Proceedings of the 2006 IEEE/RSJ
International Conference on Intelligent Robots and Systems
October 9 - 15, 2006, Beijing, China

Cooperative Driving based on Inter-vehicle Communications: Experimental Platform and Algorithm

Weihua Sheng
School of Electrical & Computer Eng.
Oklahoma State University
Stillwater, OK 74078

Qingyan Yang
System Division
Iteris Inc.
Madison Heights, MI 48071

Yi Guo
Dept. of Electrical & Computer Eng.
Stevens Institute of Technology
Hoboken, NJ 07030

***Abstract*— This paper describes our efforts in building an experimental platform to conduct research on cooperative driving in Intelligent Transportation Systems (ITS). A miniature vehicle is developed based on the COTS-BOTS robot and an overhead vision system is used to localize the vehicles. The single lane tracking control algorithm is described first. To control multiple vehicles to drive through an intersection without collision, a distributed cooperative driving algorithm is developed, which is based on velocity planning using search algorithms in the corresponding Coordination Diagram. Experimental results verify our proposed approach and we believe this platform can serve as a cost-effective testbed to study vehicle-to-vehicle communication based cooperative driving in Intelligent Transportation Systems.**

***Index Terms*— Intelligent Transportation System, Cooperative Driving, Vehicle-to-Vehicle Communication**

I. Introduction

Automated vehicles are an important constituent of future Intelligent Transportation Systems. Vehicle-to-vehicle communication will certainly be indispensable to maintain safe and efficient automated driving. Currently, major research thrusts on automated highway systems and automated driving include the PATH program in California [1]; the DARPA-supported Unmanned Ground Vehicle (UGV) project in Carnegie Mellon University [2]; the PROMETHEUS program in Europe [3] and Super Smart Vehicle System (SSVS) project in Japan [4], etc. However, it is still difficult for many researchers to access multiple full-size, automated vehicles and the various equipment necessary to conduct experiments. Thus it is hard to fully investigate and quickly test various control algorithms proposed. Pure simulation in software may help in the fast evaluation of control algorithms, but may lack accuracy. Therefore it is desirable to develop a cost effective experimental platform to study inter-vehicle communication based cooperative driving, with a goal of improving the safety and efficiency of automated driving.

Currently, vehicle-to-vehicle communication is mostly used to extend the horizon of drivers and on-board devices such as radar or sensors. In the field of ITS, collaborative anti-collision of multiple vehicles have not been thoroughly investigated. The anti-collision problem at an intersection has received even less attention compared to the rear-end collision [5], although the former account for almost 30% of all crashes. Yang et al. [6] proposed a vehicle-to-vehicle communication protocol for cooperative collision warning. Their protocol consists of congestion control policies, service differentiation mechanism and method of emergency warning dissemination. The major goal of their research is to achieve low-latency in message delivery. However, only simulation is conducted and no physical implementation is carried out. Gorman [7] investigated the vehicle coordination problem at a 4-way stop crossing (without traffic lights). He applied the group communication to perform coordination functions. However, the developed coordination protocols were not implemented and verified in physical vehicles.

On the other hand, research in mobile robots, especially in multiple mobile robots has resulted in meaningful results [8] that can be extended to multi-vehicle collision avoidance. Motion planning in a dynamic environment is considered NP-hard [9]. Therefore the motion planning problem is usually decoupled into path planning and velocity planning problem. The work in this area includes [10], [11], [12] etc. Guo and Parker [13] proposed a distributed motion planning algorithm to coordinate multiple mobile robots that have independent goals so that there is no collision and certain performance optimality can be achieved. They divided the overall problem into path planning and velocity planning. D^* search algorithm [14] is applied in both modules. However, their algorithm assumes that the vehicle communication range is unlimited and all vehicles can communicate with each other at any time, which may not reflect the reality. Also the search algorithm results in velocity profiles that require the vehicles to stop within the intersection.

In this work, we develop an experimental platform, an overall software architecture and a new distributed motion control algorithm for multiple vehicle collision avoidance. This paper is organized as follows. In Section II, the system setup is introduced, which includes the

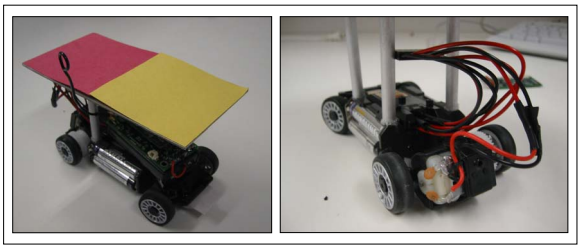

Fig. 1. The miniature vehicle. The left picture shows the overall vehicle and the right picture shows the RC car base with the Mica2 mote removed.

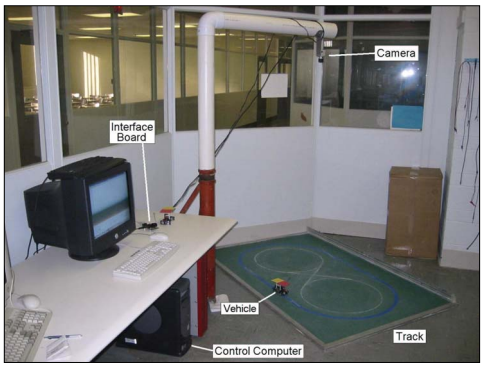

Fig. 2. The experimental platform.

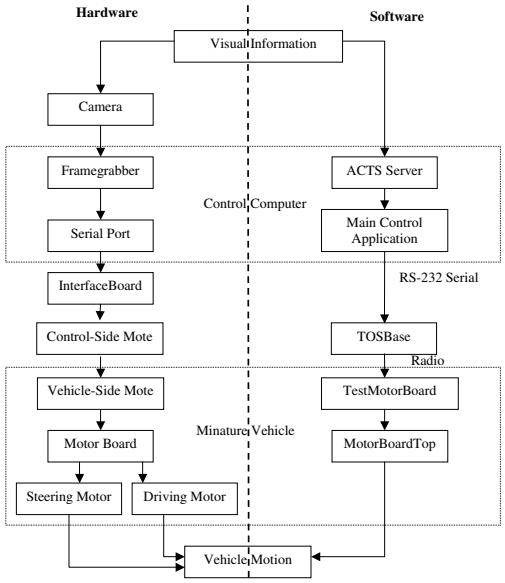

Fig. 3. The system hardware and software block diagram.

miniature vehicle and the overall platform. Section III describes the approach to control a single vehicle to track a lane. Multiple vehicle collision avoidance is presented in Section IV. Section V gives the test results and Section VI concludes this paper.

II. System Setup

A. Miniature vehicle

The miniature vehicle employed in this project is modified from the COTS-BOTS (Commercial Off-The-Shelf Robots) developed by University of California Berkeley [15]. The COTS-BOTS consists of the base of a commercially available Mini-Z Racer radio-control car and a Crossbow Mica2 mote which is a programmable wireless sensor node [16] made by Crossbow Technology Inc. The Mica2 can operate on a number of different frequency bands and in this work the 916MHz band is used. It is capable of radio communication at rates up to 38.4 kbps. The COTS-BOTS robot has a motor board that controls the steering motor and the driving motor. The motor board communicates with the Mica2 through a 51-pin expansion connector.

The control software of the COTS-BOTS runs on TinyOS [17], an open source operating system designed for use in embedded sensor networks. It is extremely lightweight and minimizes both resource usage and code size to fit the restraints of embedded architectures. The miniature vehicle is shown in Figure 1, which has four poles to support the two-color top for visual identification purpose. It has a dimension of 7 cm $\times$ 14cm.

B. Experimental platform

Figure 2 shows the overall experimental platform. It consists of a) a desktop PC with a Pentium IV processor and 512M bytes of memory, which runs on Fedora Core 2 and contains a Sensoray Model 611 PCI framegrabber; b) a Hitachi KPD-20B camera; c) multiple miniature vehicles running in a small arena, which has tracks set up on a chalkboard; d) a Mica2 mote interfacing board and a control-side mote, together serving as the communication gateway between the PC and the miniature vehicles.

ActivMedia's ACTS [18] (ActivMedia Color Tracking System) software is employed to achieve the system's visual feedback. The software uses color recognition to identify "blobs" of a user-defined color in a visual frame. It uses a client/server model to accomplish the task of processing the information and distributing it to user applications. The ACTS server must be trained via the program's interface to track the desired colors. The sensitivity of the system to subtle lighting changes is highly dependent on the quality of the training data provided. The server can be programmed with up to 32 such ranges of color (channels), and can track up to ten blobs on each channel. When requested by a client, the server captures an image frame and processes it using the training data. It then makes available the resulting blob information packet, which represents the position and size of each color blob tracked.

The overall hardware and software diagram is shown in Figure 3, with the left side representing the hardware and the right side representing the corresponding software. The ACTS server receives visual data from the camera through driver software. Client programs, such as the main

control application, can connect to the server through a TCP socket connection. When the main control application requests a packet, the server processes an image frame based on the training configuration data provided by the user at server startup. This processed data then becomes available as a color blob information packet for the main control program to read. The program processes the blob information to determine the instantaneous position and heading of the vehicle. It uses this information, along with previously defined track initialization information, to quantify the vehicle's position error and determine the action that should be taken. This desired action is then packaged into a TinyOS packet and sent to the serial port. The control-side mote runs a piece of software called TOSBase, which is provided in the TinyOS package. The TOSBase software then transmits the packet at the preprogrammed frequency of 916MHz. The vehicle-side mote receives the radio packet. The Mica2 mote on the miniature vehicle checks the identification and type fields of the packet and sends appropriate commands to the motorboard. These functions are performed by a piece of software included with the COTS-BOTS package called TestMotorBoard. The motorboard's resident control program, also provided in the COTS-BOTS package, is called MotorBoardTop. The command packet it receives from the mote contains three parameters: speed, forward or reverse direction, and turn angle. The first two are used in a simple way to control the driving motor. The last is accomplished using a proportional-integral (PI) feedback loop.

III. Single Vehicle Lane Tracking

For the single vehicle lane tracking, it is assumed that the vehicle's speed remains constant throughout the test. Thus the only control parameter that the application must determine is the vehicle's turn angle. We develop an *inverse proportional zone-targeting algorithm* for the lane tracking. Essentially, the control program attempts to orient the vehicle to pass through a certain target zone, the size of which is inversely proportional to the vehicle's position error to the the desired path.

There are three different actions that represent the entire output set of this algorithm: drive straight, turn left, or turn right. The outcome depends on the point of intersection between the desired path and the vehicle's actual heading. Figure 4 shows an example of two possible vehicle positions relative to a straight path and the target zones associated with their offsets. If a vehicle's heading would intersect the desired path beyond the target zone, as with vehicle A, whose target zone is represented by the red line/left dark line, it will be turned in toward the desired path to compensate. The turn angle is proportional to the deviation distance D. If the vehicle's path intersects the desired path within the target zone, as with vehicle B, whose target zone is represented by the magenta line/right dark line, it will be ordered to drive straight ahead. If the vehicle's current path would intersect the desired path before the target zone, it will be turned away from the line to prevent overshoot. Figure 4 only shows two scenarios of the vehicle position and heading with respect to the desired path. Other scenarios can be dealt with similarly.

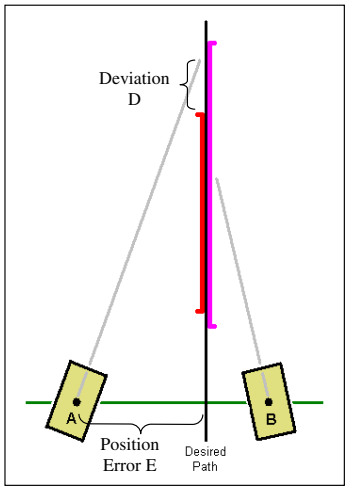

Fig. 4. Two scenarios in the zone-targeting algorithm.

To this point, the algorithm has been described only for straight path lines. However, the exact same rules apply for curves. In those cases, the "straight" outcome is simply replaced with the turn angle value of the curvature of the path, and the right and left turns are made with an offset relative to this angle.

IV. Multiple Vehicle Collision Avoidance

This section describes the distributed control algorithm for multiple vehicles to achieve collision avoidance capability. A typical intersection is shown in Figure 5, with all the possible vehicle paths numbered, from 1 to 12. Obviously, each vehicle has three choices to pass through the intersection. Only a subset of the paths have potential collisions, as marked by the collision point in Figure 5.

Figure 6 shows the software architecture for multiple vehicle collision avoidance. The motion control algorithm for each vehicle runs on the PC and is implemented in separate processes. The ACTS server implements the image processing and returns the blob location information requested by the ACTS client, which periodically sends the location of the vehicles to a communication server. Each vehicle control process runs independently and receives the periodical location update from the communication sever. Therefore, each vehicle control process knows the location and velocity of the vehicle it represents. The communication between the vehicles is enabled by the communication server, which, based on the knowledge of the global status of all the vehicles, simulates the communication connectivity among the vehicles.

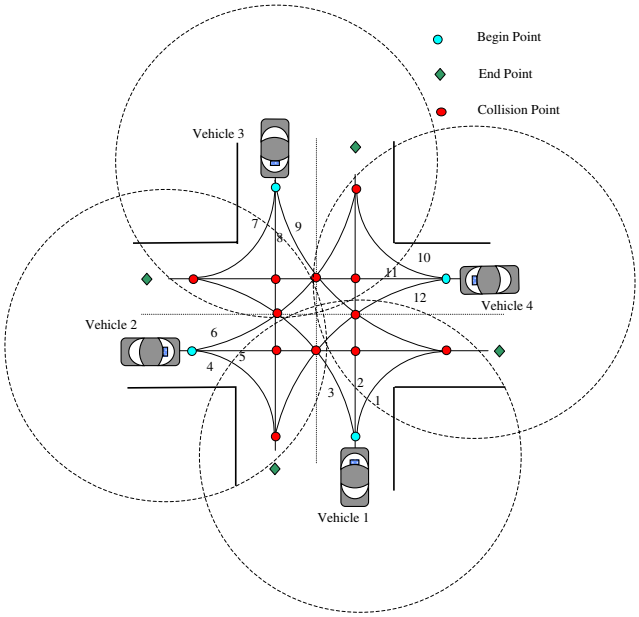

Fig. 5. A typical 4 way intersection. The dotted circles represent the communication range of vehicles.

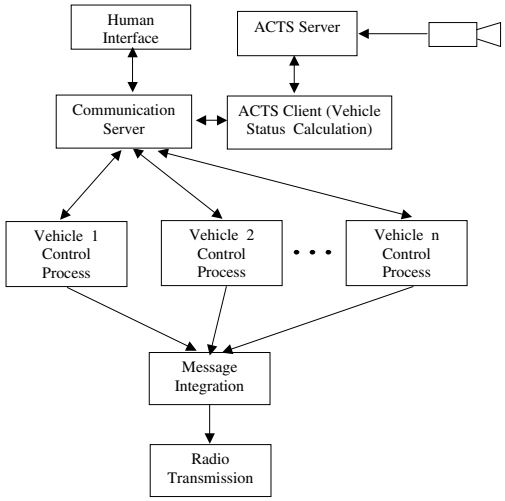

Fig. 6. The software architecture for multiple vehicle coordination.

The motion control algorithm is distributed, which implies that there is no central authority that coordinates the motion of all the vehicles. The algorithm is based on Guo and Parker's distributed motion planning algorithm [13]. However, we consider the limited communication range of vehicles, which reflects the reality. Upon approaching the intersection, if vehicle A_i is passing through the begin point as shown in Figure 5, it starts broadcasting its intended path P_i, as well as the current status $S_i = (L_i, V_i)$ to other vehicles, where L_i is the vehicle location and V_i is the vehicle velocity. Once it passes the end point, it will stop broadcasting and will no longer participate in collision avoidance. Upon receiving the broadcast (P_j, S_j) from another vehicle A_j, A_i checks the collision situation

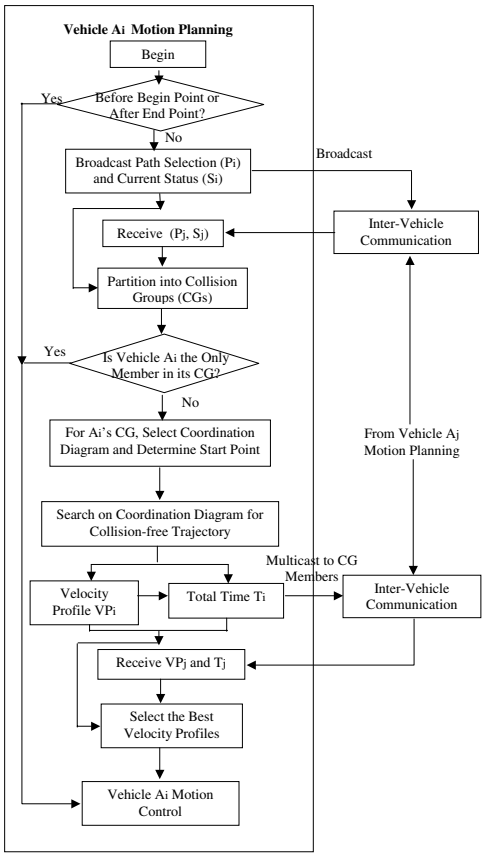

Fig. 7. The distributed motion planning algorithm.

based on the path selections and partition all the involved vehicles into *Collision Groups (CGs)*, where a Collision Group is the collection of vehicles whose path has at least one intersection with any path selected by another member in the same CG. If vehicle A_i is the only member in the CG, which means that A_i's selected path does not intersect any other vehicle's path, A_i keeps its original velocity. Otherwise, vehicle A_i and its CG members will coordinate their velocities such that they can pass through the intersection without collision and also with minimum time. The overall algorithm is shown in Figure 7.

The velocity coordination is done on a *Coordination Diagram* [13], which is an N-dimensional space representing the coordination space of the N vehicles in the CG which includes A_i. Each dimension of the Coordination Diagram represents the length that each vehicle travels. The obstacles in the Coordination Diagram represent the area where two of the vehicles will have a collision. The goal of the velocity coordination is to find a collision-free path on the Coordination Diagram so that the start and goal point are connected. Depending on the time instant when the vehicles receive the latest broadcasting, the start point varies on the Coordination Diagram. The location of the start point on the Coordination Diagram represents the location of the vehicles when they begin to

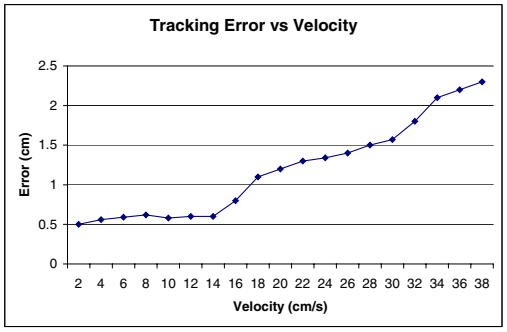

Fig. 8. The average tracking error vs. vehicle velocity in single lane tracking.

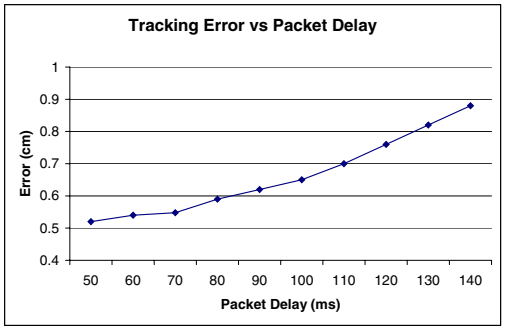

Fig. 9. The tracking error vs. packet delay in single lane tracking.

communicate with each other at the latest time. The goal point represents that all the vehicles in the same CG arrive at their end points. It should be pointed out that in the case where multiple vehicles share the same end point, such as the path 1, 5 and 9 in Figure 5, the goal point on the Coordination Diagram is only a "virtual" point and does not imply that the vehicles (1, 2 and 3) should arrive at the end point at the same time. Instead, after passing the intersection point, each vehicle will keep its final velocity and no collision will happen.

To reduce the overall time of intersection passing, the shortest path on the Coordination Diagram should be found. Various algorithms exist for collision-free path searching, such as D^* search [14]. Once the shortest path is selected, it will be interpreted into velocity profile V_i of each vehicle and the corresponding total time T_i needed for all vehicles in the same CG to pass through the intersection will be calculated. Then it will be broadcast to all the other members in the same CG and the best velocity profiles that result in the minimum T_i will be selected as the final velocity profiles. The reason for this mutual decision is that different vehicles may find different shortest paths based on the algorithm they use. Another reason is that a mutual decision will make sure that all vehicles in the same CG will agree on the same velocity profiles.

V. Experimental Results

A. Single vehicle lane tracking

We implemented the single vehicle tracking algorithm and tested its tracking performance. The data rate from the ACTS server is 30Hz. The data rate from the PC to the miniature vehicle is about 20 packets per second. First, we evaluated the average tracking error with respect to the velocity of the vehicle. At each fixed velocity, the realtime tracking errors are recorded and used to calculate the average tracking error. As shown in Figure 8, the average tracking error increases as the vehicle velocity increases. Second, we evaluated the average tracking error with respect to the packet delay, which gets bigger as the vehicle motion control algorithm gets more complicated.

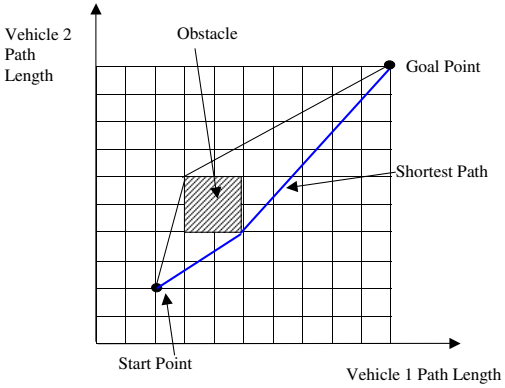

Fig. 10. The Coordination Diagram and the shortest path for vehicle 1 and 2 when path 2 and 5 are selected.

The tested packet-to-packet time delay is from 50ms to 140ms. Figure 9 shows the average tracking error vs. the packet delay time, where the velocity of the vehicle is fixed at 14 cm/s. From this figure it is clear that the tracking error increases in a roughly linear fashion as the packet delay increases.

The existence of the noticeable tracking error partially results from the limited servo control resolution and the lack of encoder feedback on the rear wheels. However, this tracking performance can still serve our purpose in studying the multiple vehicle coordination algorithm.

B. Multiple vehicle collision avoidance

For simplicity, we used two vehicles to illustrate the distributed motion control algorithm. The width of each lane is 14 cm. The communication range of each vehicle is set at 35 cm. As shown in Figure 5, when two vehicles (1 and 2) are passing through the intersection via path 2 and path 5, the Coordination Diagram is shown in Figure 10. When the same two vehicles are passing through the intersection via path 1 and path 5, the Coordination Diagram is shown in Figure 11. To achieve enough safety margin, we use a square box with sufficient dimension to represent the obstacle on the Coordination Diagram. To find a shortest path that connects the start point with the goal point on the Coordination Diagram, a *Reduced Visibility Graph* [19]

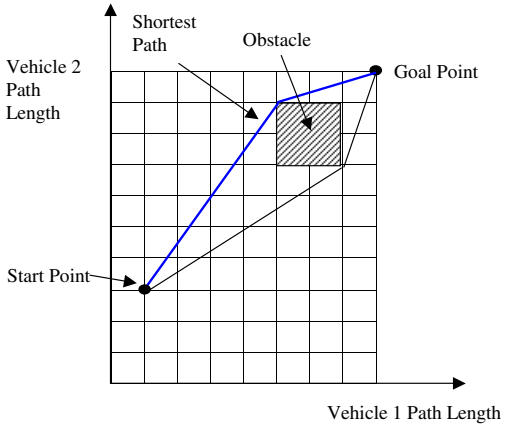

Fig. 11. The Coordination Diagram and the shortest path for vehicle 1 and 2 when path 1 and 5 are selected.

is constructed that connects the start point with the goal point. Then on the Reduced Visibility Graph, a shortest path is searched using Dijkstra's algorithm [20]. For the case when path 2 and 5 are selected, the testing runs prove that vehicle 1 speeds up while vehicle 2 slows down when approaching the intersection. The two vehicles safely pass through the intersection. The running steps are shown in Figure 12.

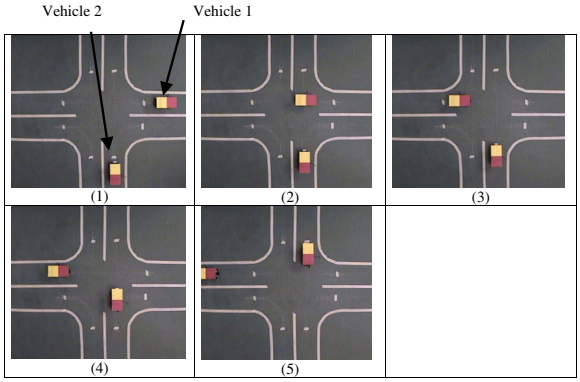

Fig. 12. The two vehicles pass through the intersection.

VI. Conclusions

In this paper, an experimental platform for cooperative driving using vehicle-to-vehicle communication is developed. The compact form factor and the flexibility of the COTS-BOTS robot make it a suitable vehicle for this platform. The vision module is employed to localize the vehicles, which can be replaced by the GPS or other localization units in real ITS environment. Single lane tracking experiments verify that the tracking performance is satisfying, although better servo motor and an encoder velocity feedback will certainly improve the tracking performance. A new algorithm for multiple vehicle collision avoidance is proposed and validated using two vehicles. This algorithm improves existing mobile robot motion coordination algorithm so that it fits in our application. This platform will certainly lead us to study and test more collision avoidance algorithms using vehicle-to-vehicle communication. In our future work, we will conduct experiments for scenarios that involve more than 2 vehicles. Also, another type of miniature vehicle that has much more computational power than the current one is under development. The goal is to shift the control processes from the PC to the miniature vehicles so that they will be fully autonomous and the experiments will be even closer to the real vehicle scenarios.

References

[1] P. Varaiya. Smart cars on smart roads: Problems of control. Technical report, University of California, Berkeley, TECH MEMO-91-05, Berkeley, CA, 1991.

[2] D. Bradley, S. Thayer, A. Stentz, and P. Rander. Vegetation detection for mobile robot navigation. Technical report, Robotics Institute, Carnegie Mellon University, CMU-RI-TR-04-12, Pittsburgh, PA, 2004.

[3] *Automatic Vehicle Guidance: the experience of the ARGO Autonomous Vehicle*. World Scientific Co. Publisher, Singapore, 1999.

[4] Sadayuki Tsugawa. An introduction to demo 2000: The cooperative driving scenario. *IEEE Intelligent Transportation Systems, July/Aug*, 4:78–79, 2000.

[5] R. Miller and Q. Huang. An adaptive peer-to-peer collision warning system. In *Proceedings of Vehicular Technology Conference (VTC) Spring 2002*.

[6] X. Yang, J. Liu, F. Zhao, and N. Vaidya. A vehicle-to-vehicle communication protocol for cooperative collision warning. In *Proceedings of the First Annual International Conference on Mobile and Ubiquitous Systems: Networking and Services (MobiQuitous'04)*, 2004.

[7] E. Gorman. *Using group communication to support inter-vehicle coordination*. PhD thesis, Department of Computer Science, University of Dublin, 2002.

[8] Lynne E. Parker. Current state of the art in distributed autonomous mobile robotics. In *Distributed Autonomous Robotic System 4*, pages 3–12, October 2000.

[9] K. Fujimura. *Motion Planning in a Dynamic Environment*. Spring Verlag, Tokyo, 1991.

[10] B. H. Lee and C. S. Lee. Collision-free motion planning of two robots. *IEEE Transactions on Systems, Man and Cybernetics*, 17(1):21–32, 1987.

[11] S. Leroy, J.P. Laumond, and T. Simeon. Multiple path coordination for mobile robots: a geometric algorithm. In *Proceedings of the International Joint Conference on Artificial Intelligence*, 1999.

[12] T. J. Pan and R. C. Luo. Motion planning for mobile robots in a dynamic environment. In *Proceedings of the International Conference on Robotics and Automation*, 1990.

[13] Y. Guo and L. E. Parker. A distributed and optimal motion planning approach for multiple mobile robots. In *Proceedings of the International Conference on Robotics and Automation*, 2002.

[14] A. Stentz. Optimal and efficient path planning for unknown and dynamic environments. Technical report, Carnegie-Mellon Technical Report CMU-RI-TR-93-20, Pittsburgh,PA, 1993.

[15] S. Bergbreiter and K. S. J. Pister. Cotsbots: An off-the-shelf platform for distributed robotics. In *Proceedings of the 2003 IEEE/RSJ Conference on Intelligent Robots and Systems*, pages 1632–1637, 2003.

[16] *http://www.xbow.com/Products/productsdetails.aspx?sid=3*.

[17] *TinyOS, http://www.tinyos.net/*.

[18] *http://robots.activmedia.com/ACTS/*.

[19] J.C.Latombe. *Robot Motion Planning*. Kluwer Academic Publishers, Boston, 1991.

[20] E.Dijkstra. A note on two problems in connection with graphs. *Numerical Mathematics*, 1:269–271, 1959.