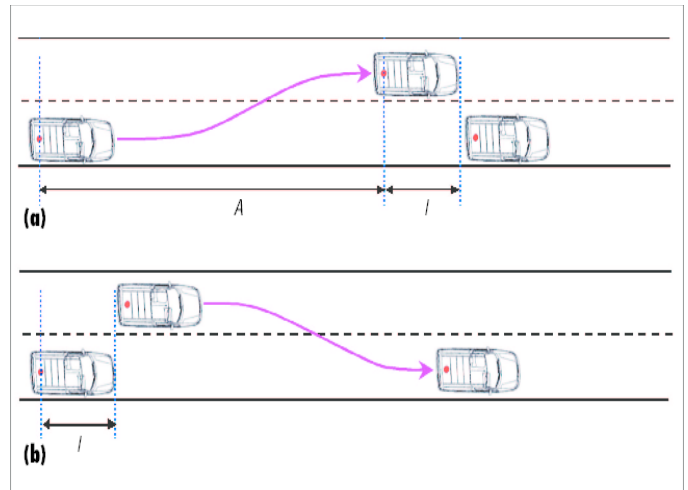


Figure 6. The overtaking maneuver (a) starts with a change from the right to the left lane and (b) ends with a change from the left to the right lane. A is the distance at which the vehicle changes lanes, and l is the vehicle's length.

When the vehicle is centered in the lane, the driving mode changes back to straight-lane mode, and driving continues as usual.

Figure 5 shows a detailed flowchart of this algorithm. We calculate the time for starting the transition from the first to the second step as a function of the vehicles' relative speed and the overtaking vehicles' length. Figure 6 illustrates the overtaking maneuver. The overtaking vehicle must change lanes at point $A+l$, where A is the distance at which the vehicle changes lanes and l is the vehicle's length. The dot on the back of each vehicle represents a GPS antenna, located over the rear axle. Vehicles use the GPS receptor and the WLAN link to continuously track their own position and that of other vehicles. The lane change proceeds only if the front of the overtaking vehicle is completely in the left lane upon reaching the rear of the overtaken vehicle in the right lane. A is speed dependent- $A=F(v)$, where v is the relative speed between the overtaking and overtaken vehicles because the higher the velocity, the larger the lane-change distance. A is a function of the relative speed between both vehicles because overtaking depends on the two mobile objects' speed. In this case, l is 4 meters, a Citroen Berlingo's length. The system transitions from step 2 to step 3 when the overtaking vehicle's angular and lateral errors are both low. Specifically, $Angular_Error$ must be less than 2 degrees and $Lateral_Error$ less than 0.8 meter. The system transitions to step 4 when the overtaking vehicle's rear end passes the overtaken vehicle's front end and the separation is l . Finally, the transition to step 5 is the same as from step 2 to 3.

Vision-based vehicle detection

To achieve reliable navigation, all autonomous vehicles must master the basic skill of obstacle detection. This vision-based task is complex. Consider, for example, common situations in urban environments, such as missing lane markers, vehicles parked on both sides of the street, or crosswalks. All such situations make it difficult for a system to reliably detect other vehicles, creating hazards for the host vehicle. To address this, we use a monocular color-vision system to give our GPS-based navigator visual reactive capacity.

Search and vehicle detection

We sharply reduce execution time by limiting obstacle detection to a predefined area in which obstacles are more likely to appear. This rectangular area-or region of interest(ROI)-covers the image's central section.

To robustly detect and track vehicles along the road, we need two consecutive processing stages. First, the system locates vehicles on the basis of their color and shape properties, using vertical edge and color symmetry characteristics. It combines this analysis with temporal constraints for consistency, assuming that vehicles generally have artificial rectangular and symmetrical shapes that make their vertical edges easily distinguishable. Second, the system tracks the detected vehicle using a real-time estimator.

Vertical-edge and symmetry discriminating analysis

After identifying candidate edges representing the target vehicle's limits, the system computes a symmetry map of the ROI to enhance the objects with strong color symmetry characteristics. It computes these characteristics using pixel intensity to measure the match between two halves of an image region around vertical axis. It then considers the vertical edges of paired ROIs with high symmetry measures(rejecting uniform areas). It does this only for pairs representing possible vehicle contours, disregarding any combinations that lead to unrealistic vehicle shapes.

Temporal consistency

In the real world, using only spatial features to detect obstacles leads to sporadic, incorrect detection due to noise. We therefore use a temporal validation filter to remove inconsistent objects from the scene. That is, the system must detect any spatially interesting object in several consecutive image iterations to consider that object a real vehicle; it discards all other objects.

We use the value $t=0.5s$ to ensure that a vehicle appears in a consistent time sequence. A major challenge of temporal-spatial validation is for the system to identify the same vehicle's appearance in two consecutive frames. To this end, our system uses the object's(x,y) position in correlative frames. That is, it can use the position differences to describe the vehicle's evolution in the image plane. At time instant t_0 , the system annotates each target object's (x,y) position in a dynamic list, and starts a time count to track all candidate vehicles' temporal consistency. At time t_0+1 , it repeats the process using the same spatial-validation criterion. We increase the time count only for those objects whose distance from some previous candidate vehicle is less than d_v . Otherwise, we reset the time count. A candidate object is validated as a real vehicle when its time count reaches $t=0.5s$. Given that the vision algorithm's complete execution time is 100 ms, an empirical value $d_v=1m$ has proven successful in effectively detecting real vehicles in the scene. Figure 7 shows examples of the original and filtered images along with the ROI symmetry map and the detected vehicle's final position.

Vehicle tracking

We track the detected vehicle's position using position measurement and estimation. We use the detected vehicle's

ROI image as a template to detect position updates in the next image using a best-fit correlation. We then use the vehicle's (x,y) location in data association for position validation. Basically, we want to determine whether any object in the current frame matches the vehicle being tracked. To do this, we specify a limited search area around the vehicle position, leading to fast, efficient detection. We also establish a minimum correlation value and template size to end the tracking process if the system obtains poor correlations or if the vehicle moves too far away or leaves the scene.

Next, we filter the vehicle position measurements using a recursive least-squares estimator with exponential decay. To avoid partial conclusions, the system keeps the previously estimated vehicle position for five consecutive iterations—without calculating any validated position—before considering the vehicle track as lost. Given a loss, the system stops vehicle tracking and restarts the vehicle detection stage. Figure 8 illustrates our algorithm, showing how the system tracked the lead vehicle in real traffic situations.

Adaptive Navigation

After detecting the lead vehicle's position, we must ensure safe navigation in ACC mode if the lead vehicle suddenly brakes within the safety gap limits. This event could easily lead to a crash unless the host vehicle rapidly detects the braking situation and brakes hard. To ensure this, the system must detect the lead vehicle's brake light activation, which clearly indicates braking.

A vehicle's brake light position varies depending on its model and manufacturer. So, the system must carry out a detailed search to accurately locate these lights inside the vehicle's ROI. We do have some priori information to ease the search: brake indicators are typically two red lights symmetrically located near the vehicle's rear left and right sides. Once the system locates these lights, it must detect sudden brake light activation; it does this by continuously monitoring the lights' luminance. In case of sudden activation, the system raises an alarm to the vehicle navigator to provoke emergency braking.

Figure 9 shows an example of sudden braking decision. Brake lights are a redundant safety feature: if they're activated, a braking procedure has already started. Fortunately, our system continuously computes the distance to the lead vehicle. If this distance is too short, it automatically stops the vehicle.

III. CONCLUSION

We carried out all our experiments within a private circuit and the results shows that the fuzzy controllers perfectly mimic human driving behavior in driving and route tracking, as well as in more complex, multiple-vehicle maneuvers, such as ACC or overtaking. In the near future, we're planning to run new experiments involving three automatic driving cars in more complex situations such as intersections or roundabouts. Fuzzy control's flexibility let us integrate a host of sensorial information to achieve our results. Also, using vision for vehicle and obstacle detection lets the host vehicle react to real traffic conditions, and has proven a crucial complement to the GPS-based navigation system. To

improve and further this work, we're collaborating with other European institutions specializing in autonomous vehicle development under the UE Contract Cyber-Cars-2. Through this collaboration, we plan to perform a cooperative driving involving more than six vehicles, adding new sensors for pedestrian detection, traffic-sign detection, and infrastructure monitoring. We'll also integrate new wireless communication systems that include vehicle-to-vehicle, vehicle-to-infrastructure, and in-vehicle information transmission. Finally, we're planning to use Galileo , next-generation GPS systems that address some existing GPS positioning problems and improve location accuracy.

REFERENCES

- [1] M.A. Sotelo et al., "Vehicle Fuzzy Driving Based on DGPS and Vision," Proc. 9th Int'l Fuzzy Systems Assoc., Springer, 2001, pp.1472-1477.
- [2] R. Garcia et al., "Frontal and Lateral Control for Unmanned Vehicles in Urban Tracks," IEEE Intelligent Vehicle Symp. (IV2002), vol.2, IEEE Press, 2002, pp. 583–588.
- [3] A. Broggi et al., "The Argo Autonomous Vehicle's Vision and Control Systems," Int'l J. Intelligent Control and Systems, vol. 3, no.4, 2000, pp. 409–441.
- [4] M.A. Sotelo, Global Navigation System Applied to the Guidance of an Terrestrial Autonomous Vehicle in Partially Known Outdoor