Path Planning and Robust Tracking for A Car-Like Robot

M. Egerstedt*, X. Hu, H. Rehbinder and A. Stotsky

Optimization and Systems Theory
Royal Institute of Technology
S - 100 44 Stockholm, Sweden

Abstract. In this paper the problem of path planning and path following for a car-like robot is considered. Some new algorithms are proposed, which, by experiments, have been proven to be quite robust.

1 Introduction

From the point of view of systems and control, starting from the work of Brockett [3] several methods were developed for solving path planning and following problems for a car-like robot [4],[5],[9],[10],[13],[16].
However, most control algorithms presented in the literature either use a simple kinematic model of the following form

$$\dot{x} = v\cos\theta \tag{1}$$
$$\dot{y} = v\sin\theta \tag{2}$$
$$\dot{\theta} = \omega \tag{3}$$

and/or do not consider the actual constraints on the range of steering angles. Where in the equations x and y are cartesian coordinates of the middle point on the front axle , θ is orientation angle, v is longitudinal velocity, ω is angular velocity.
The more realistic model can be derived based on the assumption that the vector of longitudinal velocity is parallel to the front wheels. This more sophisticated model can be found in, for example, [14].

$$\begin{aligned}\dot{x} &= v\cos(\theta+\phi)\\ \dot{y} &= v\sin(\theta+\phi)\\ \dot{\theta} &= \frac{v}{l}\sin\phi\\ \dot{\phi} &= u\end{aligned} \tag{4}$$

where x and y are cartesian coordinates of the middle point on the front axle, θ is orientation angle, v is longitudinal velocity, l is the distance of the two axles, and ϕ is the steering angle. (Note that in some papers, for example, [16], the middle point of the back axle is considered instead.) It is evident that the model (4) is more realistic than (3).

Although there already exist a number of algorithms to solve the path planning and tracking problem for model (4), such as in [9] and [16], none of them actually considers the constraints on the steering angles and on the environment (obstacles). Furthermore some of the path following controls have proven to be not robust in our radio controlled car system, where the A/D and D/A conversions of the velocity and steering angle are fairly coarse.

In this paper, we first formulate the path planning problem as a non-convex programming problem in which the distance is to be minimized. Then we use a path following control algorithm, which has proven to be fairly robust to the conversion errors, to drive the car. We will demonstrate our methods in the parallel parking problem.

* Email: magnuse@math.kth.se

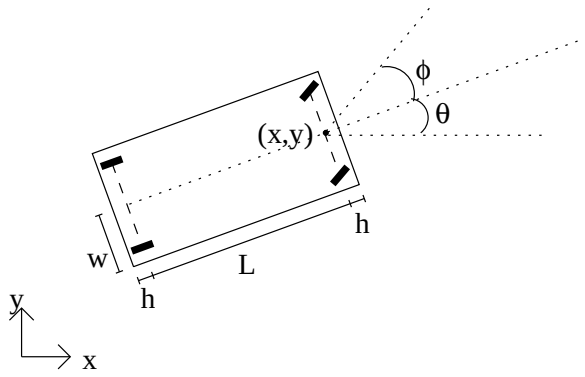

Fig. 1.: The geometry of the car-like robot, with position (x, y), orientation θ and steering angle ϕ.

2 Car-Like Robot

In this section we briefly review the nonholonomic constraints on a car-like robot. The nonholonomic constraints are basically due to the two facts of a car: no movement in the direction orthogonal to the front wheels and in the direction orthogonal to the rear wheels, and the range of steering is limited. (Naturally if dynamical factors such as friction forces acting on the tires are considered, then the situation is much more complicated.)

That the velocity orthogonal to the front wheels should be zero implies

$$\dot{x}\sin(\theta+\phi) - \dot{y}\cos(\theta+\phi) = 0, \tag{5}$$

That the velocity orthogonal to the rear wheels should be zero implies

$$\frac{d}{dt}(x - l\cos(\theta))\sin(\theta) - \frac{d}{dt}(y - l\sin(\theta))\cos(\theta) = 0, \tag{6}$$

The reason we use the differential one-form constraints here, instead of the vector field form of (4), is that we want to end up with a formulation that is suitable for mathematical programming. Using this as our motivation, we now derive the maximal steering angle constraint.

If we are driving a car with the steering angle fixed at maximum, we would get a curve that looks like

$$x_{\phi_{\max}} = \rho\sin(\theta+\phi_{\max}) \tag{7}$$
$$y_{\phi_{\max}} = -\rho\cos(\theta+\phi_{\max}), \tag{8}$$

so differentiating this with respect to time gives us that

$$\dot{x}_{\phi_{\max}} = \rho\dot{\theta}\cos(\theta+\phi_{\max}) \tag{9}$$
$$\dot{y}_{\phi_{\max}} = \rho\dot{\theta}\sin(\theta+\phi_{\max}). \tag{10}$$

Comparing this with equation (4) gives us that

$$\dot{\theta}\rho = v_1(t) = \frac{\dot{\theta}L}{\sin(\phi_{\max})} \Rightarrow \rho = \frac{L}{\sin(\phi_{\max})}. \tag{11}$$

Now the maximal steering angle constraint can be expressed as

$$\dot{x}^2 + \dot{y}^2 = v_1(t)^2 \geq \rho^2\dot{\theta}^2, \tag{12}$$

so we get

$$\dot{x}^2 + \dot{y}^2 - \rho^2\dot{\theta}^2 \geq 0. \tag{13}$$

3 The Programming Problem

We are now ready to actually find a feasible path between two state configurations, where x, y, and θ are our state variables. Here we want to minimize the distance, so

$$\min \int_{t_0}^{t_f} v_1(t)dt, \tag{14}$$

but for the sake of simplicity, we focus on

$$\min \int_{t_0}^{t_f} v_1(t)^2 dt = \min \int_{t_0}^{t_f} (\dot{x}^2 + \dot{y}^2)dt. \tag{15}$$

We also restrict our allowed set of solutions to a part of $\mathbf{R}^2$, $(\mathbf{R}^2 \backslash \mathcal{S})$, that is not occupied by obstacles, where we believe that a good choice of $\mathcal{S}$ is the union of the convex hulls around each physical obstacle. We also need to make sure that not only the middle point of the front axle is in the allowed set, but that the whole car is in this set. One way of doing this is to view the car as a rigid body and make sure that neither of the car's corner points are in $\mathcal{S}$.

We now, however, have a infinite dimensional programming problem. Though it is our goal to eventually solve this original problem, in this paper we first discretize the problem. Then we end up with

$$\min \sum_{k=1}^{N-1} [(x_{k+1} - x_k)^2 + (y_{k+1} - y_k)^2] \tag{16}$$

when

$$\begin{cases} (x_{k+1} - x_k + l(\theta_{k+1} - \theta_k)\sin(\theta_k))\sin(\theta_k) - \\ (y_{k+1} - y_k - l(\theta_{k+1} - \theta_k)\cos(\theta_k))\cos(\theta_k) = 0 \quad k = 1, \cdots, N-1 \\ (x_{k+1} - x_k)^2 + (y_{k+1} - y_k)^2 - (\theta_{k+1} - \theta_k)^2\rho^2 \geq 0 \quad k = 1, \cdots, N-1 \\ (x, y)_{\text{left}}^{\text{front}} \notin \mathcal{S}, \quad (x, y)_{\text{right}}^{\text{front}} \notin \mathcal{S} \\ (x, y)_{\text{left}}^{\text{rear}} \notin \mathcal{S}, \quad (x, y)_{\text{right}}^{\text{rear}} \notin \mathcal{S} \\ x_1 = x_{\text{start}}, \quad x_N = x_{\text{end}} \\ y_1 = y_{\text{start}}, \quad y_N = y_{\text{end}} \\ \theta_1 = \theta_{\text{start}}, \quad \theta_N = \theta_{\text{end}}. \end{cases} \tag{17}$$

However, since this is a non-convex programming problem, the solution depends heavily on a wise choice of the starting guess for the system, since we can only, in general, hope to come up with local minimas. If we, for instance, start with cubic splines interpolating between the initial and final state space configurations, we get a solution that can be seen in figure 2.

The reason why we chose to use cubic splines as our start guess is that they minimize

$$\int_{x_0}^{x_T} f''(x)^2 dx. \tag{18}$$

This is a useful property since in order for a trajectory to be feasible, the curvature of the planned trajectory can not be greater than the curvature of the trajectory generated when driving the car with a maximal steering angle. The curvature of a scalar function is given by

$$\kappa = \frac{f''(x)}{(1 + f'(x)^2)^{3/2}}. \tag{19}$$

One way of generating feasible paths, hopefully within the steering angle constraints, for a car-like robot, is thus by making the second derivative of the curve as small as possible, since

$$\kappa^2 = \frac{f''(x)^2}{(1 + f'(x)^2)^3} \leq f''(x)^2, \tag{20}$$

and this is just what we hope to accomplish when basing our solution on cubic splines.

There is, however, one other problem that we need to solve, and that is that the number of iterations needed, in order to come up with a solution, is rather high. Therefore we are presently working on different

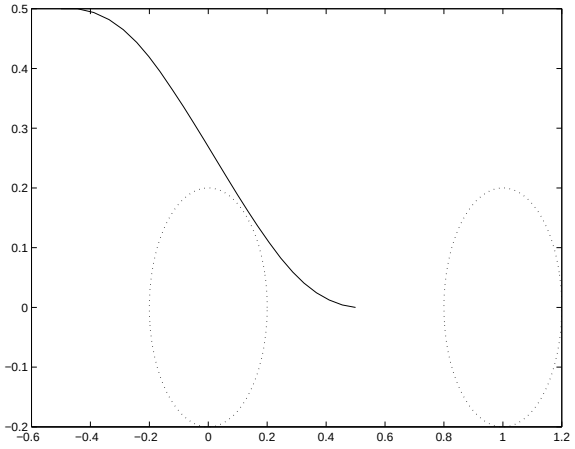

Fig. 2.: The optimal parallel parking trajectory, produced with the other cars represented as circles in order to simplify the problem somewhat.

ways of to simplify the programming problem formulation, and we hope that it is going to be possible, in the future, to make some sort of convexification as well, all in order to reduce the required number of iterations. Since we do nor want our RC-car to "think" for five minutes before it actually starts moving, we need to come up with some planning scheme that may not produce optimal solutions, but hopefully feasible ones that are not too far away from the optimal one, in order for us to have something to try out our control algorithms on.

4 Bang-Bang Type Planner

Based on the work of [11], [9] and [6], we now proceed with a type of planner that combines splines with a bang-bang planner. The general idea is that splines are used to plan a path that takes the car close enough to the place where we want to do fine maneuvering, such as parallel parking. We then switch to a different planning mode where we use a bang bang type of planner, using parts of circles, produced by a maximal steering of the car, combined with straight lines. The results from such an approach can be seen in figures 3-4, where in figure 3 a comparison can be made with the optimal path, which gives us an indication that this type of approach isn't so bad after all.

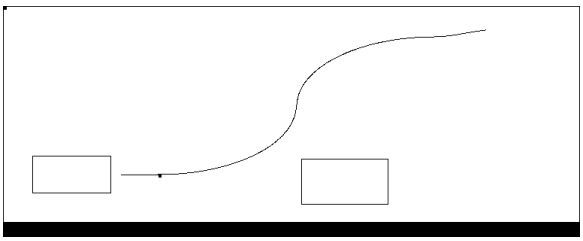

Fig. 3.: The planned parallel parking path for our actual car-like robot, where the rectangles represent other cars.

One main advantage with this type of planner is that it is based on algebraic calculations only. We do not need to solve any programming problems and our solutions depend explicitly on the desired safety margins, since the interpolation points can be specified directly, depending on how far away from the obstacles, the other parked cars, we want to be.

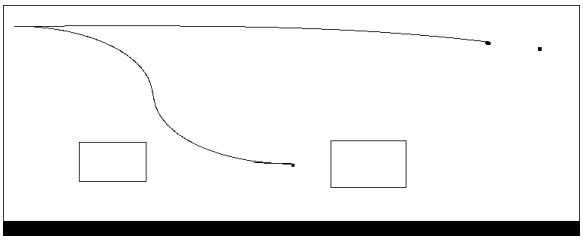

Fig. 4.: The planned parallel parking path for our actual car like robot, where a backing maneuver is needed since a forward drive will not produce a path within safety margins.

5 Simulations

Before one implements a new controller, it is always wise to first try it by simulation. For path planning the kinematic model used might be good enough, but for simulation and estimation it is too far from reality. For example, one basic feature that the kinematic model fails to recapture is that when driving a real car in a circle, the radius is not only dependent on the steering angle but also on the speed. A more advanced model can be found in [1]. We will here give a short review of that model.

The single track model

As a primary simplification, we consider a car with only one front wheel and one rear wheel. Let the steering angle be δ, the speed of the front and rear wheels $\bar{v}_f$ and $\bar{v}_r$, their deviation from the longitudinal direction of the car β_f and β_r and the side forces acting on the tires f_f and f_r (figure 5).

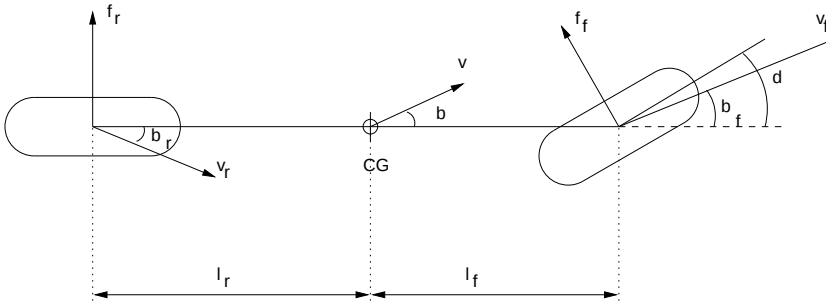

Fig. 5.: The single-track model. Redrawn from Ackermann

Let the cars orientation relative to an inertially fixed coordinate system be ψ, its mass m and its moment of inertia J. As states are taken: v the speed of the cars center of gravity, β the deviation of $\bar{v}$ from the cars longitudinal axis - the *vehicle sideslip angle* and the *yaw rate* $r = \dot{\psi}$. The equations of motion can now be found to be

$$\begin{pmatrix} mv(\dot{\beta} + r) \\ m\dot{v} \\ J\dot{r} \end{pmatrix} = \begin{pmatrix} -\sin\beta & \cos\beta & 0 \\ \cos\beta & \sin\beta & 0 \\ 0 & 0 & 1 \end{pmatrix} \begin{pmatrix} f_x \\ f_y \\ m_z \end{pmatrix} \tag{21}$$

where

$$\begin{pmatrix} f_x \\ f_y \\ m_z \end{pmatrix} = \begin{pmatrix} -\sin\delta & 0 \\ \cos\delta & 1 \\ l_f\cos\delta & -l_r \end{pmatrix} \begin{pmatrix} f_f \\ f_r \end{pmatrix} \tag{22}$$

For the side forces f_f and f_r a simple linear model can be used if the slipping is small.

$$\begin{pmatrix} f_f \\ f_r \end{pmatrix} = \mu \begin{pmatrix} \delta_f - \beta_f \\ -\beta_r \end{pmatrix} \tag{23}$$

Here μ is the adhesion coefficient between the wheels and the floor. It is assumed to be the same for all wheels. Finally a geometrical consideration gives

$$\tan\beta_f = \tan\beta + \frac{l_f r}{v\cos\beta} \tag{24}$$

$$\tan\beta_r = \tan\beta - \frac{l_r r}{v\cos\beta} \tag{25}$$

The main difficulties in using the dynamical model are that the mass is usually uncertain (varying with load) and that the tire characteristics are uncertain.

For our indoor application, the mass is known and the car runs only on surface of the same type. So the main issue is just to measure the adhesion coefficient. We have devised a way to measure the coefficient based on relations between β and v. The coefficient for our tires (assuming they are identical) is $\mu = 3.00 \pm 0.5$.

The measurement has been verified by comparing simulation with actual experiment in which we compare the "breaking away speed" (when the car starts to slip away from the circle).

6 Path following control

The car system we have is a fairly cheap radio controlled toy car. the A/D and D/A conversions are coarse and there exists a deadzone in the servo system of the car. Thus the steering is far from precise. So any open-loop control is not going to work, neither are closed-loop controls which require fairly accurate steering.

On the other hand, bad mechanical systems require good control strategies. The algorithm we propose here basically only requires that the direction of the steering is right.

The main ideas of our algorithm are as follows (see Figure 6).

Step 1. choose a point P on the planned path to track. The angle ϕ between the orientation of the car and the direction from the middle point A on the front axle to P will be the steering angle.

Step 2. when the car is "close enough" to that point, update the point in Step 1 and repeat.

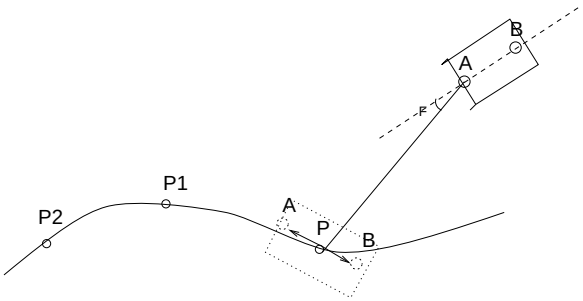

Fig. 6.: The path following control algorithm

The main issue here is how we define "close enough". Any distance related criteria proved not working well (the car might get stuck around that point).

Our criterion here is that as long as the car has "passed" the point, we update. We define "passing" as follows: if $(x_A - x_P)(x_B - x_P)$ or $(y_A - y_P)(y_B - y_P)$ becomes negative, then the car has passed the point P.

7 Conclusions

One of the reasons why we got interested in these problems to start with was that we noticed that in the literature nonholonomic kinematical constraints seemed to be neglected in most control and planning situations. The idea of using local, reactive controllers, as for example suggested in the influential paper [2], seemed to conflict with the idea of constructing feasible paths for robots of car-like type. We were also

interested in finding out how complex controllers would work in a real life situation where the knowledge of how the inputs affect the outputs is limited. In the case of our RC-car, we basically just know the sign of the desired steering angle, and therefore the need for a very robust control strategy is high. We have used our planners and control algorithms on the RC-car and our aim is to make it do parallel parking in a cluttered environment, with good safety margins. Now in real experiments, the car tracks the planned paths with a reasonably high accuracy, but we hope to improve both the planner and the controller further on in order to make them even more robust.

References

1. J. Ackermann, "Robust Control", 1993, Springer-Verlag, London pp 371-375
2. R.A Brooks, "A Robust Layered Control System for a Mobile Robot", *IEEE Journal of Robotics and Automation*, 2(1):14-23, March 1986.
3. R.W. Brockett " Asymptotic stability and feedback stabilization" in *Differential Geometric Control Theory* (Brockett,Millmann and Sussman, eds), pp.181-191, Boston, MA,USA: Birkhauser: 1983.
4. C.Canudas de Wit and O.J. Sordalen " Exponential stabilization of mobile robots with nonholonomic constraints" IEEE Tran. on Automatic Control vol. 11, N11, pp. 1791-1797, 1992.
5. B. d'Andrea Novel, G. Bastin, and G. Campion, "Dynamic feedback linearization of nonholonomic wheeled mobile robots" in *Proc. IEEE Conf. on Robotics and Automation*, (Nice, France), pp. 2527-2532, 1992.
6. L.E Dubin, "On Curves with a Minimum Length with a Constraint on Average Curvature and with Prescribed Initial and Terminal Positions and Tangents", *American Journal of Mathematics*, 79:497-516, 1957.
7. aybusovich L. and J.B. Moore, "Infinite Dimensional Quadratic Optimization: Interior-Point Methods and Control Applications", to appear in *Journal of Applied Mathematics and Optimization*.
8. Faybusovich L. and J.B. Moore, "Infinite Dimensional Quadratic Optimization: Interior-Point Methods and Control Applications", to appear in *Journal of Applied Mathematics and Optimization*.
9. Frezza R. and G. Picci "On line path following by recursive spline updating", Preprint.
10. Guldner J. and V. Utkin " Stabilization of nonholonomic mobile robots using Lyapunov functions for navigation and sliding mode control. *Proc. of the 33-rd CDC*, Lake Buena Vista, FL-December 1994, pp.2967-2972.
11. J-C. Latombe: *Robot Motion Planning*, Kluwer Academic Publishers, 1991.
12. D.G. Luenberger: *Optimization by Vector Space Methods*, John Wiley and Sons, Inc., New York, 1969.
13. R. Murray and S. Sastry " Nonholonomic motion planning: steering using sinusoids", IEEE Transactions on Automatic Control vol. 38, N5, pp. 700-716, 1993.
14. Nelson E., "Tensor Analysis" Princeton University Press, Princeton, 1967.
15. Nijmeijer H. and A.J. Van der Shaft " Nonlinear Dynamical Systems", 1990, Springer-Verlag, New-York,Inc., 467 pp.
16. C. Samson, "Velocity and torque control of a nonholonomic cart", in *Advanced Robot Control* (C. Canudas de Wit, ed), pp. 125-151, Berlin, Germany: Springer-Verlag, 1991.
17. M. Vidyasagar, Decomposition techniques for large-scale systems with nonadditive interactions: stability and stabilizability, *IEEE Trans. Automatic Control* **AC-25**, 1980, pp. 773-776.

This article was processed using the TEX macro package with SIRS97 style