A Third Generation Micro-vehicle Testbed for Cooperative Control and Sensing Strategies

M. Gonzalez, X. Huang, B. Irvine, D. S. Hermina Martinez, C. H. Hsieh, Y. R. Huang,
M. B. Short, and A. L. Bertozzi

***Abstract*— This paper describes the third generation of an economical cooperative control testbed, last described in [K. K. Leung *et al*., Proc. Am. Cont. Conf., 2007]. The new testbed vehicles are improved with powerful on-board computing, upgraded and expanded on-board sensing, and enhanced wireless communication, while maintaining economic feasibility and scale. The new hardware allows for increased autonomy of each vehicle and for the implementation of new, more advanced algorithms that rely on increased computational capability. We demonstrate practical use of the testbed for algorithm validation by implementing both previous and new cooperative steering and barrier avoidance algorithms.**

I. Introduction

The motivations behind the development of algorithms for multi-agent cooperative behavior have roots in a variety of disciplines. For example, unmanned aircraft reduce the risks put on human lives in hazardous environments and combat zones, and greater development of autonomous motor vehicle behavior could greatly reduce the number of traffic accidents, of which the vast majority are caused by human error. In addition, increases in mechanical autonomy have already reduced the need for many human operators in industry and commerce, and further work in this field can only accelerate their efficiency. The need to understand these problems has therefore resulted in the construction of many laboratory testbeds [4], [11], [1], [9], [7].

Laboratory testing of cooperative control algorithms is important to the development of the field because it brings real-world sensor issues, communication issues, and movement issues to the forefront of the research. One of the biggest challenges for laboratory testbeds, however, is lack of adequate space. Rarely do users have access to a dedicated area large enough to allow for the testing of multi-vehicle path planning algorithms with vehicles that possess the capacity for on-board computing. Typically, such a vehicle footprint would be at least the size of a laptop computer, necessitating a testbed arena on the order of 10 m across or more for meaningful experiments.

To avoid such space constraints, we have focused on the development of a micro-sized testbed that allows for testing of complex algorithms with vehicles that have a footprint smaller than a typical mobile phone. This requires the design and implementation of custom vehicles as well as careful thought into how algorithms are developed and managed. In prior work [3], [5], we developed a robust testbed based on modified toy cars that was successful in testing cooperative control algorithms, but also suffered from some major disadvantages. For example, the first generation testbed [3] required a centralized computer to perform all algorithm processing and send commands to the micro-cars, which lacked any form of on-board computing or sensors. The second generation [5] added IR range sensors and very modest on-board computing to the microcars, but they were still reliant on a processing computer for any advanced algorithms. This paper describes a third generation vehicle that is completely customized with vastly increased computational power, allowing not only for on-board algorithm processing but also for realtime user interaction via a remote terminal, as well as an array of sensors and enhanced communication capabilities. All this is done while still maintaining the compact footprint of prior work, avoiding space constraint issues.

The remainder of the paper is organized as follows: Section II describes the setup of the testbed and the new generation micro-car hardware and software, Section III presents some of the experiments that have been implemented on the testbed, and Section IV outlines future testbed goals and conclusions.

II. Testbed Setup and Hardware

The UCLA Applied Mathematics Laboratory Testbed (AMLT) is divided into three subsystems: an overhead-camera and PC tracking system, a remote terminal PC, and the micro-car robotic vehicles (Fig. 1). Physically, the testbed is a 1.5 m x 2.0 m rectangular area in which the micro-cars operate. The area itself is made of black asphalt felt paper with a white boundary, providing a uniform, non-reflective background for imaging purposes. The cars' positions are tracked by 2 overhead Imaging Source DMK 21F04 1/4 Monochrome CCD cameras with a resolution of 640 x 480 pixels. They have a frame rate of 30 fps and are connected to an image processing PC via firewire cable. The cars are identified using an OpenCV contour searching function that

This paper is supported by ARO MURI grant 50363-MA-MUR and NSF grants DMS-0914856, DMS-0907931, DMS-0601395, and EFRI-1024765.

M. Gonzalez is with the Department of Engineering, Harvey Mudd College, Claremont, CA 91711, U.S.A. E-mail: max_gonzalez@hmc.edu.

X. Huang and D.S. Hermina Martinez are with the Department of Electrical Engineering, University of California, Los Angeles, CA 90059, U.S.A. E-mail: {xinhenghuang, dherminamartin}@ucla.edu.

B. Irvine, M.B. Short, and A.L. Bertozzi are with the Department of Mathematics, University of California, Los Angeles, CA 90059, U.S.A. E-mail: jirvine89@ucla.edu, {mbshort, bertozzi}@math.ucla.edu.

C.H. Hsieh and Y.R. Huang are with Anteros Labs, Inc, Torrance, CA 90505, U.S.A. E-mail: {chsieh, rhuang}@anteroslab.com.

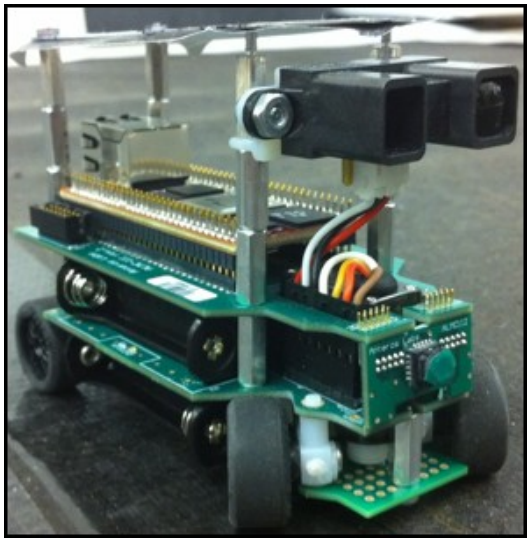

Fig. 1. One of the ALMC-100 micro-cars.

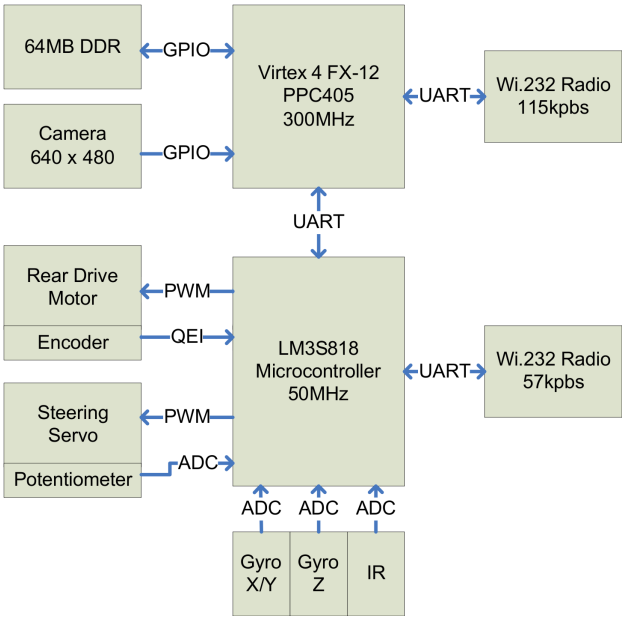

Fig. 2. A block diagram of the ALMC-100 micro-car hardware.

recognizes black and white ID tags, or "hats" , that are fixed atop each micro-car, giving each car's current position and orientation [3], [5]. This information is then broadcast through a serial radio module to be received by the cars on the testbed, for use in control algorithms. This mimics the functionality of a GPS unit that may be present in more advanced vehicles in the field.

The cars are given commands and can relay status information by communicating via a separate serial radio with an interface PC that serves primarily as a remote terminal for the cars. The ability to broadcast messages to the interface PC for display proves to be a very useful debugging tool.

A. *Vehicle Hardware*

The third generation micro-cars (model ALMC-100, see Fig. 2 for a hardware schematic) are purpose built from the ground up, in contrast to previous generation vehicles that were modified off-the-shelf toy cars. The ALMC-100 is designed to mimic many of the features one would expect to find in a full sized autonomous vehicle, in a compact package. The vehicle measures approximately 8 cm (wheelbase) x 5 cm (width); the height varies from 5.8 cm to 8 cm depending on configuration. The ALMC-100 is a rear wheel drive, front steering car with a maximum speed of 20 cm/s and maximum turning angle of $\pm 18°$. Power comes from four AAA batteries with approximately 3.8 W, yielding a run time of greater than 30 minutes.

The ALMC-100 features two processing units on individual printed circuit boards, which are stacked atop each other. The lower "chassis" board is the base of the vehicle where the drive train is bolted in addition to the electronics. The chassis board contains a 50MHz ARM Cortex-M3 processor with 8KB SRAM and 64KB flash memory. A 1KB EEPROM is also included to store unique, non-volatile information such as vehicle identification number and motor control gains and offsets. The chassis board also houses two gyroscopes for 3-axis measurements, a $0.45°$ optical encoder used for velocity estimation, and is attached to either a long-range or short-range IR module; the long range sensors can detect objects in the 10 cm - 140 cm range, while the short range sensors can only detect from 10 cm - 80 cm.

The upper "processing" board contains a Virtex4 FX-12 FPGA with a PowerPC 405 processor running at 300MHz. The PPC405 has access to 64MB DDR SDRAM and 4MB of flash memory, as well as a 640x480 camera.

The wireless communication system consists of two Wi.232 radio modules, one on each board, capable of transmitting and receiving at 11520 bps. The wireless module on the chassis board is configured to 57600 bps and receives only information from the overhead tracking system, mimicking GPS. The wireless module on the processing board is configured to 115200 bps and is intended for inter-vehicle communication and for access of the vehicle via the remote terminal. The two radios operate on different frequencies to avoid interference.

The driving factor behind the use of two processing units is to segregate motion control and path planning. The motion control is accomplished on the chassis board, which maintains its control loop at 1000 Hz while sampling the various sensors at 500 Hz. The chassis processor extracts the vehicle's own position from the overhead tracking system's broadcast sent at 30 Hz. The vehicle's position and other vehicle and sensor states are relayed to the processing board also at 30 Hz over the universal asynchronous receiver/transmitter (UART) connecting the two boards. Thanks to the powerful processing available to the upper board, the cars can perform all required path planning themselves; in previous versions of the AMLT, vehicles relied on a desktop computer to perform all such calculations and relay instructions to the cars.

B. Vehicle Software

Each vehicle runs two separate controlling programs. The chassis board controller is based on FreeRTOS [10], a realtime operating system that can run multiple tasks at up to 1000 Hz, and is designed to provide two main functionalities: control the vehicle's motion by providing PWM signals for the drive motor and the servo, and supply sensory data and system information for the processing board. At the center of the chassis board controller is a path-generation task that calculates target velocities at 100 Hz and generates PWM signals at 50 Hz that feed the servo. Based on the target velocities, a velocity-control task generates a 1000Hz PWM signal to control the drive motor. The controller is also designed to recognize a set of serial commands, thus allowing the processing board to control high-level motion and access data.

The processing board boasts much deeper memory space and faster processing speed, and can thus support more complex programs that can feature user-friendly interfaces and large data-sets. Currently, this controller serves as a remote terminal interface [2], an example of which is shown in Fig. 3. In order to interface with the user, the micro-

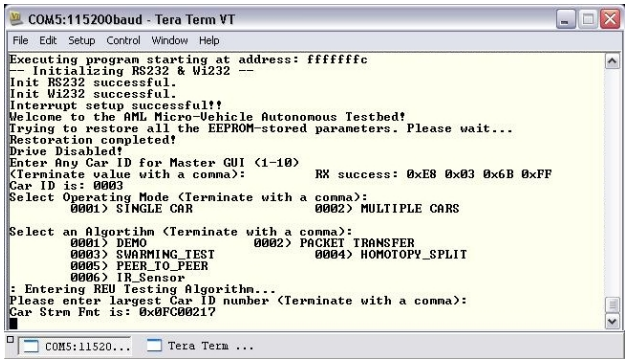

Fig. 3. A typical screenshot of the remote terminal on the interface PC, showing several menus in its structure. Near the bottom, the multiple car mode menu is shown with several of the algorithm selections visible.

car transmits messages to the interface PC, via its upper serial radio. If many cars are in use at once, only one car, chosen by the user, broadcasts the terminal messages. The user interacts with the micro-cars through the interface PC's keyboard. These inputs are transmitted to the micro-cars, which then execute the chosen algorithms. The interface is composed of two modes: a single car mode for demonstrating basic operation of the vehicles, and a multiple car mode to implement cooperative algorithms.

III. Testbed Experiments

The motion of the cars is modeled based on a Frenet-Serret framework as in the second-generation testbed [8]. Each car k has its own coordinate frame relative to its heading, with $\mathbf{x}_k$ being the unit vector oriented in the direction the car's motion and $\mathbf{y}_k$ being the unit vector oriented perpendicular to $\mathbf{x}_k$ (Fig. 4). The car's motion can then be described with the following equations:

$$\dot{\mathbf{z}}_k = \mathbf{x}_k, \qquad \dot{\mathbf{x}}_k = v u_k \mathbf{y}_k, \qquad \dot{\mathbf{y}}_k = -v u_k \mathbf{x}_k \ , \tag{1}$$

where $\mathbf{z}_k(s)$ is the arclength parametrized path of the car with respect to a fixed coordinate frame, the scalar u_k is the curvature of the path at a specific point, and v is the (typically constant) speed of the vehicle. Thus, the path of each car can be determined simply by specifying its curvature over time. To convert from curvature u_k to the desired turning angle ϕ_k of the vehicle's wheels, we use the equation

$$\phi_k = \tan^{-1}(u_k L_{\text{car}}) \ , \tag{2}$$

where L_{car} is the car length of 8 cm. If $|\phi_k| > 18°$, the maximum turning angle that the servos can turn, $|\phi_k|$ is rounded down to the maximum. This limits the minimum turning diameter to approximately 50 cm, a notable constraint for the testbed, which is 1.5 m x 2.0 m.

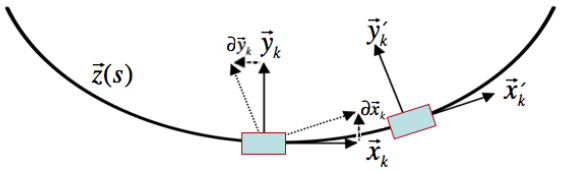

Fig. 4. Diagram of the coordinate frame of a micro-car moving along a parametrized path $\mathbf{z}(s)$.

A. Cooperative Motion Algorithms

From Morgan & Schwartz's model for swarming [8], the curvature for each car k is calculated as

$$u_k = \sum_{j \neq k} u_{jk}, \tag{3}$$

where j cycles through the indices of all the other cars on the testbed. For simple swarming, the following equation for u_{jk} is used:

$$u_{jk} = -\eta \left(\frac{\mathbf{r}_{jk}}{\|\mathbf{r}_{jk}\|} \cdot \mathbf{x}_k \right) \left(\frac{\mathbf{r}_{jk}}{\|\mathbf{r}_{jk}\|} \cdot \mathbf{y}_k \right) - \alpha \left[1 - \left(\frac{r_0}{\|\mathbf{r}_{jk}\|} \right)^2 \right] \left(\frac{\mathbf{r}_{jk}}{\|\mathbf{r}_{jk}\|} \cdot \mathbf{y}_k \right) + \mu \mathbf{x}_j \cdot \mathbf{y}_k, \tag{4}$$

where $\mathbf{r}_{jk}$ is the vector from car j to car k, r_0 is the desired distance between cars for the swarm, and α, η, and μ are weighting parameters for three separate aspects of the desired motion. The term with coefficient η works to turn each car perpendicular to $\mathbf{r}_{jk}$; the term with coefficient α turns the cars toward each other if they are further than r_0 apart, and turns them away from each other if they are closer than r_0; and the term with coefficient μ orients the cars toward a common heading. By varying the three weights of these terms and introducing slight modifications, different cooperative motion can be achieved, such as the circle-tracking, leader following, and homotopy control laws described in [5].

The summation in Eq. 3 is typically over all cars, leading to global coupling. However, such coupling should rarely be expected in real-world scenarios, where each vehicle may only know its own position and perhaps the positions of a

few other nearby vehicles. Therefore, in addition to global coupling, we have also tested a form of daisy-chain coupling whereby each car k is only coupled to the two cars with indices $j = k \pm 1$, as illustrated in Fig. 5. The implementation

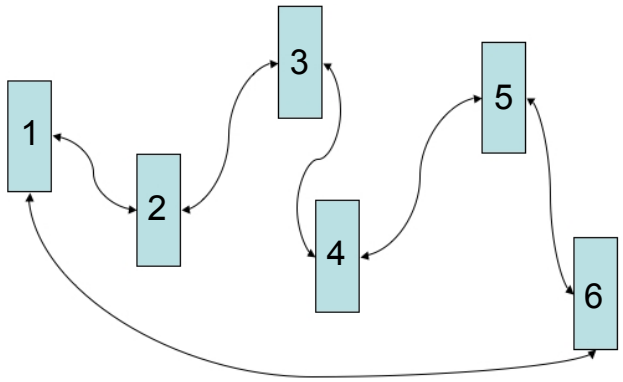

Fig. 5. Diagram of daisy-chain coupling with the cars iteratively connected. End conditions were used to create a closed loop.

of this daisy-chain algorithm on larger swarms has yielded promising results on the testbed. Although each car is only aware of two others, the swarm operates as a whole because of the iterative coupling utilized by the daisy-chain system. The performance of the swarm is somewhat dependent on initial placement of the micro-cars, however, a behavior noted in similar systems [6]. If the cars are ordered by ID, the daisy chain swarm performs excellently; the cars rarely collide and find a common heading quickly. However, if they are placed in a random order, collisions may occur as the cars are not necessarily aware of their closest neighbors, which may no longer be the cars they are coupled to. This especially occurs if the cars are initially placed in close proximity to each other, typically less than approximately two car lengths. However, if groups of cars are initially separated by distances larger than this, they are usually able to regroup and find a common heading with very few collisions, as shown in Fig. 6.

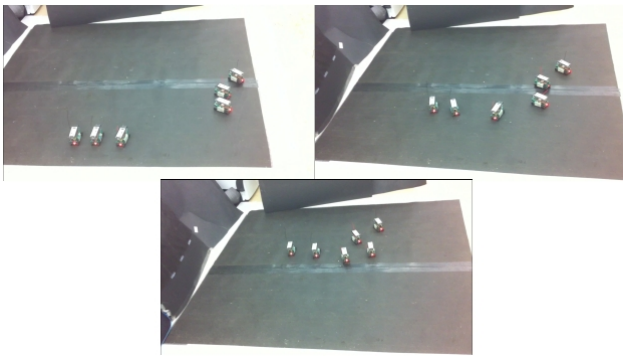

Fig. 6. Three frames of an experimental run using daisy-chain coupling. The cars are originally separated in two groups (top left). During the run, the cars regroup (top right) and find a common orientation before exiting the testbed (bottom).

In addition to the fixed daisy-chain based on car index numbers described above, we have also implemented an algorithm that creates a closed daisy-chain upon algorithm startup, based on initial vehicle locations. For each car k, the algorithm determines the two cars "connected" to k in the following way:

- Partition all cars into two groups – those to the left of k (subset $\mathbb{L}_k$) and those to the right of k (subset $\mathbb{R}_k$), based upon the position and orientation of k and the positions of all other cars.
- If $\mathbb{R}_k$ is not empty, partner one is the physically closest member of $\mathbb{R}_k$. Otherwise, partner one is the physically furthest member of $\mathbb{L}_k$.
- If $\mathbb{L}_k$ is not empty, partner two is the physically closest member of $\mathbb{L}_k$. Otherwise, partner two is the physically furthest member of $\mathbb{R}_k$.

This algorithm allows the cars to create a daisy-chain that operates well under a broader range of initial conditions, since partners are chosen at startup in such a way as to generally minimize the initial distances between partners while maintaining the closed loop that allows the chain to function as a whole.

B. Barrier Avoidance and Target Seeking

A control law similar to Eq. 4 can be used to avoid barriers and seek a target. This is accomplished with the equation

$$u_{jk} = \gamma \left[1 - \left(\frac{r_0}{\|\mathbf{r}_{tk}\|}\right)^2\right] \left(\frac{\mathbf{r}_{tk}}{\|\mathbf{r}_{tk}\|} \cdot \mathbf{y}_k\right) - \beta \text{ sign} \left[\sum_b C\left(\mathbf{r}_{bk}, 0, \omega\right) \left(\frac{\mathbf{r}_{bk}}{\|\mathbf{r}_{bk}\|} \cdot \mathbf{y}_k\right)\right] \left[\sum_b C\left(\mathbf{r}_{bk}, 0, \omega\right) \left(\frac{\mathbf{r}_{bk}}{\|\mathbf{r}_{bk}\|} \cdot \mathbf{x}_k\right)\right] , \quad (5)$$

where

$$C(\mathbf{r}, q, \omega) = \begin{cases} 1 & \|\mathbf{r}\| < \omega \\ q & \text{otherwise} \end{cases} , \quad (6)$$

$\mathbf{r}_{tk}$ is the vector from car k to the target, $\mathbf{r}_{bk}$ is the vector from car k to the barrier b, and γ and β are the weights assigned to the target seeking and barrier avoidance behaviors, respectively.

Eq. 5 is dependent on the car's knowledge of the position of the target and of all barriers within a certain distance ω of itself. This has been implemented on our testbed in two ways. The first uses a hard-coded target position and makes use of hats sitting on the barriers. The tracking cameras pick up the barrier locations and relay the information to the target-seeking car, which then weaves its way through the obstacles to its goal. Several photos of such an experiment are shown in Fig. 7.

Ideally, though, the cars would not rely on the tracking computer to provide them with information on the location of barriers, but would instead use their IR sensors to detect barriers on the fly and adjust their movement accordingly; our second implementation of barrier avoidance does just this. It again uses a hard-coded target position, but now uses the IR sensor readings of the cars to estimate values

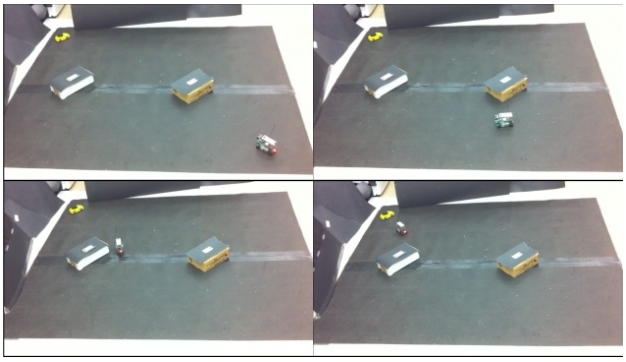

Fig. 7. Four frames of an experimental run of tracking camera-assisted barrier avoidance, with the yellow object in the top left of each frame as the target. The frames are ordered from left to right, top to bottom. The car detects the first barrier in the second frame and begins to avoid it. It continues toward the target and avoids the second barrier in frame three and reaches the target in frame four.

for $\mathbf{r}_{bk}$. Since the sensors only work over a limited range, this naturally incorporates a term such as $C(\mathbf{r}_{bk}, 0, \omega)$ into the behavior. One disadvantage of this method is that only barriers nearly directly ahead of the car can be sensed, so that cars may occasionally turn into nearby barriers that lie at their sides. We find, though, that the relatively slow motion and large turning radius of the cars minimizes these issues, and that swarms of cars employing this algorithm, in conjunction with Eq. 4, typically navigate the barriers successfully, as show in Fig. 8

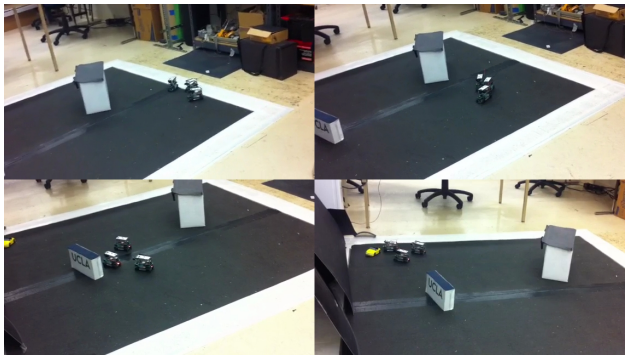

Fig. 8. Four frames of an experimental run of IR sensor barrier avoidance, with the yellow object near the top left of frames three and four as the target. The frames are ordered from left to right, top to bottom. The cars detect the first barrier in the second frame and avoid it. They continue toward the target and avoid the second barrier in frame three and reach the target in frame four.

IV. Future Work and Conclusions

Though the third generation of the AMLT has already surpassed previous versions in terms of successfully performed experiments, there are still many aspects of the new hardware that have not yet been utilized. Foremost amongst these is the potential for a true peer-to-peer communication network for the vehicles. Creating such a network would allow for the implementation of many advanced algorithms. For example, it would allow for the vehicles in our daisy-chain setup to determine their nearest neighbors in realtime, decreasing the chances of collisions occurring.

Another avenue of exploration is greater usage of the many on-board sensors. The optical encoder and gyroscopes could allow each car to estimate its own position over time with less reliance on the tracking computer, assuming the car's initial coordinates are known. This would mimic the intermittent GPS outages expected in field vehicles, and give methods for dealing with them. Also, the on-board camera could perhaps be used in conjunction with the IR sensor for enhanced barrier detection, or possibly target detection when the location of the target is unknown.

V. Acknowledgments

We would like to thank Ahbijeet Joshi for helpful comments.

References

[1] D. Cruz, J. McClintock, B. Perteet, O. A. A. Orqueda, Y. Cao, and R. Fierro, "Decentralized cooperative control - a multivehicle platform for research in networked embedded systems," *Control Systems Magazine, IEEE*, vol. 27, no. 3, pp. 58 – 78, jun. 2007.

[2] D. S. Hermina Martinez, "Integration of 3rd generation vehicles to the applied mathematics laboratory autonomous micro-vehicle testbed," Master's thesis, Department of Electrical Engineering, University of California, Los Angeles, 2010.

[3] C. H. Hsieh, Y. L. Chuang, Y. Huang, K. K. Leung, A. L. Bertozzi, and E. Frazzoli, "An economical micro-car testbed for validation of cooperative control strategies," in *Proceedings of the American Control Conference*, 2006, pp. 1446–1451.

[4] Z. Jin, S. Waydo, E. B. Wildanger, M. Lammers, H. Scholze, P. Foley, D. Held, and R. M. Murray, "MVWT-II: the second generation Caltech Multi-Vehicle Wireless Testbed," in *Proceedings of the American Control Conference*, 2004, pp. 5321 – 5326.

[5] K. K. Leung, C. H. Hsieh, Y. R. Huang, A. Joshi, V. Voroninski, and A. L. Bertozzi, "A second generation micro-vehicle testbed for cooperative control and sensing strategies," in *Proceedings of the American Control Conference*, 2007, pp. 1900–1907.

[6] J. A. Marshall, T. Fung, M. E. Broucke, G. M. T. D'Eleuterio, and B. A. Francis, "Experimental validation of multi-vehicle coordination strategies," in *Proceedings of the American Control Conference*, 2005, pp. 1091–1095.

[7] T. McLain and R. W. Beard, "Unmanned air vehicle testbed for cooperative control experiments," in *Proceedings of the American Control Conference*, 2004, pp. 5327 – 5331.

[8] D. S. Morgan and I. B. Schwartz, "Dynamic coordinated control laws in multiple agent models," *Physics Letters A*, vol. 340, pp. 121–131, 2005.

[9] S. Punpaisarn and S. Sujitjorn, "SUT-CARG car-like robots: their electronics and control architecture," *WSEAS Transactions on Circuits and Systems*, vol. 7, no. 6, pp. 579–589, 2008.

[10] Real Time Engineers Ltd., "FreeRTOS-A Free RTOS for ARM7, ARM9, Cortex-M3, MSP430, MicroBlaze, AVR, x86, PIC32, PIC24, dsPIC, H8S, HCS12 and 8051," http://www.freertos.org.

[11] A. E. Turgut, F. Gokçe, H. Celikkanat, L. Bayındır, and E. Sahin, "Kobot: A mobile robot designed specifically for swarm robotics research," Department of Computer Engineering, Middle East Technical University, Ankara, Turkey, Tech. Rep., 2007.