Trajectory Tracking Control of a Car-Trailer System

Adam W. Divelbiss and John T. Wen, *Senior Member, IEEE*

***Abstract*—This paper presents the experimental results of the tracking control of a car-trailer system. The proposed scheme involves three steps: 1) generate a path off-line using a path space iterative algorithm; 2) linearize the kinematic model about a trajectory which is constructed using the path; and 3) apply a time-varying linear quadratic regulator to track the trajectory. Experiments presented include parallel parking a car, docking a tractor-trailer vehicle, and parallel parking a double tractor-trailer vehicle.**

***Index Terms*—Mobile robots, nonholonomic motion planning, obstacle avoidance, path tracking, vehicle control.**

I. Introduction

MECHANICAL systems are nonholonomic if they are subject to inequality conditions on the generalized velocity which are not integrable to equivalent conditions on the generalized coordinate. For such systems, the instantaneous velocity is restricted to lie in certain directions which may be functions of time and/or configuration. Nonholonomic conditions arise in wheeled vehicles under the no-slip constraint and two rigid bodies in rolling contact, and also in systems in free fall or in space, where the total angular momentum is conserved. Examples of nonholonomic systems include mobile robots, automobiles, tractor-trailer vehicles, finger contacts in robot hand grasping, orbiting satellites, space-based robot manipulators, and even falling cats.

From the control engineering point of view, the main problems associated with a nonholonomic system are stabilization, path planning, and path following. Good summaries of recent research in these areas can be found in [1] and [2]. This paper addresses the trajectory tracking problem: control a nonholonomic system to track a preplanned path. In particular, we will focus on the implementation of a trajectory tracking controller on an experimental car-trailer system, called the *CAT*mobile, in our laboratory.

The overall approach consists of three steps (see Fig. 1).

1) *Path Planning.* Generate a path off-line connecting the desired initial and final configurations by using the nonholonomic path planner that we have developed [3].
2) *Trajectory Generation.* Impose a velocity profile to convert the path to a trajectory.

Manuscript received May 2, 1994. Recommended by Associate Editor, J. Winkelmann. This work was supported in part by the National Science Foundation under Grant IRI-9 408 874, and the New York State Center for Advanced Technology (CAT) in Automation, Robotics, and Manufacturing at Rensselaer Polytechnic Institute. The CAT is partially funded by a block grant from the New York State Science and Technology Foundation.

A. W. Divelbiss is with Reveo Corporation, Hawthorne, NY 10532 USA.

J. T.-Y. Wen is with ECSE Department, Renssalear Polytechnic Institute, Troy, NY 12180 USA.

Publisher Item Identifier S 1063-6536(97)03269-7.

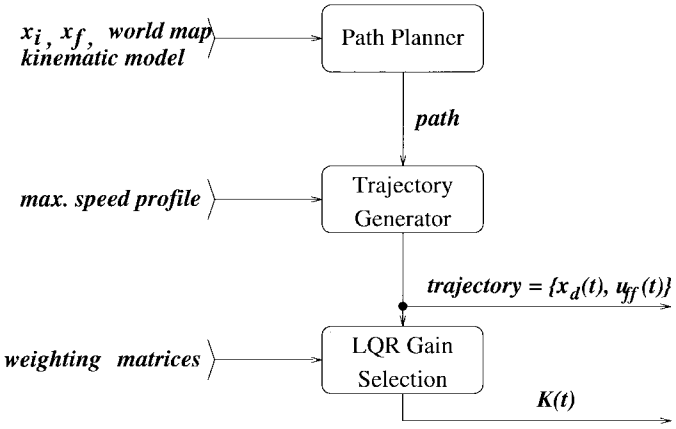


Fig. 1. Trajectory tracking method for the *CAT*mobile.

3) *Trajectory Tracking*. Apply a real-time tracking controller to follow the trajectory.

There have been many path planning algorithms proposed in the literature. We have chosen the iterative method developed in [3] due to its ability to handle constraints.

For the tracking controller, we note that nonholonomic systems are frequently controllable when they are linearized about a nonstationary trajectory [4]. Because of this fact, a locally stabilizing time-varying controller can be applied to follow a given trajectory. We have chosen to use a time-varying linear quadratic regulator (LQR) as the local tracking controller. The LQR gains are computed using the method presented in [5, pp. 52–53]. The full state feedback required by the LQR controller is replaced by an estimate obtained using road-wheel and steering encoder data in a dead-reckoning scheme.

The remainder of this paper is organized as follows. Section II presents a brief review of our path-planning algorithm. A description of the hardware and system architecture of the *CAT*mobile mobile robot is presented in Section III. Trajectory generation and trajectory tracking are presented in Section IV. This section also provides the details on state estimation and vehicle calibration. Finally, Section V presents the results from the trajectory tracking experiments with the *CAT*mobile.

II. Nonholonomic Path Planning

A. Path Planning Subject to Equality Constraints

A common starting point for the control theoretic approaches to the nonholonomic path-planning problem is to

pose it as a general nonlinear control problem of the form

$$\dot{x} = f(x)u; \quad x(0) = x_0 \tag{1}$$

where $x(t) \in \boldsymbol{R}^n$ is the configuration vector; $u(t) \in \boldsymbol{R}^m$ is the admissible velocity input, and $f(x)$ is a tall matrix $(n > m)$ parameterizing the permissible velocity directions. The problem is to find an input u which drives the system from the initial configuration x_0 to the desired final configuration x_f in finite time.

Our path-planning formulation is based on converting (1) to a nonlinear algebraic equation

$$x(1) = \hat{F}(u) \quad \hat{F}: L_2^m \rightarrow \boldsymbol{R}^n \tag{2}$$

where $u \triangleq \{u(t): t \in [0,1]\}$. Note that the path variable has been normalized to one. The input trajectory u can be expanded as an infinite Fourier series

$$u(t) = \sum_{i=1}^{\infty} \phi_i(t)\lambda_i \tag{3}$$

where the $\phi_i(t)$'s are the standard Fourier basis elements and λ_i is an $m \times 1$ constant vector representing the ith Fourier coefficients for the components of $u(t)$. For implementation, only the first M elements are used.

The path planning problem is now posed as a nonlinear root finding problem as follows:

$$y(\lambda) \triangleq F(\lambda) - x_f = 0. \tag{4}$$

Since the dimension of the search space (infinite for the complete Fourier basis representation) is much larger than the dimension of y, there are a large number of solutions to (4), many of which will lead to physically unrealizable paths. There are many numerical schemes for solving the root finding problem. We have chosen Newton's method

$$\lambda^{k+1} = \lambda^k - h[\nabla_\lambda F(\lambda^k)]^+ y(\lambda^k) \tag{5}$$

where $[\nabla_\lambda F(\lambda^k)]^+$ denotes the Moore–Penrose pseudoinverse of $\nabla_\lambda F(\lambda^k)$ and $h > 0$ is found to minimize $||y(\lambda^{k+1}(h))||^2$ at each k. The gradient $\nabla_\lambda F(\lambda^k)$ can be efficiently computed from the linearization of (1); a computation algorithm can be found in [7]. The convergence of the iteration depends on the gradient operator being full rank (equivalently, the corresponding linear time varying (LTV) system is controllable). There has been various theoretical results related to this issue (see e.g., [8] and [9]), but their discussion is outside the scope of this paper.

B. Path Planning Subject to Inequality Constraints

Suppose that a feasible region in *path* space is defined by a set of p inequalities $c(\underline{x}) \leq 0$ where $\leq$ is interpreted as a component-wise relationship and $\underline{x}$ denotes the entire path. This feasible region can be thought of as inequality constraints applied to the system at each point along the path. In the previous section we defined a mapping F relating the control λ to the final configuration $x(1)$. In the same manner, we define functions relating each point in $\underline{x}$ to λ as $x(t_j) = F_j(\lambda), j = 1, 2, \cdots, N$. Stacking each of these functions into a single vector function, we obtain a mapping $\mathcal{F}$ where $\underline{x} = \mathcal{F}(\lambda)$. Substituting this equation into the inequality above defines a feasible region in ℓ_2^m

$$c(\mathcal{F}(\lambda)) \leq 0. \tag{6}$$

To enforce the p inequality constraints represented by (6) we define a constraint equation corresponding to the ith constraint as

$$z_i(\lambda) = \gamma_i \sum_{j=1}^{N} g(c_i(F_j(\lambda))) \tag{7}$$

where $\gamma_i > 0, c_i$ is the ith constraint and g is a continuous scalar function with the property that g is equal to zero when $c_i \leq 0$ and is strictly increasing in c_i when $c_i > 0$. The choice for g which we have used most frequently is

$$g(c) = \begin{cases} (1 - e^{-rc})^2 & \text{if } c > 0 \\ 0 & \text{if } c \leq 0 \end{cases} \qquad r > 0. \tag{8}$$

Each constraint function z_i forces $\underline{x}$ toward the feasible region when the constraint is violated and has no effect when the constraint is satisfied. We write the composite constraint function in vector form as $z = [z_1, \cdots, z_p]^T$.

Using this formulation the motion-planning problem becomes: *Find* λ *such that* $y(\lambda) = 0$ *and* $z(\lambda) = 0$. The same Newton step approach presented in the previous section is now applied to the composite constraint vector

$$\psi(\lambda) = \begin{bmatrix} y(\lambda) \\ z(\lambda) \end{bmatrix}; \quad \psi(\cdot): \ell_2^m \rightarrow \boldsymbol{R}^{n+p}. \tag{9}$$

The new update law becomes

$$\lambda^{k+1} = \lambda^k - h[G(\lambda^k)]^+ \psi(\lambda^k). \tag{10}$$

where

$$G(\lambda^k) \triangleq \begin{bmatrix} \nabla_\lambda F(\lambda^k) \\ \nabla_\lambda z(\lambda^k) \end{bmatrix}. \tag{11}$$

As in the previous section, a sufficient condition for the convergence of this algorithm is that the gradient operator $G(\lambda)$ be full rank for all k. Though this cannot be guaranteed for all situations, this condition has not been a problem for the experiments considered in this paper.

III. The *CAT*Mobile Mobile Robot

A. Hardware Description

The mobile robot used for the experiments in this paper is known as the *CAT*mobile. The *CAT*mobile is a one-quarter-scale radio-controlled model race car which has been converted for use in an indoor laboratory. The purpose of this versatile mobile robot is to provide a platform for use in a variety of autonomous wheeled vehicle and tractor-trailer experiments. The *CAT*mobile consists of a one-quarter-scale grand national stock car chassis, a passive suspension system, an on-board 33-MHz 486 computer, and road wheel and steering encoders for sensing the car's position and orientation. There are also two semitrailers available for use in single

Fig. 2. *CAT*mobile mobile robot.

and multiple trailer experiments. The *CAT*mobile can be run in teleoperated mode through a keyboard or a four-channel radio system, or it can be set to autonomous mode for use in nonholonomic path tracking experiments.

The original car kit comes with a gasoline engine and is controlled remotely by a two-channel radio. The current car is equipped with a passive suspension system complete with shock absorbers and antisway bars. Other mechanical features include a disk brake, a simple differential for the rear axle, and adjustable front and rear end alignment rods. The frame is constructed of welded steel tubing and the tires provided with the kit are wide racing slicks. Since the car chassis was originally designed for racing on oval or grand prix style tracks the car has severe limitations on the steering angle. Currently hard stops on the steering permit an angular range of only $\pm 15^{\circ}$ from center. This value is at least half that of a full size car or truck. The car is 48 in long, 18.5 in wide, and 12 in in height. The vehicle testbed also includes two adjustable length trailers for use in single and multiple trailer experiments. These trailers are constructed in the shape of a standard boat trailer and have rear wheels on separate axles at the rear of the trailer. Nominally the trailers are 39 in long and 22.5 in wide. Their length is adjustable by a total of about 6 in. Fig. 2 is a picture of the *CAT*mobile mobile robot.

The *CAT*mobile is driven by a 48-W permanent magnet dc pancake motor located at the rear of the vehicle. This motor was chosen mainly because of its thinness and good torque constant. Mechanical power is transmitted directly to the rear axle by a simple chain and sprocket system. Several gear ratios are available for performing low-speed high torque or high-speed experiments with the car. The dc motor is supplied by a 12-V nicad battery pack which under nominal usage will provide power for several days to a week. The steering mechanism of the car is driven by a standard Futaba high torque servomotor. This motor is sold with an embedded position control system. However, for application in our control scheme, the position control circuitry has been removed and the motor is controlled directly by the on-board computer. Both the steering and the drive motor use the same battery supply and are controlled using pulse width modulation (PWM). PWM was selected for its simplicity and low cost: there is no need for digital to analog conversion, and digital pulses can be generated by an inexpensive counter/timer card. An Omega PCL-830 counter/timer board is used to interface the computer with both drive and steering motors. PWM amplifiers using a standard H-bridge design were built in-house to convert the digital control pulses sent by the computer into the proper input current for the motor.

Sensors currently in use on the car include optical encoders mounted to three of the four road wheels, an optical encoder mounted to the output shaft of the steering motor, and an optical encoder to measure each trailer jackknife angle. The code wheels used for the road wheel and steering encoders have 1000 counts each. Jackknife angle code wheels have 500 counts each. Each of these seven encoders is interfaced to the on-board computer by an Omega PC-CTR-20 counter timer card equipped with a four times decoding circuit which can service up to eight encoders. Four times decoding of the encoder signals increases the counts to 4000 for the road wheel and steering sensors and 2000 for the jackknife angle sensors. These encoders are used to determine the relative position and orientation of the car and trailers using dead-reckoning.

A 33-MHz 486 computer is mounted inside the car. This computer is used to perform off-line motion planning as well as for run time control. The car is also equipped with a radio receiver and translator circuit to interface the radio to the computer. This circuitry provides the capability to use a standard four-channel radio transmitter to control the car remotely.

B. System Architecture

The operating software for the *CAT*mobile has been written in C++ and currently provides four modes of operation. The *teleoperation* mode allows the user to control the car via a four-channel radio system. The *autonomous* mode causes the car to track a preplanned trajectory. The *calibration* mode is used to determine the initial steering angle of the car. A special *test* mode allows the user to specify an input function to test the drive and steering motor controllers. In each of the four operating modes, a two-level control hierarchy is used to control the motion of the car. The lower level consists of two separate control systems, one for the drive motor and one for the steering motor. The upper level provides steering and driving commands to the lower level and is different for each operating mode. Fig. 3 shows a block diagram of the control system for the autonomous mode.

The lower level in the control hierarchy provides position control for both the drive motor and the steering motor as shown on the right-hand side of Fig. 3. Position control for each motor is provided by a corresponding PID controller. An optical encoder mounted to the left back wheel provides position feedback for the drive motor and an optical encoder mounted to the output shaft of the steering motor provides steering angle feedback. Each motor controller was tuned individually using the test mode. The lower level is the same for each of the operating modes. In autonomous mode, the upper level of the control hierarchy serves as the trajectory tracking controller. As will be seen in the next section, this controller is implemented using an LQR controller. The

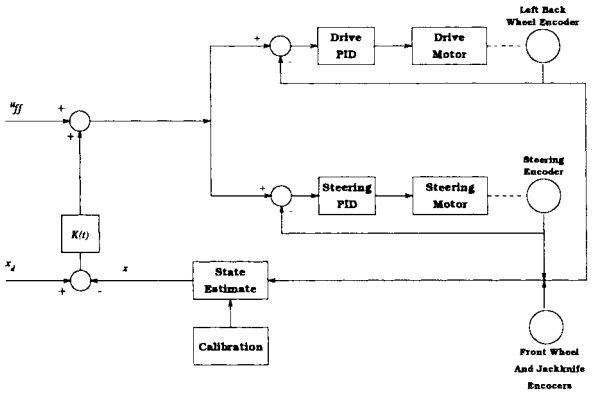


Fig. 3. *CAT*mobile autonomous mode system architecture.

left side of Fig. 3 shows the tracking control structure. The two input signals are the feedforward control, u_{ff}, and the desired configuration, x_d. Both signals are provided by a trajectory generator. The time-varying gain matrix $K(t)$, is the precomputed LQR gain matrix used by the tracking controller. In teleoperated mode, motor position inputs are provided by the radio receiver. In the calibration and test modes, the position inputs are provided by a specified test pattern.

The code structure used to implement this two-level control scheme for the car in autonomous mode consists of a main routine and an interrupt service routine. The main routine is responsible for sending steering and drive motor position setpoint commands to the interrupt service routine via shared memory at time intervals specified in a trajectory file. The set points are generated based on the output of the tracking controller. Vehicle configuration is read from shared memory and used as control feedback for the tracking controller. The main routine also checks the standard input stream for user inputs such as emergency stop. The purpose of the interrupt service routine is to collect the sensor data, determine the current system configuration using dead-reckoning, and to implement the two low-level PID motor controllers. The sampling rate for these controllers is determined by the frequency at which the interrupt service routine is called. A counter on one of the counter/timer boards is used to generate a hardware interrupt, which invokes the service routine, every 5 ms. This sample rate is defined by the operating system software and can be set to other values.

IV. Trajectory Tracking with the *CAT*mobile

A. Tracking Controller

Given a nonholonomic path, $x^{(0)}$ and the corresponding input $u^{(0)}$, the nonholonomic system (1) linearized about this path is given by

$$\delta\dot{x}(t) = A(t)\delta x + B(t)\delta u, \quad \delta x(0) = 0 \tag{12}$$

where

$$A(t) \triangleq \left[\left.\frac{\partial f}{\partial x^1}\right|_{x^{(0)}(t)} u^{(0)}(t) \cdots \left.\frac{\partial f}{\partial x^n}\right|_{x^{(0)}(t)} u^{(0)}(t) \right]$$
$$B(t) \triangleq f(x^{(0)}(t)).$$

Assume that this LTV system is controllable (as stated before, this is equivalent to ∇F being full rank). There are many methods for its control. We have chosen to use a time-varying LQR as was suggested in [10]. For implementation on a digital computer, the system equation is discretized as

$$\delta x_{k+1} = A_k \delta x_k + B_k \delta u_k, \quad \delta x_0 = 0 \tag{13}$$

where

$$A_k \triangleq e^{A(t_k)h} \quad B_k \triangleq \int_{t_k}^{t_{k+1}} e^{A(t_k)(t_{k+1}-\tau)}\, d\tau B(t_k) \tag{14}$$

and where h is the step size and t_k is the time at the kth step. The performance index to be optimized is

$$J = \tfrac{1}{2} x_N^T P_N x_N + \tfrac{1}{2} \sum_{k=0}^{N-1} (x_k^T Q_k x_k + u_k^T R_k u_k). \tag{15}$$

The optimal feedback control gains are computed backward in time according to the following recursive equations [5]:

$$K_k = [B_k^T P_{k+1} B_k + R_k]^{-1} B_k^T P_{k+1} A_k$$
$$P_k = A_k^T P_{k+1}[A_k - B_k K_k] + Q_k$$

where $P_N \geq 0$, is the end configuration weighting matrix, $Q_k \geq 0$ is the configuration weighting matrix, and $R_k > 0$ is the control weighting matrix. The P_N, Q, and R matrices are chosen to be time-invariant and diagonal. Determining P_N, Q, and R is no trivial task. Choosing these control gains for vehicle experiments presented in this paper took literally days of trial and error tuning. The gains chosen seem to work well for the particular path examples shown, but since the LTV system is path-dependent, radically different paths may require different gains. There is great need for future research in this area.

Finally, the control law for the LTV system is defined as

$$\delta u_k = -K_k \delta x_k. \tag{16}$$

Written in terms of the original control, we have

$$u = u_{\text{ff}} - K_k(x - x_d) \tag{17}$$

where u_{ff} and x_d are determined by the trajectory generator.

B. Trajectory Generation

Having described the method to be used for tracking a trajectory, we now address the problem of generating a time profile specifying the speed at which the path is to be followed. Let $\dot{x}_k$ denote the *path velocity* (i.e., velocity of the time-normalized path) at time t_k. We need to generate the true desired velocity $\dot{x}_{d_k}$. Suppose the maximum allowable speed for the vehicle is S_k and the maximum path speed $||\dot{x}_k||$ is S_m. The desired velocity can be chosen as the scaled version of $\dot{x}_k$

$$\dot{x}_{d_k} = \frac{S_k}{S_M} \dot{x}_k. \tag{18}$$

This can be further converted to a desired time stamp for the kth configuration on the path x_k

$$t_{k+1} = t_k + \frac{S_M}{S_k} \frac{1}{N-1} \qquad t_0 = 0. \tag{19}$$

As we scale the velocity, the path control also needs to be scaled correspondingly. Let u_k be the path control at t_k, i.e., $\dot{x}_k = f(x_k)u_k$, or, equivalently, $u_k = f^+(x_k)\dot{x}_k$. The feedforward control to be used in real time control is then (17) is computed as

$$u_{\mathrm{ff}k} = f^+(x_k)\dot{x}_{d_k}. \tag{20}$$

To summarize, trajectory generation involves the following steps.

1) Read into memory a time-normalized path generated by the nonholonomic path planner.
2) Use interpolation to generate a path with the desired number of trajectory set points.
3) Translate the path so that the starting configuration is zero.
4) Scale the normalized x- and y-positions of the path to real-world coordinates. Paths are normally planned with the linear position states normalized to one wheel base of the car (26.5 in).
5) Using the forward difference method, determine the velocity profile of the time normalized path.
6) Redistribute the path points in time to achieve a specified velocity profile. The velocity profile currently used consists of a ramp up to a maximum velocity at the beginning of the path, followed by a ramp back down to a minimum velocity at the end of the path.
7) Interpolate the resulting trajectory to evenly distribute the path set points at a regular time interval.
8) Calculate the feedforward control and LQR gains based on the resulting trajectory.

The final output of the trajectory generator is a file containing the trajectory that includes the time profile and feedforward control as well as a file containing the LQR gains.

C. State Estimation Using Dead Reckoning

The configuration vector of the vehicle is determined by using a dead-reckoning scheme which is based on an idealized kinematic model of the vehicle. This idealized model is unrealistic as it assumes point contact of the wheels, ignores the effects of friction and stiction, and ignores backlash and hysteresis in the chain drive. Nevertheless, with feedback, we have found this model adequate for our experiments. The kinematic model for the car alone is

$$\begin{aligned}\dot{x}_1 &= \cos(x_4)u_1\\ \dot{x}_2 &= \sin(x_4)u_1\\ \dot{x}_3 &= u_2\\ \dot{x}_4 &= \frac{1}{l_1}\tan(x_3)u_1\end{aligned} \tag{21}$$

where l_1 is the wheel base length of the car, u_1 is the driving velocity, u_2 is the steering velocity, (x_1, x_2) are the (x, y)-position of the center of the rear axle, x_3 is the steering angle, and x_4 is the car orientation. Note that this model is different than the usual front-wheel drive car model in which x_1, x_2, and x_4 equations are multiplied by a $\cos(x_3)$ term.

In the dead-reckoning scheme, u_1 and u_2 represent velocities measured using the road wheel and steering encoders. Since there are optical encoders placed on the road wheels and on the steering servo output shaft, it is possible to measure these velocities by taking the difference between encoder counts from successive sample periods. Suppose $\theta(k)$ represents the encoder count of the left rear road wheel, and $\phi(k)$ represents the encoder count of the steering servo shaft at discrete time k. Then using the forward difference approximation of the derivative in the kinematic equation above, we can write the system configuration at discrete time k in terms of encoder counts as

$$\begin{aligned}\hat{x}_1(k) &= \hat{x}_1(k-1) + \cos(\hat{x}_4(k-1))K_{\mathrm{drv}}\Delta\theta\\ \hat{x}_2(k) &= \hat{x}_2(k-1) + \sin(\hat{x}_4(k-1))K_{\mathrm{drv}}\Delta\theta\\ \hat{x}_3(k) &= K_{\mathrm{str}}\phi(k)\\ \hat{x}_4(k) &= \hat{x}_4(k-1) + \frac{1}{l_1}\tan(\hat{x}_3(k-1))K_{\mathrm{drv}}\Delta\theta\end{aligned}$$

where K_{drv} is the drive constant in units of inches per drive count, K_{str} is the steering constant in units of radians per steering count, and $\Delta\theta = \theta(k) - \theta(k-1)$ is the change in drive counts measured between successive sample periods. These equations give us the approximate configuration vector of the vehicle $\hat{x}$, which is used by the LQR tracking control previously described. In the case of car-trailers, the trailer jackknife angles can be directly measured and used in the state vector.

D. Calibration

To conclude this section on nonholonomic trajectory tracking, we briefly discuss some of the issues related to calibration of the *CAT*mobile prior to running a trajectory tracking experiment. In order to track the desired trajectory, the vehicle must be placed in the proper position on the laboratory floor and the initial configuration must be known. Since the current *CAT*mobile has neither proximity sensors nor a run-time obstacle avoidance scheme, initial errors in placement and configuration knowledge may cause the vehicle to collide with obstacles it would otherwise have avoided. Currently, the initial calibration is done by alignment of calibration marks, except in the case of determining the front wheel steering angle. Since there are encoders on both front road wheels, it is possible to determine the absolute steering angle independently of the steering motor encoder.

The calibration method for determining the front wheel steering angle is based on measuring the distance traveled by each front wheel and then calculating the steering angle of an equivalent virtual front wheel centered between the two actual front wheels. Fig. 4 shows the geometry of the car along with the rolling compatibility condition required for pure rolling contact of the vehicle [11]. In this figure, $x_{3\ell}$ is the steering angle of the left front wheel, x_{3r} is the steering angle of the right front wheel, and x_3 is the steering angle of the virtual wheel. The steering angle for the virtual wheel is the same as the steering angle defined in the kinematic model of (22). For each wheel of the car to be in pure rolling contact with the ground, it is required that the lines extending along each wheel axis intersect at the same point as shown in the figure.

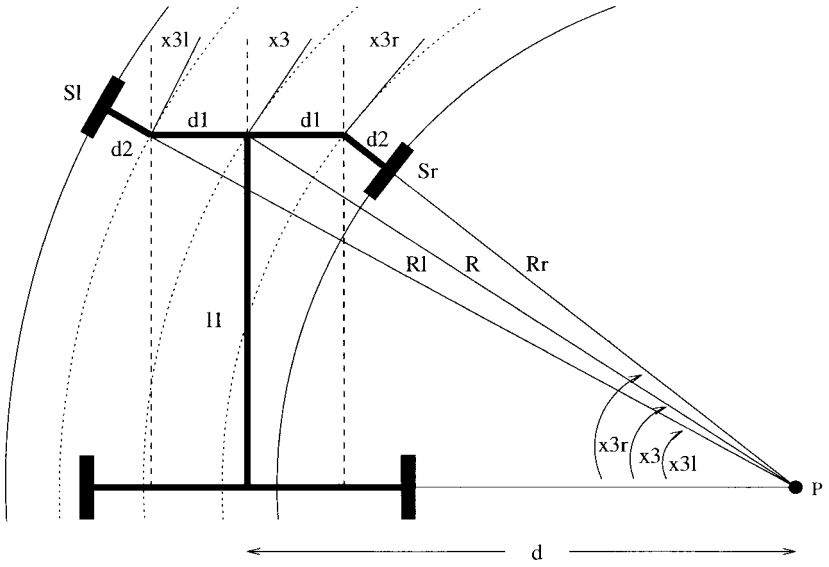


Fig. 4. *CAT*mobile calibration geometry.

Other values shown in the figure are the car wheel base length l_1 and the front wheel lateral offset d_2.

To perform the steering calibration, the car is driven backward and forward in a sinusoidal fashion for a short period of time while the front wheel steering angle is held constant (and as close to zero as possible). The distance traveled by both front wheels during the time interval is measured by the front road wheel encoders. Since the steering angle is fixed, the front wheels each trace an arc of a circle on the ground. Thus the distance measured by the road wheel encoders represents the arc length traveled along the circle. The radii for the circles traced out by both front wheels, the virtual front wheel, and the center of the rear axle are shown in Fig. 4 as R_l, R_r, R, and d, respectively. Under the no-slip condition with the steering angle fixed, it is possible to derive a relationship between the ratio of the arc lengths traveled by each front wheel and the steering angle of the virtual wheel. Let S_l be the distance traveled by the left front wheel and S_r be the distance traveled by the right front wheel. The ratio S_l/S_r can be written as a function of the virtual steering angle x_3 as

$$\frac{S_l}{S_r} = \frac{(l_1 - \operatorname{sgn}(x_3)\, d_2 \sin|x_{3\ell}|)\sin|x_{3r}|}{(l_1 + \operatorname{sgn}(x_3)\, d_2 \sin|x_{3r}|)\sin|x_{3\ell}|} \tag{22}$$

where

$$x_{3\ell} = \operatorname{sgn}(x_3)\tan^{-1}\left(\frac{l_1 \sin|x_3|}{l_1\cos|x_3| - \operatorname{sgn}(x_3)\, d_1 \sin|x_3|}\right) \tag{23}$$

$$x_{3r} = \operatorname{sgn}(x_3)\tan^{-1}\left(\frac{l_1 \sin|x_3|}{l_1\cos|x_3| + \operatorname{sgn}(x_3)\, d_1 \sin|x_3|}\right). \tag{24}$$

The front wheel axle length d_1 is 5.9375 in, and the lateral offset d_2 is 1.75 in. The derivation of this equation can be found in [6]. Fig. 5 shows a plot of this function for the virtual steering angle $x_3 \in [-\pi/2, \pi/2]$. Close observation of this plot shows that the function is not quite antisymmetric about zero as it appears on first glance. This condition is due to the fact that $x_{3\ell}$ always leads x_3 while x_{3r} always lags. Since it is desired to find the steering angle x_3, the inverse to the above equation must be found. From the plot, it is clear that the inverse exist for a range of steering angles $[-a, a]$ where $0 < a < \approx 50°$. A lookup table is constructed to determine x_3 from a given S_l/S_r. Since the steering angle of the *CAT*mobile is limited to about $\pm 15°$ from center, the lookup table need only contain values on this interval.

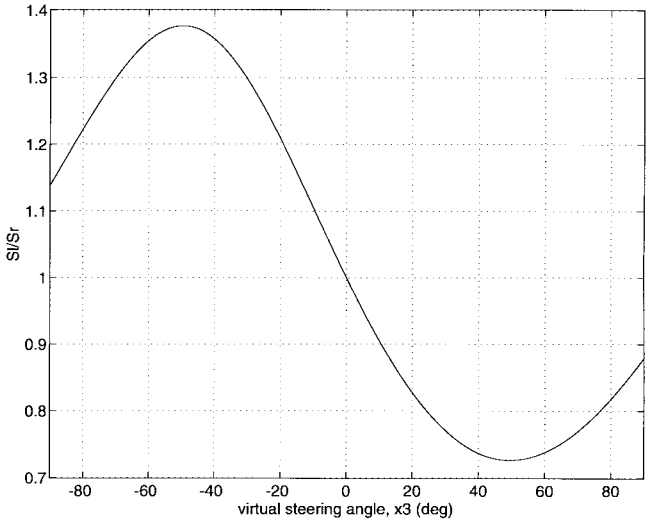


Fig. 5. *CAT*mobile steering calibration function.

To summarize, calibration of the front wheel steering angle is performed by driving the car backward and forward in a sinusoidal fashion and measuring the total distance traveled by both front wheels to obtain the ratio S_l/S_r. The lookup table representing the inverse of (22) on the interval $[-15°, 15°]$ is then used to compute the virtual steering angle x_3. Since nonholonomic systems are driftless, and all else being equal, the vehicle ends at the same position from which it started when the calibration is completed.

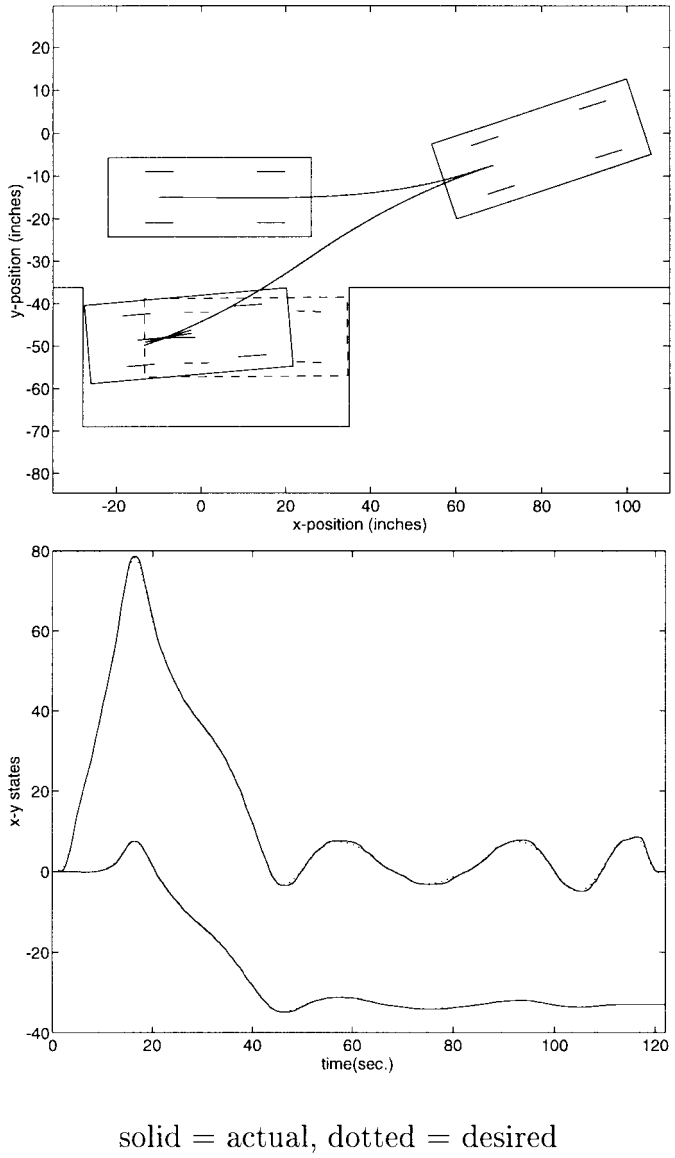


Fig. 6. *CAT*mobile parallel parking: actual and desired paths.

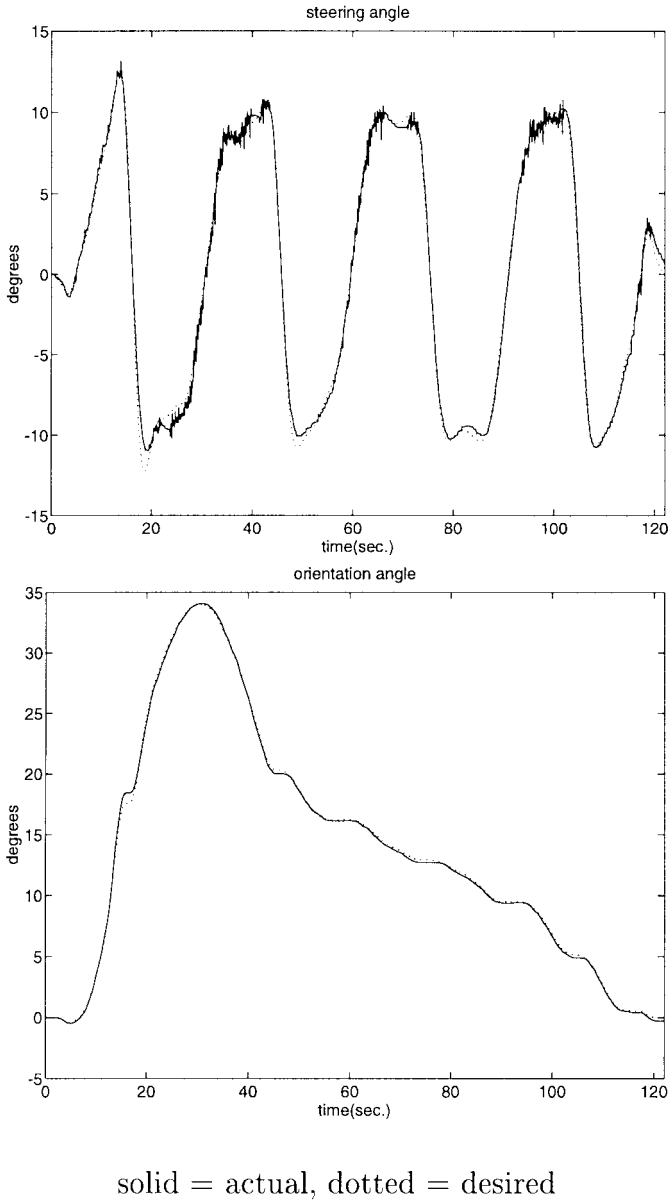


Fig. 7. *CAT*mobile parallel parking: steering and orientation angles.

V. Experiments

A. The CATmobile with No Trailers

In the first experiment the car with no trailers attached is required to parallel park itself between two boxes placed 63 in apart. Since the car is 48 in long, maximum clearance is only 7.5 in (2.5 ft scaled). To perform this task, the path planner was run to obtain a feasible collision free path where the initial configuration is $(-10 \text{ in }, -15 \text{ in}, 0^\circ, 0^\circ)$ and the final configuration is $(-10 \text{ in}, -48 \text{ in}, 0^\circ, 0^\circ)$. Limits on the steering angle were set at $\pm 12^\circ$. Using the first 51 terms of the Fourier series to represent the control and an initial guess path in which the car dives forward, backward and then forward again, the path planner took 135 iterations and a total time of 10.28 min on a Sparc 10/51 to determine the path. The large iteration and solution time values are a reflection of the difficulty of planning paths in highly constrained environments. In fact, by relaxing the steering limits to $\pm 35^\circ$, which is approximately the steering limits of an actual car, the algorithm took only 15 iterations and 62 s to solve the exact same problem; a reduction of nearly one order of magnitude for both iteration and solution time.

Finally a trajectory was formed using the trajectory generation algorithm described earlier and the trajectory tracking experiment was run. The results of this experiment are shown in Figs. 6 and 7. Fig. 6 shows plots of the desired path (dotted) and the actual path (solid) taken by the car. In the left-hand plot the car starts at the top and proceeds into the parking space. The three extreme points of the path are marked by scale drawings of the actual car. The right-hand plot shows the x- and y-positions of the car with time, where the dotted lines are the desired and the solid lines are the measured values. Fig. 7 shows actual and desired steering angles in the left-hand plot, and actual and desired orientation angles in the right-hand plot. It can be seen from each of these plots there are only slight

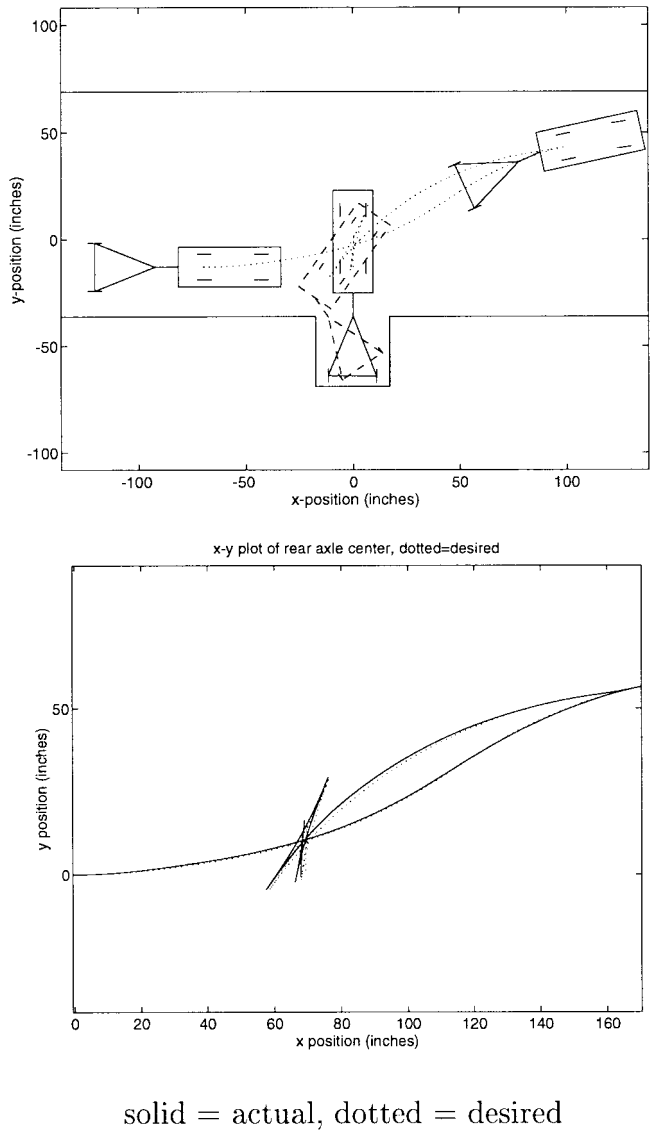


Fig. 8. Tractor-trailer docking: actual and desired paths.

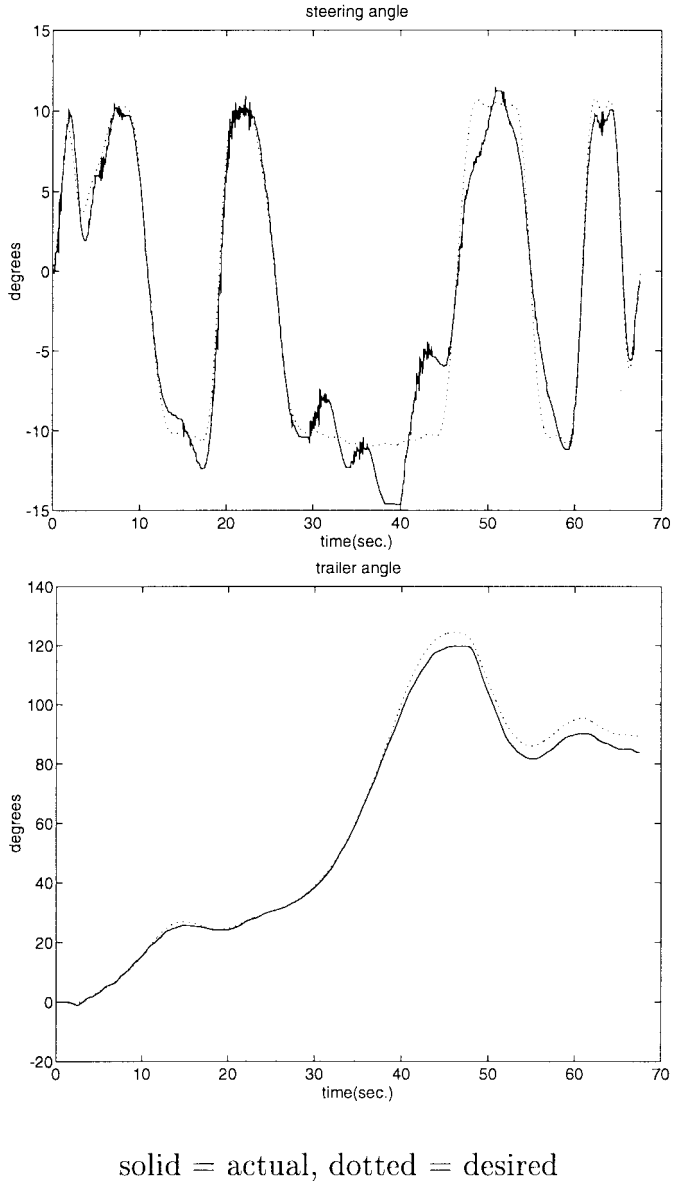


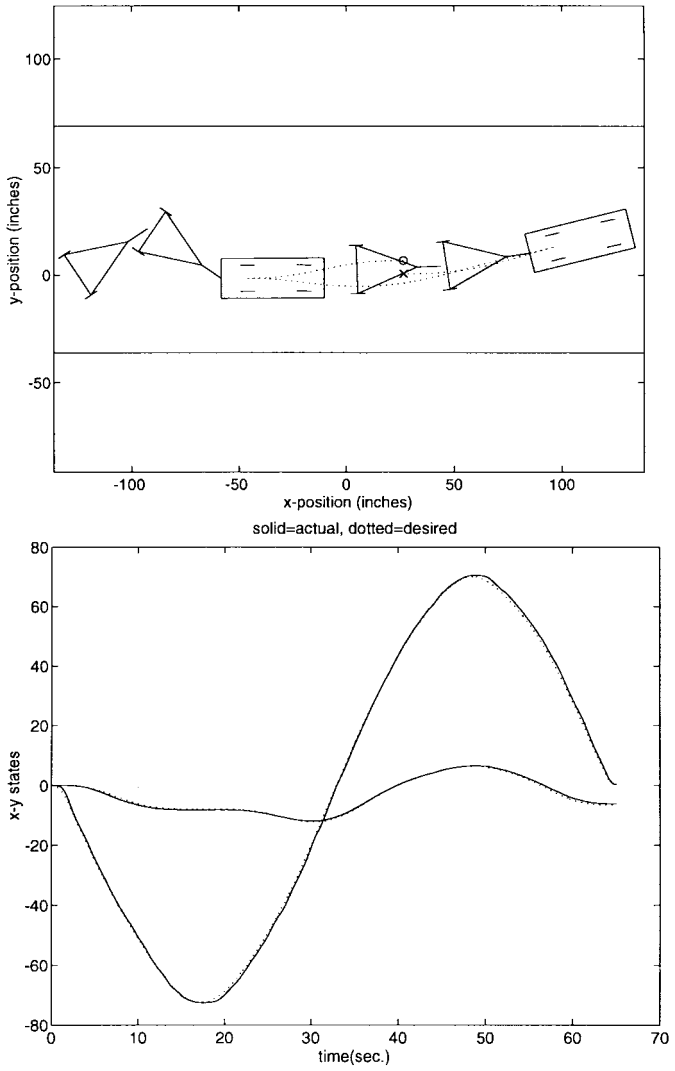
Fig. 9. Tractor-trailer docking: steering and trailer angles.

errors in tracking the desired command signal. Considering the fact that the kinematic model used to represent the car is highly idealized (i.e., zero friction, point contact of wheels, etc.), it is remarkable that the vehicle can track the path so closely.

B. The CATmobile with One Trailer

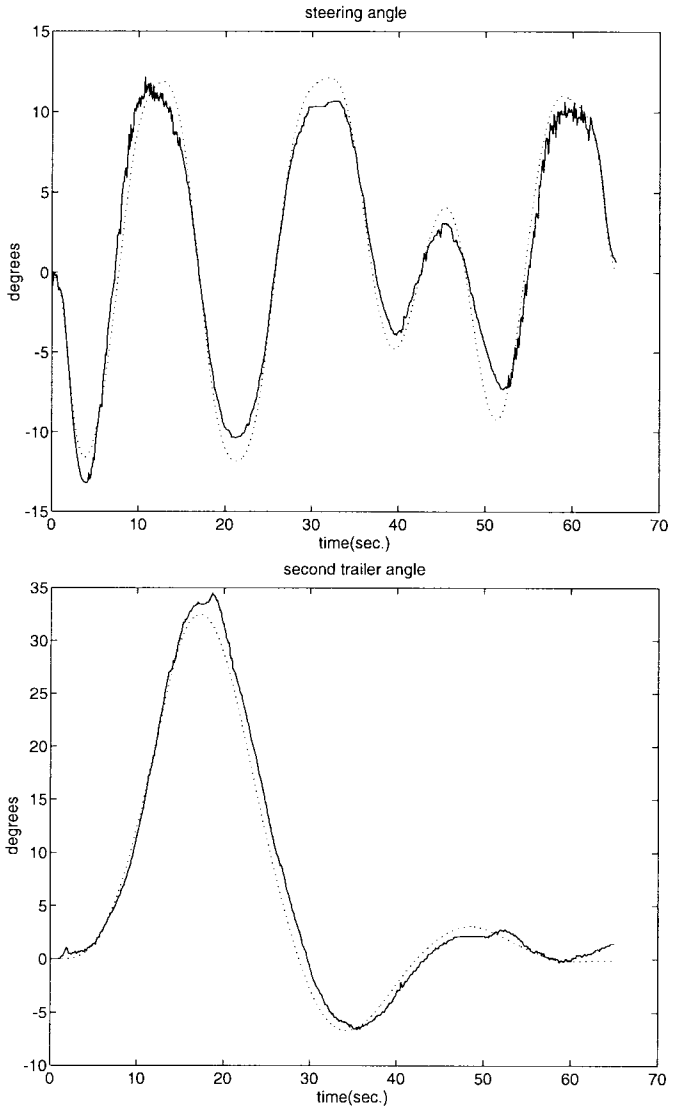
The second example involves driving the *CAT*mobile with one trailer attached to the rear bumper. In this case the driving task is to dock the trailer between two boxes which simulate a loading dock. The loading dock is 30.5 in wide and the trailer is 22.5 in wide yielding a maximum clearance of 4 in (1.33 ft scaled) on either side. Again, the path planner is first used to determine a feasible collision free path. The initial and final configurations are $(-69\text{ in}, -12.875\text{ in}, 0^\circ, 0^\circ, 0^\circ)$ and $(0\text{ in}, -12.875\text{ in}, 0^\circ, 90^\circ, 90^\circ)$, respectively. The steering limits used were $\pm 12^\circ$ and the trailer jackknife angle limits (i.e., the difference in orientation between the car and the trailer) were $\pm 70^\circ$. Using the first 51 terms of the Fourier series to describe the control, the path planning was performed in two steps. In the first step, both steering and jackknife angle constraints were enforced but the boundary constraints did not include the sides of the loading dock. From an initial guess in which the tractor-trailer was driven in a sinusoidal fashion, the planner took 131 iterations in 14.25 min to reach the solution. Using the results of the first step as the initial guess for the second step, the dock sides were added to the constraints and the planner took 64 iterations in 11.5 min to reach the final solution. As in the previous example, relaxing the steering constraints greatly decreased the number of iterations and the solution time.

To perform the experiment, the trajectory generator was again used to enforce a suitable time profile on the path. Results of the experiment are shown in Figs. 8 and 9. Fig. 8 shows an $x-y$ plot of the path traced out by the tractor-trailer. The left hand plot shows the desired path of the tractor-trailer system. Two intermediate configurations are shown where the vehicle is at the boundary of the room and where the jackknife



solid = actual, dotted = desired

Fig. 10. Double tractor-trailer parallel parking: actual and desired paths.



solid = actual, dotted = desired

Fig. 11. Double tractor-trailer parallel parking: steering and second trailer angle.

angle is at its constraint limit (70°). This figure also shows the room boundary and the loading dock. A scale drawing of the *CAT*mobile with one trailer is superimposed on the path to demonstrate the vehicle's motion. In Fig. 9, note the significant deviation of the steering angle from the nominal trajectory while the trailer jackknife angle stays fairly close. This is due in part to the need for steering input to compensate for the modeling error. Also note that the jackknife angle error grows in the backing up segment of the motion. This is not surprising, as backing up motion tends to amplify the jackknife angle while forward motion reduces it.

C. The CATmobile with Two Trailers

The final experimental example involves parallel parking the *CAT*mobile with two trailers. This task is very challenging for an average human driver. In this example, the vehicle is required to make a movement of 5.3 in to the right, starting and ending with the tractor and both trailers in alignment. Further restrictions on the vehicle include $\pm 12^\circ$ steering angle limits and $\pm 60^\circ$ jackknife angle limits. The double tractor-trailer is also required to stay entirely within the rectangular region shown in Fig. 10. Using an initial control in which the vehicle is again driven in a sinusoidal fashion, the path planner took 11 iterations in 2.5 min to reach a solution.

Applying the trajectory generator and running the tracking experiment for this path yielded the results shown in Figs. 10 and 11. In the left-hand plot of Fig. 10, the vehicle starts with the center of the rear tractor axle on the spot marked "$\circ$." It then backs up the left-most configuration, keeping the rear trailer from hitting the wall while at the same time keeping the jackknife angles within their limits. The vehicle then pulls forward to the right-hand location, and then finally backs up to the point marked "$\times$." For clarity, only two intermediate configurations are shown where the vehicle is near the boundary of the room. Again we see that the vehicle

follows the desired path quite closely and the vehicle boundary remains inside the room. The x- and y-positions and car orientation angles again track fairly well with the jackknife angles tracking a little worse than the single trailer case. As in the previous example, the steering angle significantly deviates from the preplanned trajectory in order to compensate for the modeling error.

VI. Conclusions

We have presented a trajectory tracking control strategy for an experimental tractor-trailer vehicle. Our algorithm involves three steps. First, a path satisfying the nonholonomic constraints of the vehicle is planned off-line using the method presented in Section II. This path is converted to an equivalent trajectory consisting of the path and a velocity profile. The method used for generating the trajectory involves stretching the path in time until the maximum velocity constraint is satisfied. Once the trajectory is determined, a feedforward control is computed. For the actual experiment, this feedforward control is added to the output of a time-varying LQR controller to track the desired trajectory.

Experimental results from the application of this tracking control scheme to the *CAT*mobile with no, one, and two trailers are presented. In each experiment, the robot car is required to perform some challenging driving maneuvers to demonstrate the performance of the trajectory tracking algorithm. In all cases, the algorithm has been shown to track the desired trajectory in a satisfactory manner. Due to the very tight steering constraints, the path planning algorithm can take several minutes to reach a solution.

Since dead-reckoning is used for state estimation in the experiments, calibration is of extreme importance. We have presented an algorithm for determining the initial steering angle based on the ratio of the distances traveled by each wheel over some time interval.

We are currently working to improve the efficiency of the planning algorithm and to develop alternate feedback controllers for trajectory tracking.

References

[1] Z. Li and J. F. Canny, Eds., *Nonholonomic Motion Planning*. Boston, MA: Kluwer, 1993.
[2] S. S. Sastry, R. M. Murray, and Z. Li, *A Mathematical Introduction to Robotic Manipulation*. Boca Raton, FL: CRC, 1993.
[3] A. W. Divelbiss and J. T. Wen, "Nonholonomic path planning with inequality constraints," in *Proc. 32nd IEEE Conf. Decision Contr.*, San Antonio, TX, Dec. 1993.
[4] Y. Kanayama, Y. Kimura, F. Miyazaki, and T. Noguchi, "A stable tracking conotrol method for an autonomous mobile robot," in *Proc. 1990 IEEE Robotics Automat. Conf.*, Cincinnati, OH, May 1990, pp. 384–389.
[5] Frank L. Lewis, *Optimal Control*. New York, Wiley, 1986.
[6] A. W. Divelbiss, "Nonholonomic motion planning in the presence of obstacles," Ph.D. dissertation, Rensselaer Polytechnic Institute, Troy, NY, Dec. 1993.
[7] A. Bryson and Y. C. Ho, *Applied Optimal Control*. Bristol, PA: Hemisphere, 1975.
[8] E. D. Sontag, "Control of systems without drift via generic loops," *IEEE Trans. Automat. Contr.*, vol. 40, pp. 1210–1219, July 1995.
[9] H. J. Sussmann, "A continuation method for nonholonomic path-finding problem," in *Proc. 32nd IEEE Conf. Decision Contr.*, San Antonio, TX, Dec. 1993, pp. 2718–2723.
[10] G. Walsh Murray, R. M. Sastry, and S. S. Sastry, "Stabilization and tracking for nonholonomic control systems using time-varying state feedback," in *Proc. IFAC Symp. Nonlinear Control System Design*, Bordeaux, France, 1992, pp. 1–6.
[11] J. Y. Wong, *Theory of Ground Vehicles*. New York: Wiley, 1978.

Adam W. Divelbiss received the B.E.E. and M.S. degrees in electrical engineering from Auburn University, AL, in 1988 and 1990, respectively, and the Ph.D. degree in electrical engineering from Rensselaer Polytechnic Institute, Troy, NY, in December 1993.

His research at the masters level involved the design and construction of a prototype optical aspect system for NASA's HEIDi solar telescope, launched by balloon in the summer of 1993. The aspect system provided azimuth and elevation pointing information for the telescope. His research at the doctoral level was conducted at the New York State Center for Advanced Technology in Automation and Robotics at Rensselaer from 1990 to 1993. His work involved the motion planning and control of nonholonomic robotic systems operating in the presence of obstacles and the development of a quarter-scale robotic car with two trailers for use as a testbed for nonholnomic experiments. Since May 1994 he has been a Member of the Technical Staff at Reveo Inc., Hawthorne, NY, where he is involved in research and development of motion base systems and simulators for the entertainment industry.

John Ting-Yung Wen (S'79–M'81–SM'93) received the B.Eng. degree from McGill University, Montreal, Canada, in 1979, the M.S. degree from the University of Illinois, Urbana, in 1981, and the Ph.D. degree from Rensselaer Polytechnic Institute, Troy, NY, in 1985, all in electrical engineering.

From 1981 to 1983, he was a System Engineer in Fisher Control Company, Marshalltown, IA, where he worked on the coordination control of a pulp and paper plant. From 1985 to 1988, he was a Member of the Technical Staff at the Jet Propulsion Laboratory, Pasadena, CA, where he worked on the modeling and control of large flexible structures and multiple-robot coordination. Since 1988, he has been with the Department of Electrical, Computer, and Systems Engineering at Rensselaer Polytechnic Institute, where he is currently a Professor. He is also a Member of the New York Center for Advanced Technology in Automation, Robotics, and Manufacturing. His current research interests are in the area of modeling and control of multibody mechanical systems, including flexible structures, robots, and vehicles, and material processing systems such as extrusion and resistive welding.