An Economical Micro-Car Testbed for Validation of Cooperative Control Strategies

Chung H. Hsieh[1], Yao-Li Chuang[3], Yuan Huang[1], Kevin K. Leung[1]
Andrea L. Bertozzi[3], Emilio Frazzoli[4]

***Abstract*— This paper describes the design of an economical cooperative control testbed using 1/64 size micro-car vehicles. The vehicles have similar motion constraints to those found in common kinematic models for UAVs. We demonstrate the practical use of this testbed for algorithm validation by implementing a recently proposed UAV-routing algorithm [J. J. Enright *et al* Proc. AIAA Conf. Guidance, Nav. and Control, 2005]. The experimental results lie within the theoretical bounds predicted for this algorithm.**

I. Introduction

This paper documents the construction of a small area, cost-effective, multiple vehicle testbed. With the widespread study of multiple vehicle coordination on the rise in academia and industry, a vehicle testbed is often necessary to validate the effectiveness of the algorithms. Vehicle testbed are also an invaluable learning tool for students to see theory in action. While many such testbeds exist [1-7], cost per vehicle can be large, on the order of thousands of dollar per vehicle, which can make multi-vehicle studies with more than five vehicles prohibitive. In addition, many of the current testbeds require a large area to accommodate a number of vehicles with non-holonomic motion constraints.

Our testbed addresses both issues by using very small and inexpensive radio controlled cars (off-the-shelf cost of approximately $30 per vehicle) on a modest size driving surface. We first present all the physical dimensions of the testbed and its subsystems, including an analysis of measurement errors and a mathematical vehicle model. Second, as a demonstration of the testbed's potential, we implement a path planning algorithm for UAV vehicle routing.

II. Multiple micro-vehicle testbed

A. System overview

The testbed is comprised of three main systems (Fig. 1)

1) Vehicles
2) Position Tracking System
3) Control Computer

The vehicles are converted from standard 1/64th scale radio-controlled cars. The Position Tracking System includes the an $1.5m \times 2.0m$ arena and two monochrome CCD cameras.

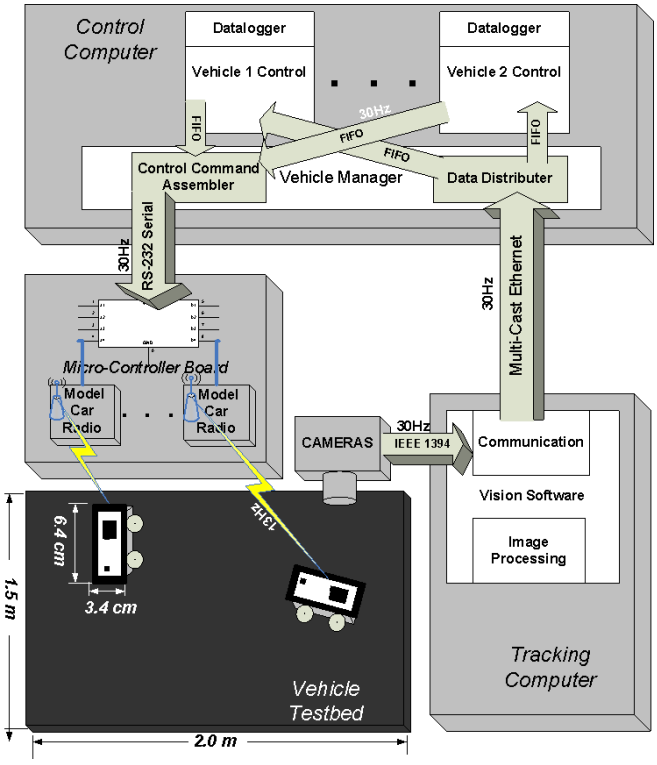

Fig. 1. A system overview of the testbed. All physical components are shown as 3-D objects, software objects are represented in 2-D. The arrows indicate communication connection and direction. The communication rate is shown next to the arrows.

The vision software is OpenCV for Windows Beta5 [8] with DirectShow Software Development Kit [9] for the camera interface. For the controls computer we use Linux OS and programs in standard ANSI C, with POSIX system calls.

B. Vehicle design

We re-engineered off-the-shelf products suitable for the size and scope of the test bed. We chose Hobbico microsizers radio control (RC) cars (1:64) for their low cost, compactness, weight, minimal modification requirements, and ease of purchase. These cars operate on two carrier frequencies 27MHz and 49MHz using a Time Division Multiple Access (TDMA) transmission scheme. The commands are updated at 13Hz. Each channel is capable of controlling three cars simultaneously to allow up to six vehicles. We made slight modifications to both the vehicle and the radio transmitter to match the needs of the test bed. Out of the box, the

[1]Dept. of Electrical Engineering, University of California Los Angeles, Los Angeles, CA 90095 chung_hsieh@ieee.org, kevinlg@icedepot.net, yuanh@seas.ucla.edu
[3]Department of Physics, Duke University, Durham, NC 27708, Dept. of Mathematics, University of California Los Angeles, Los Angeles, CA 90095 {chuang, bertozzi}@math.ucla.edu
[4]Dept. of Mechanical and Aerospace Engineering, University of California Los Angeles, Los Angeles, CA 90095 frazzoli@ucla.edu

Dimension (L x W x H)	(6.4 x 3.4 x 3.2)cm
Weight (with batteries)	50g
Steady State Speed	60 cm/s
Acceleration (from zero)	78 $\mathrm{cm/s}^2$
Turning Radius	11-13 cm
Angular velocity	3.3 rad/s

TABLE I

SPECIFICATIONS OF CURRENT TESTBED VEHICLE

Fig. 2. (Left) An unprocessed shot of the testbed. (Right) Partial screen shot of the processed image. The large box surrounds the outer region of the vehicle. The medium size rectangle is the heading box. The small squares denotes the binary identification boxes. In the figure we see car #2 and car #3.

vehicles have only five minutes of continuous run time. To extend the run time to 30 minutes, we replaced the original battery with two AAA batteries secured to the top of the vehicle. We also replaced the transmitter's button switches with electronic switches in order to incorporate this existing radio control architecture into our testbed. However, we did not increase the control update rate which remained at 13Hz. The vehicle has six states of motion: forward left, forward right, backward left, backward right, forward, and backward. Mechanical limitations apply to both the steering and the velocity control. The steering control is limited to strictly right and left only with a fixed radius. The cars operate with a fixed forward or backward cruise velocity which limits the option of changing the effective velocity to pulsing. Physical characteristics of the vehicle are described in Table I.

C. Tracking system

The vehicle arena floor has a top layer of asphalt felt paper, which provides a uniform, non-glossy, black background for imaging. An overhead position tracking system is composed of two Imaging Source DMK 21F04 1/4" Monochrome CCD cameras with progressive scan of 30 images/sec at a resolution of 640 x 480 pixels. The cameras are connected via an IEEE 1394a (firewire) interface to a 3.0GHz PC with an IEEE1394 adapter card for image processing. The camera lenses are Pentax H612A(KA) with focal length of $6.0mm$ and 56° horizontal angle of view. The cameras are mounted 2.6 meters above the testbed resulting in an observable area for each camera of $1.5m \times 1.1m$, with each pixel representing a $2.4mm \times 2.4mm$ area. The cameras have a 0.144 m image overlap resulting in a total visible arena of $1.5m \times 2.1m$. Because the whole vehicle must be visible for tracking, the actual trackable region is slightly smaller, $1.4m \times 1.9m$.

We experienced only minor barrel distortion, common with all camera use. The difference of undistorted radius from center and distorted radius is 2 pixels resulting in an 0.8% error on position, which we considered to be small enough to not require a correction in our data. For larger barrel distortions, a correcting filter would be needed before image processing.

The Vehicle Tracking System software utilizes Intel's computer vision software (OpenCV) [8]. We detect the vehicle position and identification using a black and white two dimensional bar code marking similar to that used on the CalTech Multi-Vehicle Wireless Testbed [1,2]. The vehicles are located with an OpenCV contour searching function and the following algorithm.

1) Record all contrasting shapes detected on the platform and bound them within a rectangle.
2) Sort the rectangles into three groups by size.
3) Assign the large rectangles as possible car object.
4) Go through medium sized rectangles:
   - if within bound of a large rectangle, mark as car's heading symbol.
5) Go through small sized rectangles:
   - if within bound of a large rectangle, mark as car's ID symbol.
6) Once all rectangles have been sorted and placed, evaluate the car object:
   - Position : The center of the large rectangle.
   - Heading : the vector formed by the center of the large and medium rectangle.
   - Identification : the distance formed by the center of the medium and small rectangles. The distance determines the bit position.

The processed data is then multi-casted via ethernet to the controller computers to close the feedback loop. The whole cycle of identifying and transmitting data for 10 different markers averages 15ms per frame.

D. Software architecture

As shown in Fig. 1, the Control Computer determines the motion of each vehicle. The current testbed architecture has off-board computation, however we have engineered the algorithm to operate as it if were running separately on each vehicle. To achieve the above goal, the system has $n+1$ programs to handle an $n-$vehicle experiment, and linux-based C is used because of its ready-to-use first-in-first-out (FIFO) functions for easy communication between programs. Each vehicle has its own independent path-planning algorithm, while an additional manager program is in place to take care of communication functions and positioning information.

The manager program bears two major responsibilities, as a data distributer and a control-command assembler. To accomplish the first task, it receives position data from the tracking computer, one vehicle at a time. Once all of the position data is received for one time frame, this information is sent to each of the n vehicle programs through an exclusive

FIFO. The individual vehicle programs then calculate their own control commands, which are then sent back to the manager program through a separate set of FIFOs. Upon collecting all the control commands, the manager program packages them into a string and sends them through a RS-232 cable to the Micro-Controller Board. For the application described in Section IV, with three vehicles and algorithms of modest computational complexity, it takes the manager about 0.5ms to finish all the above procedures. Considering that the tracking computer requires 15ms to send out a whole frame of data, there is still a comfortable margin to increase either the computational complexity of the path planning algorithms or the number of vehicles.

The Micro-Controller Board's micro processor receives a character string through the serial cable connected to the Control computer. It unpacks the string into sets of vehicle control commands, and distributes each set to the radio transmitter of the corresponding vehicle. Each transmitter then converts the control commands to radio frequency signals and broadcast it to the vehicle running on the platform.

III. Basic vehicle model

In this section we present a computer simulation model which is helpful for performing virtual experiments for comparison with the testbed. We assume that the vehicle engine has a constant output, the frictional force is linear, and that there is a fixed turning radius for each vehicle. The basic motion model is follows:

$$\begin{aligned} \dot{x} &= v\cos(\theta), \\ \dot{y} &= v\sin(\theta), \qquad (1) \\ M\dot{v} &= u_1 F - \beta v, \\ \dot{\theta} &= u_2 \frac{v}{R}, \qquad (2) \end{aligned}$$

where x and y represent the position coordinates of the vehicle in the laboratory frame, v is the vehicle speed (positive if going forward and negative if going backwards), θ indicates the angle of the vehicle heading, M is the vehicle mass, F is the constant driving force of the vehicle, and β is the friction coefficient with the floor. The parameter R is the turning radius. The input control parameter $u_1 \in \{-1, 0, 1\}$ corresponds to backward, zero, or forward acceleration. The input control parameter $u_2 \in \{-1, 0, 1\}$ corresponds to a right turn, straight motion, or a left turn. The above model is common in the literature on UAV motion planning and is similar to the one studied in [10] but with constant speed. In the UAV routing application described in section IV we fix $u_1 = 1$ and let u_2 be the only input control parameter.

In practice there are additional factors that affect vehicle motion. The left and right turning radii can be slightly different and varies with vehicle. Second, the vehicle alignment has a small offset resulting in a slight drift when commanded to go straight. Third, the friction can vary depending on the motion command (left vs. right vs. straight). To address these issues we let R and β depend on the direction of turn and also on the particular vehicle in motion.

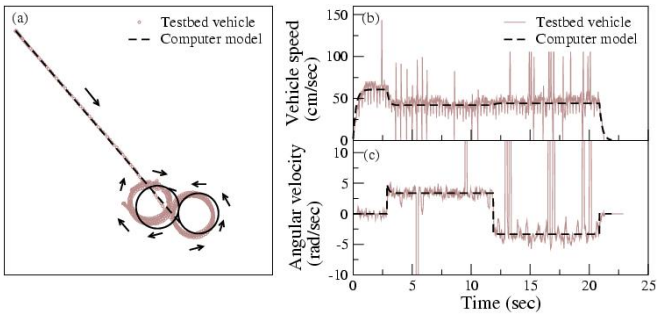

Fig. 3. Comparison of the open loop test to simulation model. (a) The trajectories of the vehicle. (b) Linear speed of the vehicle vs. time. (c) Angular velocity of the vehicle vs. time measured in radians per second. Parameters of the model: the left and right turn radii are respectively $R_L = 12.61\ cm$, $R_R = 12.73\ cm$. $F = 62.82\ M \cdot cm/sec^2$. The left, right, and straight motion friction coefficients are $\beta_L = 1.49\ M/sec$, $\beta_R = 1.42\ M/sec$, and $\beta_S = 1.04\ M/sec$, where M is the vehicle mass. The simulation also takes into account that there is a 0.17-second response delay of the vehicle to the control signals. Temporal data is filtered by a running 5-point average.

We compare the model with an open loop test on the testbed. During the test, the vehicle is ordered to go straight, then turn left, and finally turn right. Fig. 3 shows the comparison between the testbed data log and the computer simulated results. With the model parameters measured from the testbed, both trajectories agree as do the linear speeds and the angular velocities. Our model is second order in linear translation and first order in rotation. However, the equation for v typically has a very short characteristic time. In practice one can simplify the model to first order with a fixed velocity and with very similar results.

IV. A case study on UAV routing

As an example of the capabilities of the testbed, we report experimental results from the implementation of a very recent algorithm for UAV routing [11] and compare the observed performance to analytically-derived bounds. Consider a number of vehicles with constant speed and bounded curvature that must visit stochastically-generated targets in a convex, compact two dimensional plane. Targets appear according to a spatio-temporal Poisson process, uniformly in space. We want to minimize the expected waiting time between the appearance of a target and the time it is visited. We limit our analysis to the case in which targets appear infrequently.

A. Problem formulation

Let the environment $\mathcal{Q} \subset \mathbb{R}^2$ be a convex, compact set with unit area, and let $\|\cdot\|$ denote the Euclidean norm in $\mathbb{R}^2$. Consider $n \geq 1$ vehicles constrained to move at constant speed v along a path with bounded curvature, and let $1/\rho$ be the maximum curvature. Let the configuration $g_i \in SE(2)$ of the i-th vehicle ($1 \leq i \leq n$) be given in coordinates by $g_i = (x_i, y_i, \theta_i)$; the dynamics of each vehicle are given by the ODE (1)-(2).

The vehicles have unlimited range and target-servicing capacity. In the following, we will indicate by $p_i = (x_i, y_i)$

the position of the i-th vehicle. Moreover, we will indicate by $g = (g_1, g_2, \ldots, g_n) \in SE(2)^n$ and $p = (p_1, p_2, \ldots, p_n) \in \mathcal{Q}^n$ the configuration of the n-vehicle system, and the positions of the n vehicles, respectively.

Information on outstanding targets—the demand—at time t is summarized as a finite set of target positions $D(t) \subset \mathcal{Q}$, with $m(t) := \mathrm{card}(D(t))$. Targets are generated, and inserted into D, according to a homogeneous (i.e., time-invariant) spatio-temporal Poisson process, with time intensity $\lambda > 0$, and uniform spatial density. In other words, given a set $\mathcal{S} \subseteq \mathcal{Q}$, the expected number of targets generated in $\mathcal{S}$ within the time interval $[t, t']$ is

$$\begin{aligned}\mathrm{E}\left[\mathrm{card}(D(t') \cap \mathcal{S}) - \mathrm{card}(D(t) \cap \mathcal{S})\right] = \\ = \lambda(t' - t)\mathrm{Area}(\mathcal{S}).\end{aligned}$$

(Strictly speaking, the above equation holds in the case in which targets are not being removed from the queue D.) Servicing of a target $e_j \in D$, and its removal from the set D, is achieved when the UAV moves to the target's position.

A static feedback control policy for the system is a map $\mu : SE(2)^n \times 2^{\mathcal{Q}} \to \{-1, 0, 1\}^n$, assigning a control input to each vehicle (u_2 in equation (2)), as a function of the current state of the system, i.e., $\omega_i(t) = \mu_i(g(t), D(t))$. (The subscript i denotes the component of the output vector, and of the control policy, that are relevant to the i-th vehicle.) The policy μ is stable if, under its action,

$$m_\mu := \lim_{t \to +\infty} \mathrm{E}\left[m(t) | \dot{g} = \mu(g, D)\right] < +\infty,$$

that is, if the UAV is able to service targets at a rate that is—on average—at least as fast as the rate at which new targets are generated.

Let T_j be the time that the j-th target spends within the set D, i.e., the time elapsed from the time e_j is generated to the time it is serviced. Define $T_\pi := \lim_{j \to +\infty} \mathrm{E}\left[T_j\right]$ as the steady-state waiting time under the policy π. Our objective is to minimize the steady-state waiting time, over all stabilizing feedback control policies, i.e.,

$$T^* = \inf_{\mu \text{ stable}} T_\mu.$$

We now summarize some results from [11], including bounds on the achievable system time T^* in light-load conditions, i.e., when $\lambda \to 0^+$, and a routing policy whose performance is provably within a constant factor from the optimal lower bound. In light-load conditions, UAVs which such constrained motion must have holding or loitering pattern while they are waiting for new targets. Optimal loitering patterns depend on the shape of the loitering region; for simplicity we consider *circular loitering patterns* which allow us to derive algorithms and bounds independent of the shape of the environment.

Define

$$H_n(p, \mathcal{Q}) = \int_{\mathcal{Q}} \min_i \|q - p_i\| \, dq,$$

to be the expected distance between p and a point q randomly sampled from a uniform distribution on $\mathcal{Q}$. The function H_n is the *continuous Weber function* or the *continuous multi-median function* ([12], [13] and references therein), and is equivalent to

$$H_n(p, \mathcal{Q}) = \sum_{i=1}^{n} \int_{\mathcal{V}_i(p)} \|q - p_i\| \, dq$$

where $\mathcal{V}(p) = \{\mathcal{V}_1(p), \mathcal{V}_2(p), \ldots, \mathcal{V}_n(p)\}$ is the Voronoi partition of the set $\mathcal{Q}$ generated by the points p. In other words, $q \in \mathcal{V}_i(p)$ if $\|q - p_i\| \leq \|q - p_k\|$, for all $k \in \{1, \ldots, n\}$. The set $\mathcal{V}_i$ is referred to as the Voronoi cell of the generator p_i. The n-median of the set $\mathcal{Q}$ is the global minimizer

$$p_n^*(\mathcal{Q}) = \arg \min_{p \in \mathcal{Q}^n} H_n(p, \mathcal{Q}).$$

In this paper, we associate to each vehicle a unique and distinct Voronoi cell. Fig. 4 shows the Voronoi cells used for our testbed with one, two, and three vehicles. Under the conditions described above, we have [11]

Theorem 1: The system time T^* satisfies:

$$T^* \geq \frac{H_n^*(\mathcal{Q})}{v} + \left(\frac{\pi}{2} - \frac{2}{\pi}\right)\frac{\rho}{v} \approx \frac{H_n^*(\mathcal{Q})}{v} + 0.9342\frac{\rho}{v}. \quad (3)$$

Consider the following control policy, which we call the offset median (OM) policy. Let c^* be the n-median of $\mathcal{Q}$, and define the *loitering state* for the i-th agent as a circular trajectory of radius $\alpha\rho$ centered at c_i^*, with $\alpha \approx 2.918$. In the offset median policy, each agent visits all targets in its own Voronoi region $\mathcal{V}_i(c^*)$ in a greedy fashion: in other words, it always pursues the closest target in a Dubins' distance sense. When no targets are available, it returns to its loitering state; the orbit orbit direction chosen to minimize the time to return to the state.

Theorem 2: An upper bound on the system time of the Offset Median policy in light load is

$$T_{\mathrm{OM}} \leq H_n^*(Q)/v + \gamma\rho/v \text{ as } \lambda \to 0, \quad (4)$$

with $\gamma = 3.756$.

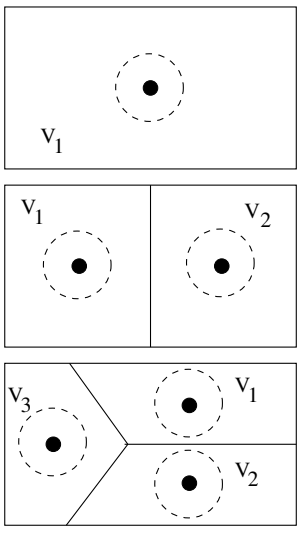

Fig. 4. Loitering regions (Voronoi partition) for the testbed with 1 (top), 2 (middle) and 3 (bottom) vehicles. The dashed lines show the loitering path for each vehicle.

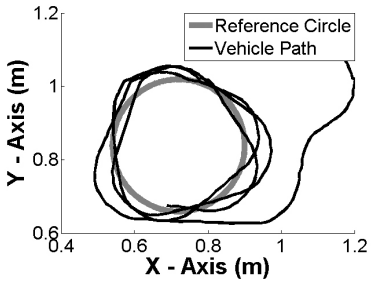

Fig. 5. Measured vehicle path about a reference circle.

B. Implementation of UAV routing on testbed

Fig. 5 shows a sample circle loitering state; it has a 12% average distance error from the reference circle. Since our vehicles have only three available forward or backward input states (left, right, straight) a hybrid control [14] method makes sense for implementing both the loitering patterns and straight line motion to targets. In our implementation of a point-to-point controller, we deviate from [14] in that when the heading error is within 10° of the reference angle we use a straight line motion. This less aggressive control law yields a smoother path to the target on our testbed vehicle. We also implement a circular path tracking using the distance error to the desire radius. We select the input that would minimizes the distance error when applied to the vehicle model. We use heuristic and some switching logic to track the circle. By being less aggressive in moving the wheel from side to side, we gain a smoother path along a polygon inside the desired circle. Fig. 5 shows a sample circle loitering state; it has a 12% average distance error from the reference circle. Fig. 6 (top) shows a complete flowchart for the algorithm.

We now present data from the testbed running experiments of the algorithm discussed above. Initially, vehicles are placed at any point and they automatically drive toward their respective loitering state. Once a target appears each vehicles checks whether the target lies within its patrol domain and the vehicle responsible for the target performs a point-to-point maneuver to approach the target. Once the vehicle reaches the target, it returns to its designated loitering state. We generate targets randomly within the testbed arena (using a uniform distribution). In order to guarantee the light load case, targets generated are put into queue only after the vehicle returns to its loitering state. In practice our target generation rate is less than $\frac{1}{6}Hz$ and the average service rate is smaller than 6 seconds (see Fig. 6). Due to the limited arena size, the loitering circle has radius 1.5 times the turning radius of the vehicle and the corresponding γ is 5.4. We use these values when computing bounds for the theory for comparison with the testbed. To demonstrate that the performance is within the theoretical upper and lower bounds (resp. (4) and (3)), we perform the experiment using 1, 2, and 3 vehicles. In each continuous run, we sequentially generate approximately 200 targets. Less than 10% of targets serviced involve a vehicle approach that either requires more

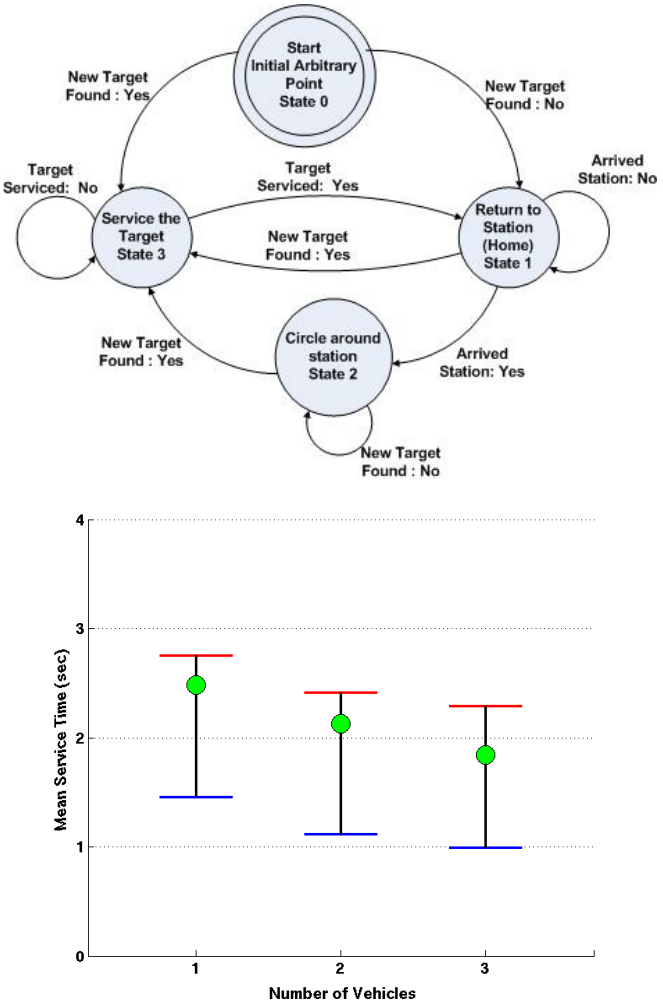

Fig. 6. (top) Vehicle control state diagram. (bottom) Average service time of randomly generated targets in the light load case. Data from three experiments with respectively one vehicle, two vehicles, and three vehicles. The lower and upper bars denote theoretical bounds (3) and (4).

than two passes to reach the target or involves out of bounds motion of the vehicle. We exclude these multi-pass (greater than 2) values from the data collected and compute the arithmetic mean of the service time for the remaining targets. As is expected, the average time decreases as the number of vehicles increases and lies within the bounds of the theory described in the previous section (see Fig. 6). Fig. 4 shows the loitering regions and loitering circles used in the experiment. A brief videoclip demonstrating the routing algorithm with human generated targets can be found online [15].

V. Conclusions and Future Work

The aim of this testbed is to address the cost and space constraints typical to many research groups. By using many off-the-shelf items, open-source software and the C programming language we were able to construct a fully operational testbed in twelve weeks with three full time students (two masters students and one undergraduate). Total material cost was under $4,000 including the vision and the controls

computers. By using small 1/64th scale micro-cars we can increase the vehicle number without requiring a prohibitively large testbed arena. In this paper we experimentally validate recently proposed algorithm for cooperative UAV routing. Our simple testbed vehicles possess the same kind of non-holonomic constraints found in kinematic UAV models, such as finite turning radius and constant speed, and thus serve as a feasible and economical model for testing cooperative control algorithms. Our testbed results, with up to three vehicles running simultaneously, lie within the theoretical bounds for the UAV routing algorithm.

In the future, we plan to build a second generation of vehicles that will have limited on board computing. The new vehicles will have a broader range of motion (variable turn angle and speed) and two-way communication. We plan to use this testbed to study a variety of different cooperative control strategies.

VI. Acknowledgments

We thank J. Enright for helpful comments. This research is supported by ARO grant W911NF-05-1-0112, ONR grant N000140410054, NSF grant ACI-0321917, and the Air Force Office of Scientific Research.

References

[1] T. Chung, L. Cremean, W.B. Dunbar, Z. Jin, E. Klavins, D. Moore, A. Tiwari, D. van Gogh, and S. Waydo, *A platform for cooperative and coordinated control of multiple vehicles: The Caltech Multi-Vehicle Wireless Testbed*, Proc. of the 3rd Conference on Cooperative Control and Optimization, Dec. 2002.

[2] Z. Jin and S. Waydo and E. B. Wildanger and M. Lammers, H. Scholze and P. Foley and D. Held and R. M. Murray, *MVWT-II: The Second Generation Caltech Multi-vehicle Wireless Testbed*, Proc. of the 2004 American Control Conference, pp. 5321-5326, 2004.

[3] R. DAndrea, *Robot soccer: A platform for systems engineering*, Computers in Education Journal, 10(1):5761, 2000.

[4] T. W. Mclain and R. W. Beard, *Unmanned Air Vehicle Testbed for Cooperative Control Experiments*, Proc. of the 2004 American Control Conference, pp. 5327-5331, 2003.

[5] The Minnow Project at Carnegie Mellon University, http://www.cs.cmu.edu/∼coral/minnow/.

[6] Robotic Embedded Systems Laboratory, Univ. of Southern California (http://robotics.usc.edu/∼embedded/research/videos.html).

[7] A. Stubbs and G. E. Dullerud, *Networked control of distributed systems: a testbed*, Proc. 2001 ASME International Mechanical Engineering Congress & Exposition, New York, New York, 2001.

[8] http://www.intel.com/technology/computing/opencv/index.htm

[9] http://msdn.microsoft.com/directx

[10] L.E. Dubins. *On Curves of Minimal Length with a constraint on average curvature and with prescribed initial and terminal positions and tangents.*American Journal of Mathematics 79, 497-516, 1957.

[11] J.J. Enright and E. Frazzoli and K. Savla and F. Bullo. *On Multiple UAV Routing with Stochastic Targets: Performance Bounds and Algorithms.* Proc. of the AIAA Conf. on Guidance, Navigation, and Control,August 2005.

[12] P. K. Agarwal and M. Sharir. *Efficient algorithms for geometric optimization.*ACM Computing Surveys, 30(4):412-458, 1998.

[13] Z. Drezner, editor. *Facility Location: A Survey of Applications and Methods.* Springer Series in Operations Research. Springer Verlag, New York 1995

[14] A. Balluchi, P. Soures and A. Bicchi, *Hybrid Feedback Control for Path Tracking by a Bounded-Curvature Vehicle*, Proc. 4th International Workshop on Hybrid Systems: Computation and Control. pp. 133-146, 2001.

[15] http://www.math.ucla.edu/∼bertozzi/target_servicing.avi