2007

Open-loop Vehicle Control Using an Abstraction of its Model

P. Mellodge

Virginia Polytechnic Institute and State University

Pushkin Kachroo

University of Nevada, Las Vegas, pushkin@unlv.edu

Follow this and additional works at: https://digitalscholarship.unlv.edu/ece_fac_articles

Repository Citation

Mellodge, P., Kachroo, P. (2007). Open-loop Vehicle Control Using an Abstraction of its Model. *American Control Conference* 3023-3028. Institute of Electrical and Electronics Engineers.
https://digitalscholarship.unlv.edu/ece_fac_articles/99

This Conference Proceeding is protected by copyright and/or related rights. It has been brought to you by Digital Scholarship@UNLV with permission from the rights-holder(s). You are free to use this Conference Proceeding in any way that is permitted by the copyright and related rights legislation that applies to your use. For other uses you need to obtain permission from the rights-holder(s) directly, unless additional rights are indicated by a Creative Commons license in the record and/or on the work itself.

This Conference Proceeding has been accepted for inclusion in Electrical and Computer Engineering Faculty Publications by an authorized administrator of Digital Scholarship@UNLV. For more information, please contact digitalscholarship@unlv.edu.

Open-Loop Vehicle Control Using an Abstraction of its Model

Patricia Mellodge and Pushkin Kachroo

Abstract—Feedback control design of complex systems can be made easier by working on simpler models of the system that are their abstractions. This paper presents a method to control a car-like robot using abstraction: the car is represented by a unicycle. A transformation is provided to calculate car inputs from unicycle inputs so that the car follows the unicycle trajectory whenever proper initial conditions are met. The transformation does not give correct results for the case when the unicycle is rotating. In this case, an open-loop optimal control algorithm is presented to generate car inputs. Simulation results are given for different initial car inputs and the results are compared.

I. INTRODUCTION

The control community is continually seeking methods that make designing controllers less complex. One common method is to simplify the model, design the controller on the simpler system, and implement that controller on the actual system. Linearization and model reduction are two approaches that have been used successfully for control design. Abstraction is a method of model simplification that shows promise in this area as well.

Abstraction is a technique that can be used to represent a given system by a simpler system without losing important characteristics, such as controllability, through the transformation [1]. Our work applies the abstraction technique to the car by using the unicycle as the simpler system. With this technique, a controller is designed for the unicycle that is transformed back to the car using a mapping from the unicycle inputs to the car inputs. This transformation should cause the car to follow the unicycle. However, a problem arises when the transformation is not well defined or gives erroneous inputs due to limitations in the car's dynamics that are not present in the unicycle's dynamics. The focus of this paper is to provide a solution for the transformation when the unicycle follows a trajectory that the car cannot.

The proposed solution takes the form of an optimal controller as a method of open-loop control that generates car inputs based on the unicycle inputs. Optimal control has been used extensively for path planning and control of nonholonomic systems. For examples, see the work in [2], [3], [4], [5], and [6]. The method described in this paper differs from the previous work in that it uses the unicycle as the system for control design and transforms the inputs to the car. In our solution, the calculations to determine the car inputs must be performed with a priori knowledge of the unicycle trajectory. This limits the practicality of the proposed solution because the controller cannot be used in

Fig. 1. The models for the unicycle (a) and car (b).

a feedback setting. Future work will involve extending these ideas to the closed-loop system.

II. KINEMATIC MODEL

This paper uses kinematic models for the unicycle and rear wheel drive car as developed in [2]. These models are derived by applying no-slip conditions on all of the wheels. The model for the unicycle is given by (see Fig. 1(a))

$$\begin{bmatrix} \dot{x}_u \\ \dot{y}_u \\ \dot{\theta}_u \end{bmatrix} = \begin{bmatrix} \cos \theta_u \\ \sin \theta_u \\ 0 \end{bmatrix} v_1 + \begin{bmatrix} 0 \\ 0 \\ 1 \end{bmatrix} v_2. \quad (1)$$

The inputs for this system are given by v_1 , the linear velocity in the direction of rolling, and v_2 , the angular velocity.

The kinematic model for the car is given by (see Fig. 1(b))

$$\begin{bmatrix} \dot{x} \\ \dot{y} \\ \dot{\theta} \\ \dot{\phi} \end{bmatrix} = \begin{bmatrix} \cos \theta \\ \sin \theta \\ \frac{\tan \phi}{l} \\ 0 \end{bmatrix} u_1 + \begin{bmatrix} 0 \\ 0 \\ 0 \\ 1 \end{bmatrix} u_2 \quad (2)$$

where u_1 is the linear velocity of the rear wheels and u_2 is the angular velocity of the front steering wheels.

Based on these models, an input transformation can be found to convert $v_1(t)$ and $v_2(t)$ into $u_1(t)$ and $u_2(t)$. This transformation is given by

$$\begin{aligned} u_1(t) &= v_1(t) \\ u_2(t) &= \frac{l[v_1(t)\dot{v}_2(t) - \dot{v}_1(t)v_2(t)]}{v_1^2(t) + l^2v_2^2(t)}. \end{aligned} \quad (3)$$

If the initial conditions of the car match those of the unicycle, $x(0) = x_u(0)$, $y(0) = y_u(0)$, $\theta(0) = \theta_u(0)$, and an additional condition is met for the initial steering angle, $\phi(0) = \tan^{-1}[lv_2(0)/v_1(0)]$, then the input transformation

P. Mellodge and P. Kachroo are with the Bradley Department of Electrical and Computer Engineering, Virginia Tech, Blacksburg, VA 24061
 pmellodg@vt.edu, pushkin@vt.edu

given in (3) will cause the car's trajectory to match the unicycle's. In other words,

$$\begin{aligned} x(t) &= x_u(t) \\ y(t) &= y_u(t) \\ \theta(t) &= \theta_u(t) \end{aligned}$$

for all t for which the unicycle trajectory is defined.

This transformation (3) will fail if either v_1 or v_2 is not differentiable or if both $v_1(t)$ and $v_2(t)$ are zero for any t . In these cases, the transformation is not well defined. However, there is an instance in which the transformation is well defined, but the resulting car inputs do not cause the car to track the unicycle. When $v_1 = 0$ and $v_2 \neq 0$, the unicycle is rotating about its axis. It is not possible for the car to perform this maneuver and the transformation results in both car inputs being zero. This latter situation is the focus of this paper.

III. TRACEABILITY

In this section, we formalize the above idea that the transformation (3) exists for $v_1 = 0$ and $v_2 \neq 0$, but the resulting car inputs do not make the car track the unicycle. The car cannot track the rotating unicycle.

Below we define traceability to describe the relationship between two Φ -related control systems: the original system and its abstraction in the sense of [1]. If the abstracted system's trajectories can be followed by the original system, we say the abstracted system is traceable by the original system. For the above example of the car, this relationship does not hold. However, the car can follow a circle of arbitrarily small radius, thus approximating the rotating unicycle. This idea is formalized in the definition of ϵ -traceability.

The concepts in this section are an extension of implementability and consistency, introduced in [1], and are based on a geometric view of control systems, as in [8].

Definition 3.1: Traceability: Consider a control system $S_M = (B_M, F_M)$ and a smooth surjection $\Phi : M \rightarrow N$. Let $c_N : I \rightarrow N$ be a trajectory of S_N where $I \subseteq \mathbb{R}$. Then S_N is traceable by S_M if for every trajectory c_N there exists a trajectory in S_M , $c_M : I \rightarrow M$, such that $\Phi(c_M(t)) = c_N(t)$ for all $t \in I$.

Let $S_N = (B_N, F_N)$ be a control system defined on Riemannian manifold $(N, \mathbf{g})$ with metric tensor $\mathbf{g} = (g_{ij})$ and let $c_1, c_2 : I \rightarrow N$ be trajectories in S_N . Then the distance between c_1 and c_2 is defined by the Poincare distance

$$d_P(c_1, c_2) = \sup_{t \in I} \sup_{\tau \in I} d(c_1(t), c_2(\tau)). \quad (4)$$

The distance between points $d(\cdot, \cdot)$, which is invariant with respect to changes in coordinate systems, is given as

$$d(q_1, q_2) = \inf \int_a^b \left| g_{ij} \frac{dx_i}{dt} \frac{dx_j}{dt} \right|^{1/2} dt \quad (5)$$

where $(x_1, \dots, x_n)$ are the local coordinates and the infimum is taken over all curves $\gamma(t)$ in N such that $\gamma(a) = q_1$ and $\gamma(b) = q_2$.

For the following development, car trajectories will be denoted by $X(t) = (x(t), y(t), \theta(t), \phi(t))$ and unicycle trajectories by $Y(t) = (x_u(t), y_u(t), \theta_u(t))$. The Φ -mapping

$$\Phi(x, y, \theta, \phi) = (x, y, \theta)$$

maps points in the car's manifold to the unicycle's. For a unicycle trajectory, c_N , that has no corresponding car trajectory, c_M , with $c_N(t) = \Phi(c_M(t))$, there does exist a car trajectory with $\Phi(c_M(t))$ arbitrarily close to $c_N(t)$.

Given a unicycle trajectory $Y(t) = (\alpha, \beta, \theta_u(t))$, where α and β are constants (i.e., $\dot{x}_u = \dot{y}_u = 0$), and given any $\epsilon > 0$, there exists a car trajectory $X(t)$ such that $d_P(Y, \Phi(X)) < \epsilon$. This can be seen by considering any $\epsilon > 0$. Choose the car's initial conditions as $x(0) = \alpha$, $y(0) = \beta$, $\theta(0) = \theta_u(0)$, and $\phi(0) \in (-\pi/2, \pi/2)$ so that $\tan \phi(0) > 2l/\epsilon$. Letting $u_1(t) = l\dot{\theta}_u(t)/\tan \phi(0)$ and $u_2(t) = 0$, the car's resulting trajectory $X(t)$ is

$$x(t) = \alpha + \frac{l}{\tan \phi(0)} \sin \theta_u(t) \quad (6)$$

$$y(t) = \beta + \frac{l}{\tan \phi(0)} (1 - \cos \theta_u(t)) \quad (7)$$

$$\theta(t) = \theta_u(t) \quad (8)$$

$$\phi(t) = \phi(0). \quad (9)$$

By choice of ϕ , $|x(t) - x_u(t)| < \epsilon$ and $|y(t) - y_u(t)| < \epsilon$ for all t . Hence $d_P(Y, \Phi(X)) < \epsilon$.

Definition 3.2: ϵ -Traceability: Let $S_M = (B_M, F_M)$ and $S_N = (B_N, F_N)$ be two control systems and let $\Phi : M \rightarrow N$ be a smooth surjection. Then S_N is ϵ -traceable by S_M if given $\epsilon > 0$ and a trajectory $c_N : I \rightarrow N$, there exists a trajectory $c_M : I \rightarrow M$ such that $d_P(c_N, \Phi(c_M)) < \epsilon$.

Definition 3.3: ϵ -Consistency: Let $S_M = (B_M, F_M)$ be a control system on M and let $\Phi : M \rightarrow N$ be a smooth surjection. Then S_M is ϵ -consistent with respect to Φ whenever the following holds. If for any $p_1, p_2 \in M$ there exist sequences p_{1_n} and p_{2_n} in M such that

1) there exist trajectories connecting $\Phi(p_{1_n})$ and $\Phi(p_{2_n})$ for all n and

2) $\Phi(p_{1_n}) \rightarrow \Phi(p_1)$ and $\Phi(p_{2_n}) \rightarrow \Phi(p_2)$,

then there is a trajectory connecting p_1 and p_2 .

As with implementability and consistency in [1], the relationship between ϵ -traceability and ϵ -consistency provide a means to propagate controllability between control systems S_M and S_N .

Theorem 3.1: Controllability Equivalence: Consider control systems $S_M = (B_M, F_M)$ and $S_N = (B_N, F_N)$ which are Φ -related with respect to smooth surjection $\Phi : M \rightarrow N$. Assume that S_N is ϵ -traceable by S_M and S_M is ϵ -consistent with respect to Φ . Then S_N is controllable if and only if S_M is controllable.

IV. OPEN-LOOP OPTIMAL CONTROL

Let the unicycle inputs, $v_1(t)$ and $v_2(t)$, be given for $t \in [0, t_f]$. The resulting trajectory is specified by $x_u(t)$, $y_u(t)$, and $\theta_u(t)$. The cost function is given by

$$J = \frac{1}{2} \int_0^{t_f} [w_1 e_\theta(t)^2 + w_2 (e_x(t)^2 + e_y(t)^2)] dt \quad (10)$$

where

$$\begin{aligned} e_\theta(t) &= \theta(t) - \theta_u(t) \\ e_x(t) &= x(t) - x_u(t) \\ e_y(t) &= y(t) - y_u(t). \end{aligned}$$

The weights, $w_1, w_2 \in [0, 1]$ with $w_1 + w_2 = 1$, determine the relative importance of the translational and orientation errors.

In this section we develop an open-loop optimal control scheme for the case when the unicycle is rotating but has no linear velocity. In particular, the unicycle trajectory that the car must follow starts at the origin and includes a 90 degree left turn. This trajectory is generated by the unicycle inputs

$$v_1(t) = \begin{cases} v_s & t \in [0, t_s) \\ 0 & t \in [t_s, t_s + t_t) \\ v_s & t \in [t_s + t_t, 2t_s + t_t] \end{cases} \quad (11)$$

$$v_2(t) = \begin{cases} 0 & t \in [0, t_s) \\ \omega & t \in [t_s, t_s + t_t) \\ 0 & t \in [t_s + t_t, 2t_s + t_t] \end{cases} \quad (12)$$

where v_s and t_s determine the length of the straight portion and ω and t_t determine the angle of the turn.

Given the cost function (10), the Hamiltonian is

$$\begin{aligned} H &= \frac{1}{2}w_1 e_\theta(t)^2 + \frac{1}{2}w_2 (e_x(t)^2 + e_y(t)^2) \\ &+ p_1(t)u_1(t) \cos \theta(t) + p_2(t)u_1(t) \sin \theta(t) \\ &+ p_3(t)u_1(t) \frac{\tan \phi(t)}{l} + p_4(t)u_2(t). \end{aligned} \quad (13)$$

If we denote $[x \ y \ \theta \ \phi]^T$ by X and $[p_1 \ p_2 \ p_3 \ p_4]^T$ by p , then the necessary conditions for optimal control are given by [7]

$$\dot{X}^*(t) = \begin{bmatrix} u_1^*(t) \cos \theta^*(t) \\ u_1^*(t) \sin \theta^*(t) \\ u_1^*(t) \frac{\tan \phi^*(t)}{l} \\ u_2^*(t) \end{bmatrix} \quad (14)$$

$$\dot{p}^*(t) = - \begin{bmatrix} w_2 e_x^*(t) \\ w_2 e_y^*(t) \\ u_1^*(t) [w_1 e_\theta^*(t) p_1^*(t) \sin \theta^*(t) + p_2^*(t) \cos \theta^*(t)] \\ p_3^*(t) u_1^*(t) \frac{\sec^2 \phi^*(t)}{l} \end{bmatrix} \quad (15)$$

$$0 = \begin{bmatrix} p_1^*(t) \cos \theta^*(t) + p_2^*(t) \sin \theta^*(t) + p_3^*(t) \frac{\tan \phi^*(t)}{l} \\ p_4^*(t) \end{bmatrix} \quad (16)$$

where the $*$ denotes the optimal values.

The boundary conditions are

$$\begin{bmatrix} x^*(0) \\ y^*(0) \\ \theta^*(0) \\ \phi^*(0) \end{bmatrix} = \begin{bmatrix} x_u(0) \\ y_u(0) \\ \theta_u(0) \\ \tan^{-1} \frac{l \dot{\theta}_u(0)}{v_1(0)} \end{bmatrix} \quad (17)$$

$$\begin{bmatrix} x^*(t_f) \\ y^*(t_f) \\ \theta^*(t_f) \\ \phi^*(t_f) \end{bmatrix} = \begin{bmatrix} x_u(t_f) \\ y_u(t_f) \\ \theta_u(t_f) \\ \tan^{-1} \frac{l \dot{\theta}_u(t_f)}{v_1(t_f)} \end{bmatrix}. \quad (18)$$

Fig. 2. The car follows an inscribed circle based on its maximum steering angle.

These boundary conditions were chosen so that before and after this maneuver, the transformation (3) can be used to determine the car's inputs from the unicycle's. However, these equations are difficult to solve simultaneously, so the following steepest descent algorithm, adapted from [7], was used to find an approximate solution.

- 1) Choose an initial $u_1(t)$ and $u_2(t)$.
- 2) Using the initial conditions (17), integrate the car dynamics (2) to obtain the car's trajectory $x(t)$, $y(t)$, $\theta(t)$, and $\phi(t)$ for $t \in [0, t_f]$.
- 3) Using the boundary condition $p(t_f) = 0$ and (15), backward integrate to obtain the p_i 's.
- 4) Calculate the cost using (10).
- 5) If the cost is higher than some predetermined threshold, update the inputs according to

$$u_1^{(i+1)} = u_1^{(i)} - \tau \frac{\partial H}{\partial u_1} \quad (19)$$

$$u_2^{(i+1)} = u_2^{(i)} - \tau \frac{\partial H}{\partial u_2} \quad (20)$$

where τ is the step size constant.

Steps 2-5 are repeated until the value of the cost function is sufficiently low.

This algorithm finds the costates (p_i 's) by forcing their final values to zero. However, the boundary condition on the final states, $x(t_f)$, $y(t_f)$, $\theta(t_f)$, $\phi(t_f)$, is not enforced. As a result, the car's final state is allowed to deviate from the unicycle's final state. The success of this algorithm depends on the initial input. Two initial inputs, shown in Figs. 2 and 3, were used.

The first trajectory was created by determining the two points at which the inscribed circle touched the unicycle trajectory. This is shown in Fig. 4 for the general case of any angle α . The radius of the circle depends on the car's maximum steering angle (in radians) and the wheelbase length:

$$r = \frac{l}{\tan(\phi_{max})}. \quad (21)$$

Fig. 3. The car follows two circles based on its maximum steering angle.

The distance d_1 is given by

$$d_1 = \frac{r}{\tan \frac{\alpha}{2}}. \quad (22)$$

The car follows the unicycle trajectory until it reaches the first intersection point, whose x -coordinate is given by

$$x_c = v_s t_s - d_1. \quad (23)$$

The car must follow the arc defined by β for the same amount of time that the unicycle traveled distance d_1 , turned, and traveled distance d_1 again. Thus the car's linear velocity during this time is

$$u_{1turn} = \frac{s}{t_{arc}} \quad (24)$$

where

$$s = r\beta \quad (25)$$

$$\beta = \pi - \alpha \quad (26)$$

$$t_{arc} = \frac{2d_1}{v_s} + t_t. \quad (27)$$

The car control inputs that generate this trajectory are

$$u_1(t) = \begin{cases} v_s & t \in [0, t_c) \\ u_{1turn} & t \in [t_c, t_c + t_{arc}) \\ v_s & t \in [t_c + t_{arc}, t_f] \end{cases} \quad (28)$$

$$u_2(t) = \begin{cases} 0 & t \in [0, t_c) \\ \frac{u_{2max}}{2} & t = t_c \\ 0 & t \in (t_c, t_c + t_{arc}) \\ -\frac{u_{2max}}{2} & t = t_c + t_{arc} \\ 0 & t \in (t_c + t_{arc}, t_f] \end{cases} \quad (29)$$

where $t_c = x_c/v_s$ and $u_{2max}/2$ is the input that causes the wheels to instantaneously turn through $\phi_{max}/2$.

For the second initial trajectory, the car follows the unicycle until the turning point. Then the car made two turns, following two arcs determined by the car's maximum steering angle. The construction of the two arcs is shown in Fig. 5. The distance d_2 is given by

$$d_2 = r(\sin \alpha + \lambda) \quad (30)$$

Fig. 4. The intersection of the inscribed circle with the unicycle's trajectory.

Fig. 5. The method of sequential circles for the car to follow the unicycle's trajectory.

where $\lambda = \sqrt{3 - \cos^2 \alpha + 2 \cos \alpha}$.

The angles, β_1 and β_2 , that define the two arcs, s_1 and s_2 , are

$$\beta_1 = \pi - \alpha + \tan^{-1} \left[\frac{\lambda}{(1 - \cos \alpha)(\cos \alpha)} - \frac{\lambda}{\cos \alpha} \right] \quad (31)$$

$$\beta_2 = \tan^{-1} \left[\frac{\lambda}{1 - \cos \alpha} \right]. \quad (32)$$

The car follows the unicycle trajectory until it reaches the turn, whose x -coordinate is given by

$$x_s = v_s t_s. \quad (33)$$

As before, the car must travel the distance $s_1 + s_2$ in the same time it takes the unicycle to complete the turn and travel distance d_2 . The car's velocity during this time is

$$u_{1turns} = \frac{s}{t_{turns}} \quad (34)$$

where $s = r(\beta_1 + \beta_2)$ and $t_{turns} = t_t + d_2/v_s$.

The car inputs that result in this trajectory are

$$u_1 = \begin{cases} v_s & t \in [0, t_s) \\ u_{1turns} & t \in [t_s, t_s + t_{turns}) \\ v_s & t \in [t_s + t_{turns}, t_f] \end{cases} \quad (35)$$

TABLE I

THE VALUE OF THE COST FUNCTION J FOR VARIOUS WEIGHTS AND MAXIMUM STEERING ANGLES.

ϕ_{max}	$w_1 = 0$ $w_2 = 1$	$w_1, w_2 = 0.5$	$w_1 = 1$ $w_2 = 0$
$\pi/3$	0.0774	0.1006	0.1238
$\pi/4$	0.2100	0.2771	0.3441
$\pi/5$	0.4022	0.5070	0.6119
$\pi/6$	0.5837	0.7507	0.9178

TABLE II

THE VALUE OF THE COST FUNCTION J FOR VARIOUS INITIAL INPUTS.

Initial Input	$w_1 = 0$ $w_2 = 1$	$w_1, w_2 = 0.5$	$w_1 = 1$ $w_2 = 0$
Inscribed Circle	0.1329	0.1089	0.0127
Sequential Circles	0.4383	0.3419	0.0253
Straight Trajectory	0.5125	1.2395	0.0767

$$u_2 = \begin{cases} 0 & t \in [0, t_s) \\ \frac{u_{2max}}{2} & t = t_s \\ 0 & t \in (t_s, t_s + t_1) \\ -u_{2max} & t = t_s + t_1 \\ 0 & t \in (t_s + t_1, t_s + t_1 + t_2) \\ \frac{u_{2max}}{2} & t = t_s + t_1 + t_2 \\ 0 & t \in (t_s + t_1 + t_2, t_f] \end{cases} \quad (36)$$

where $t_1 = \frac{\beta_1}{\beta_1 + \beta_2} t_{turns}$ and $t_2 = \frac{\beta_2}{\beta_1 + \beta_2} t_{turns}$.

V. SIMULATION

The optimal control algorithm described in the previous section was simulated in MATLAB. First, the starting trajectories were generated and the cost function was calculated for various weights and maximum steering angles. The results are shown in Table I.

The MATLAB simulation was run for three different initial inputs: the two described in the previous section and one that causes the car to drive on a straight trajectory

$$u_1(t) = v_1(t) \quad (37)$$

$$u_2(t) = 0. \quad (38)$$

In all cases the car's initial conditions satisfied (17). The car constants l and ϕ_{max} were set to 0.1 and $\pi/4$ respectively. The step size τ was 0.01, the sampling time T was 0.01, and the maximum value for u_2 was 100000. The simulation was run for 1000 iterations or until the change in the cost function J was less than 0.0001, whichever occurred first. The results are summarized in Tables II and III.

TABLE III

THE NUMBER OF ITERATIONS FOR VARIOUS INITIAL INPUTS.

Initial Input	$w_1 = 0$ $w_2 = 1$	$w_1, w_2 = 0.5$	$w_1 = 1$ $w_2 = 0$
Inscribed Circle	307	182	142
Sequential Circles	772	427	294
Straight Trajectory	1001	1001	612

Fig. 6. The trajectory resulting from the initial inscribed circle with $w_1 = 1$ and $w_2 = 0$.

Fig. 7. The car's inputs resulting from the initial inscribed circle with $w_1 = 1$ and $w_2 = 0$.

In all three cases, the lowest cost and the fastest convergence were obtained with $w_1 = 1$ and $w_2 = 0$ (i.e., when only the orientation error was considered). For the initial straight trajectory when the x, y -error was included in the cost function, the algorithm did not converge in 1000 steps. The inscribed circle converged to its result the fastest and had the overall lowest cost. The final trajectory and inputs for this case are shown in Figs. 6 and 7. The final position of the car does not match that of the unicycle exactly. This was the case for all of the trajectories resulting from the optimal control algorithm. Because the final costates (p_i 's) were forced to zero, the boundary conditions for the car's final state (18) were not enforced.

VI. CONCLUSION

This paper presented an open-loop control algorithm for a car-like robot to follow a unicycle trajectory. Abstraction was used to represent the car by a unicycle. An input transformation was given that converted the unicycle inputs to the car inputs, causing the car to track the unicycle trajectory. This transformation did not provide the correct

car inputs when the unicycle rotated. When the unicycle trajectory had rotation, an optimal algorithm was used to generate inputs so that the car could closely follow the rotation. Three different initial inputs were used in simulation and their performance compared.

REFERENCES

- [1] G. Pappas, G. Lafferriere, and S. Sastry, "Hierarchically consistent control systems," *IEEE Transactions on Automatic Control*, Vol. 45, No. 6, pp. 1144-1160, June 2000.
- [2] J.-P. Laumond, *Robot Motion Planning and Control*, in *Lecture Notes in Control and Information Sciences*, No. 229, Springer, New York, NY, 1998.
- [3] D.J. Balkcom and M.T. Mason, "Extremal trajectories for bounded velocity mobile robots," in *Proceedings of the 2002 IEEE International Conference on Robotics and Automation*, pp. 1747-1752, Washington, DC, May 2002.
- [4] J. Wang and Z. Qu, "A near optimal tracking control of dynamic non-holonomic systems," in *Proceedings of the 2003 IEEE International Conference on Robotics, Intelligent Systems and Signal Processing*, pp. 202-207, Changsha, China, October 2003.
- [5] K. Moriwaki, "On automatic motion control with optimization," in *SICE Annual Conference Proceedings*, pp. 1248-1253, Fukui, Japan, August 2003.
- [6] P. Soueres and J.-P. Laumond, "Shortest paths synthesis for a car-like robot," *IEEE Transactions on Automatic Control*, Vol. 41, No. 5, pp. 672-688, May 1996.
- [7] D.E. Kirk, *Optimal Control Theory: An Introduction*, Dover Publications, Inc., Mineola, NY, 2004.
- [8] H. Nijmeijer and A.J. van der Schaft, *Nonlinear Dynamical Control Systems*, Springer-Verlag, New York, NY, 1990.