Mechatronics 54 (2018) 167–174

Contents lists available at ScienceDirect

Mechatronics

journal homepage: www.elsevier.com/locate/mechatronics

Drift control for cornering maneuver of autonomous vehicles

Fang Zhang[a], Jon Gonzales[b], Shengbo Eben Li[a], Francesco Borrelli[b], Keqiang Li[*,a,c]

[a] Department of Automotive Engineering, Tsinghua University, Beijing, China
[b] Department of Mechanical Engineering, University of California, Berkeley, USA
[c] State Key Laboratory of Automotive Safety and Energy, Tsinghua University, Beijing 100084, PR China

ARTICLE INFO

Keywords:
Autonomous vehicles
Drift control
Path planning
Motion control
Vehicle dynamics

ABSTRACT

Expert drivers have the ability to perform high side-slip angle maneuvers, like drifting, during racing to minimize lap time or avoid obstacles. Designing planning and control algorithms for autonomous drift maneuvers, however, is challenging because of the high lateral motion and nearly full saturation of rear tires. In this paper, the authors propose a complete path planning and motion control framework to plan and track a reference drift trajectory along a sharp bend in a track. The path planner divides the path horizon into three regions, finds a path using different planning sub-modules, and then concatenates the solutions to generate the reference trajectory. The controller then applies a mixed open-loop and closed-loop scheme to track the reference trajectory. We validate the planning and control algorithms in simulation using a high-fidelity model in Simulink/Carsim, and through experimentation using 1/10 scale Radio-Control (RC) vehicle.

1. Introduction

Drifting occurs when an expert driver intentionally maneuvers a vehicle to cause loss of traction in the wheels, characterized by large side-slip angles and near full saturation of the wheels. It is commonly seen in rally racing when a driver quickly turns a corner.

Drifting represents a particularly interesting control maneuver because of the tire saturation and limited control authority in a highly unstable region. Current chassis control systems, like anti-lock braking system (ABS) and traction control system (TCS), try to prevent drifting conditions from ever arising [1,2], but experimental evidence shows the high-drift maneuvers may be more efficient from the minimum time point of view or obstacle avoidance. In rally racing, expert drivers often bring the vehicle into a drift state in order to reduce lap time or avoid collisions, while still maintaining control of the vehicle. To better understand these dynamics, Velenis et al. analyzed the behavior of expert drivers during drift and provided an empirical description of the sequence of steps the driver implements to initiate and control drift [3,4].

Most research on drift maneuvers fall into one of two categories: sustained drift and transient drift. Sustained drift focuses on stabilizing the vehicle about an unstable equilibrium state, resulting in a steady state circular drift. Transient drift focuses on entering a drift state temporarily to perform a maneuver, like drift parking.

For sustained drift, researchers have used vehicle models of varying fidelity to study drift dynamics, ranging from a two-state bicycle model [5,6] to a seven-state vehicle model [7,8]. Regardless of model fidelity, the system model must accurately capture the tire forces that emerge throughout drift, especially when the tires saturate. The Fiala tire model [9] and the Pacejka tire model [7,10] have been used to capture these forces. For the control design of sustained drift, the central feature lies the coordination between steering and rear drive torque [7,9,11,12]. Gonzales et al. designed a controller by linearizing the vehicle model around one of its drift equilibria and then used a linear quadratic regulator (LQR) feedback policy to compute the steering angle and rear drive [12]. Hindiyeh et al. applied dynamic surface control to balance the inputs and enabled ‘steering’ of rear tire through novel usage of rear drive for lateral control [9]. Additionally, Hindiyeh provided stability guarantees in the control design using Lyapunov-based techniques. Outside of model-based optimal control, Cutler applied reinforcement learning with a motion capture system to achieve sustained drift [13–15].

Work on transient drift has also emerged as a research topic for vehicle applications [4,16–21]. Chakraborty et al. investigated methods for mitigating unavoidable collisions using nonlinear optimization. They found handbrake cornering drift to be optimal maneuver in some situations [16,17]. A probabilistic control strategy called multi-model LQR was applied by Kolter to slide a vehicle into a parking spot [18,19]. Velenis et al. reproduced in simulation a trail braking maneuver using nonlinear optimization. The nonlinear program was formulated to achieve maximum corner exit speed or minimal cornering time, using vehicle model with suspension dynamics as constraints [4,20].

The common strategies to control transient drift maneuvers are

* Corresponding author.
E-mail addresses: z625451327@163.com (F. Zhang), likq@tsinghua.edu.cn (K. Li).

https://doi.org/10.1016/j.mechatronics.2018.05.009
Received 31 December 2016; Received in revised form 11 May 2018; Accepted 17 May 2018
Available online 27 June 2018

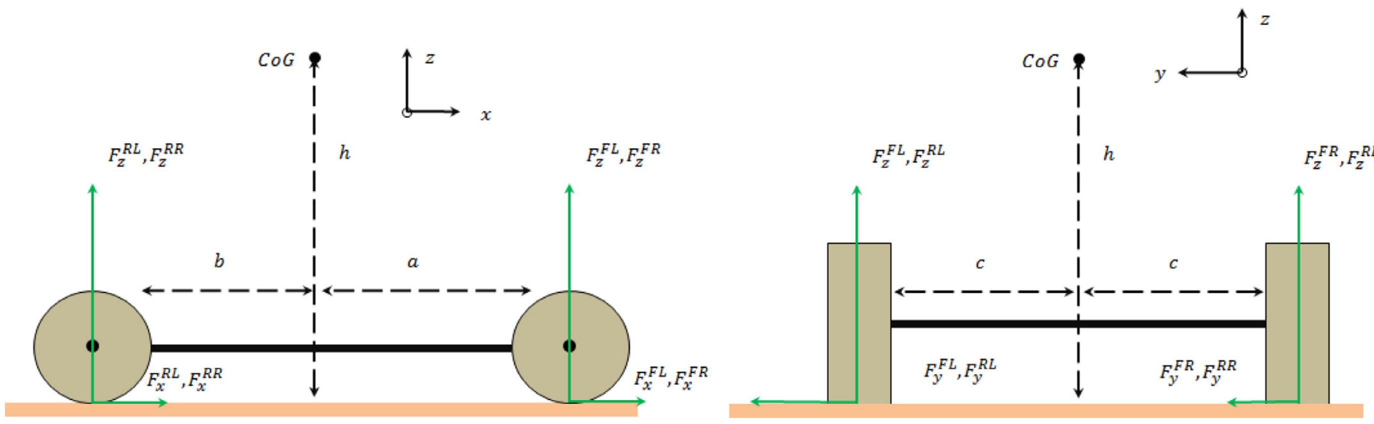

Fig. 1. x-z plane and y-z plane of the vehicle.

based on nonlinear control methods that use a high dimensional vehicle model. These models require numerous parameters, many of which are difficult to estimate, as well as significant computational resources, which makes them unsuitable for fast real-time implementation. In this paper, the authors extend previous work [22] and present a framework to plan and track a drift trajectory. Specially, the main contributions of this paper are:

- A hybrid path planning algorithm to generate a reference trajectory for drift. The planner divides the path horizon into three different types of regions, finds a path for each region using different planning algorithms (Rapidly-exploring Random Trees, rule-based sampling, Proportional Integral control) with different vehicle models, and then concatenates the solutions of each to construct the reference trajectory.
- A mixed open-loop and closed-loop control technique based on the standard bicycle model with linear tire model to track the drift trajectory, which is experimentally validated through a RC platform.

The remainder of this paper is organized as follows. Two kinds of vehicle models and tire models are discussed and compared in Section 2. Then, a hybrid rapidly-exploring random trees (RRT) and rule-based path planning algorithm is presented in Section 3. Section 4 summarizes mixed open-loop and closed-loop control strategy. Simulation and experimental results are presented in Section 5. Concluding remarks are given in Section 6.

2. System model

This section discusses vehicle models for the path planner and controller. Ideally, a single model would be used through the entire architecture, but conventional models, like the four-wheel model, begin to break down as the vehicle enters a drift state. The following subsections describe the low and high fidelity vehicle models that are used in the path planner and controller.

2.1. Four-wheel vehicle model

We use a planar four-wheel vehicle model from Falcone [23] to capture the vehicle dynamics, which models the vehicle as a single rigid body with forces acting at each of the wheels. The tire forces are modeled using the Pacejka tire model, which is a semi-empirical model similar in mathematical structure to physics-based models, which is a semi-empirical model based on fitting a curve to experimental data. The four-wheel vehicle model and Pacejka tire model describe the planar motion of the vehicle at the center of mass. The vehicle state and input are $z = [U_x, U_y, r, X, Y, \psi, \omega_{FL}, \omega_{FR}, \omega_{RL}, \omega_{RR},]$ and $u = [\delta, F_x^{RL}, F_x^{RR}]$, respectively. U_x, U_y and r are the vehicle's longitudinal speed, lateral speed and yaw rate in the body-fixed frame, and X, Y, ψ describe the position and yaw angle of the vehicle in earth-fixed frame. The four variables ω_{ij} are the wheel angular speed in tire-fixed frame, where the first subscript $i \in \{F, R\}$ indicates either the front or rear axle, and the second subscript $j \in \{L, R\}$ indicates either the right or left side. δ and F_x are the steering angle and rear drive force, respectively. For sake of brevity, we do not present the full set of equations from the models, but instead discuss only modifications to the model that take weight transfer into account.

Weight transfer means that the normal force of each tire (i.e. force in the vertical direction $F_z^{i,j}$) can change over time, especially when the yaw rate and acceleration are large. By assuming the vertical acceleration is zero and all rotations occur about the center of mass, we apply the following force constraints

$$F_z^{FL} + F_z^{FR} + F_z^{RL} + F_z^{RR} = mg \tag{1}$$

$$F_z^{FL} + F_z^{RR} = F_z^{FR} + F_z^{RL}. \tag{2}$$

For each force acting on the wheel, we use the notation $(\cdot)_k^{i,j}$, where i, j indicates the wheel, and $k \in \{x, y, z\}$ indicates the directional component of the force in the body frame of the vehicle. The $x - z$ and $y - z$ planes of vehicle are shown in the Fig. 1. We also apply balance of angular momentum equations about the center of mass

$$(F_z^{FL} + F_z^{FR})a + (F_x^{FL} + F_x^{FR} + F_x^{RL} + F_x^{RR})h = (F_z^{RL} + F_z^{RR})b \tag{3}$$

$$(F_z^{FL} + F_z^{RL})c + (F_y^{FL} + F_y^{FR} + F_y^{RL} + F_y^{RR})h = (F_z^{FR} + F_z^{RR})c \tag{4}$$

where m is the mass of the vehicle, g is the acceleration due to gravity, a, b are the distances from the center of gravity (CoG) to the front and rear axles, respectively. c is the distance from vehicle longitudinal axis to the wheels and h is the distance from the CoG to the ground.

The dynamics of the system are compactly expressed as

$$\dot{z}(t) = f^{4w}(z(t), u(t)). \tag{5}$$

where the superscript $4w$ indicates the four-wheel model.

2.2. Model accuracy

The four-wheel model from the previous section accurately describes the motion of the vehicle under typical driving conditions (i.e. non-extreme maneuvers), when the slip angle is small. This model begins to break down, however, once the slip angle grows to large values. To illustrate this, we compare the output of the four-wheel model and a high-fidelity vehicle model from CarSim. Both models use the same system parameters and the same input sequence that cause a large slip angle to emerge. The resulting trajectories are shown in Fig. 2a.

From the resulting trajectories, we observe that the four-wheel model and the CarSim model are nearly identical as the vehicle drives

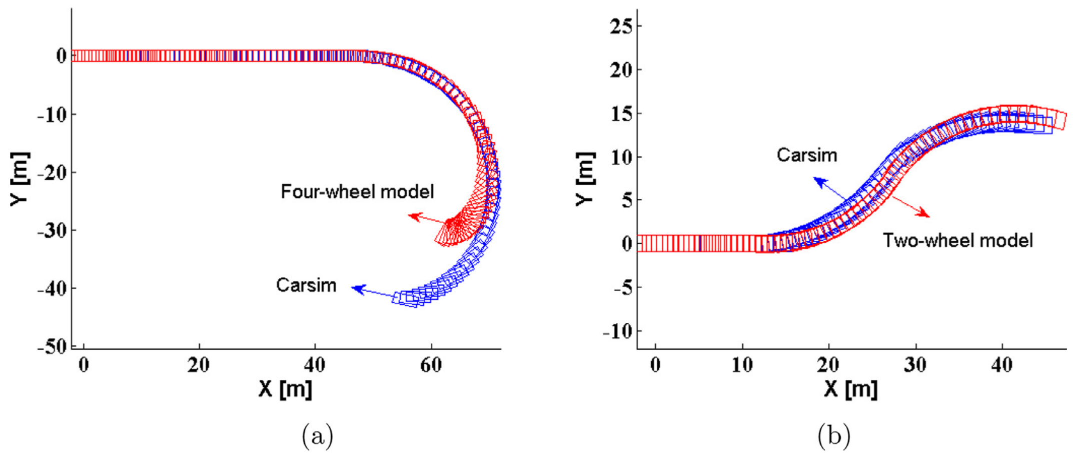

Fig. 2. (a) The four-wheel model accurately tracks the CarSim output for low-side slip motion, but then diverges for large slip angle motion (b) The two-wheel model accurately tracks the CarSim output for low-side slip motion.

straight, but then as the vehicle makes the turn and the tires begin to saturate, the trajectories begin to diverge significantly. This result indicates that the four-wheel model cannot accurately capture the dynamics of high side slip maneuvers. This motivates the use of two models: a conventional low order model for non-extreme maneuvers (i.e. low side-slip angle), and a high fidelity one for extreme maneuvers.

2.3. Bicycle model

For low side-slip angle maneuvers, we use the bicycle model, which simplifies the four-wheel model by lumping the front two tires together and the rear two tires together. This is because the bicycle model can capture the model features in low side-slip region. We compare the output of the two-wheel model and the Carsim model using the same system parameters and input sequence. The resulting trajectories in Fig. 2b are similar. This model selection reduces system complexity and eases parts of the design for the path planner and controller. The state and input vectors for the bicycle model are $z = [U_x, U_y, r, X, Y, \psi]$ and $u = [\delta, F_x^R]$, respectively. The equations of motion are

$$\dot{U}_x = \frac{1}{m}\sum F_x + U_y r = \frac{1}{m}F_x^R + U_y r \tag{6}$$

$$\dot{U}_y = \frac{1}{m}\sum F_y - U_x r = \frac{1}{m}(F_y^F + F_y^R) - U_x r \tag{7}$$

$$\dot{r} = \frac{1}{I_z}\sum M_z = \frac{1}{I_z}(aF_y^F - bF_y^R) \tag{8}$$

$$\dot{X} = U_x \cos\psi - U_y \sin\psi \tag{9}$$

$$\dot{Y} = U_x \sin\psi + U_y \cos\psi \tag{10}$$

$$\dot{\psi} = r \tag{11}$$

For small slip angles, the tire forces operate in the linear region, modeled as

$$F_y^F = C_s^F s^F \tag{12}$$

$$F_y^R = C_s^R s^R \tag{13}$$

where C_s^i is the cornering stiffness $(i = F, R)$. s^F and s^R are side-slip angles of front and rear wheels, which are given by $s^F = \delta - (U_y + ar)/U_x$ and $s^R = (br - U_y)/U_x$. The vehicle dynamics are compactly expressed as

$$\dot{z}(t) = f^{2w}(z(t), u(t)) \tag{14}$$

3. Path planning

Drift path planning is a difficult problem to solve due to the nonlinear nature of the system dynamics and the complexity of existing solution methods. We can significantly simplify the problem complexity, however, by partitioning the track into regions based on the type of maneuver the driver executes, and then solve a smaller path planning problem for each partition. Additionally, for the partition in which the driver initiates drift, we note that drift maneuvers often result from a simple combination of control inputs. With this observation in mind, we devise a path planning method to find a feasible drift trajectory.

For the race track segment in Fig. 3, the plan planner module aims to generate a sequence of states and inputs, referred as the reference trajectory, such that vehicle remains within the track boundaries and also drifts around the corner. The path planner divides the path horizon into three types of regions and solves smaller planning sub-problems. The planner then concatenates the solution path from each sub-problem to construct the reference trajectory. The region types are the ‘free’ (white) $\mathscr{R}_{\text{free}}$, ‘drift’ (purple) $\mathscr{R}_{\text{drift}}$, and ‘transit’ region (green) $\mathscr{R}_{\text{transit}}$.

In the free region, the planner applies an RRT algorithm [24] using a bicycle model to find a path. In the drift region, the planner applies a rule-based method with a high-fidelity vehicle model to search for a path. The transit region connects the two regions using a Proportional Integral (PI) controller. We designate track segments with a corner (purple) as drift regions since rally racers typically drift along the those corners. The segments before cornering are defined as free region. The transit region is defined to connect free region and drift region.

3.1. Free region

For low side-slip segments of the track in Fig. 3, the path planner uses a modified RRT in Table 1 to find a feasible trajectory. At each iteration, the RRT algorithm first generates a sample position, and then finds the nearest node z_{near} within the tree in terms of Euclidean distance. Next, the RRT algorithm samples a random input u and then propagates the dynamics forward one-time step, using the bicycle model, starting at z_{near} and resulting in node z_{new}. The sampled input accounts for input rate constraints by constraining the sampling domain to be around the previously applied input, which limits the input magnitude and ensures a smooth path. After propagating the node forward, the algorithm ensures the vehicle remains within the track boundary, and also checks for violations of low slip-angle assumptions. We assume no obstacles inside the track. As the bicycle model only holds in low side-slip angle conditions, the lateral acceleration of the new node z_{new} must satisfy the following conditions before it is added to

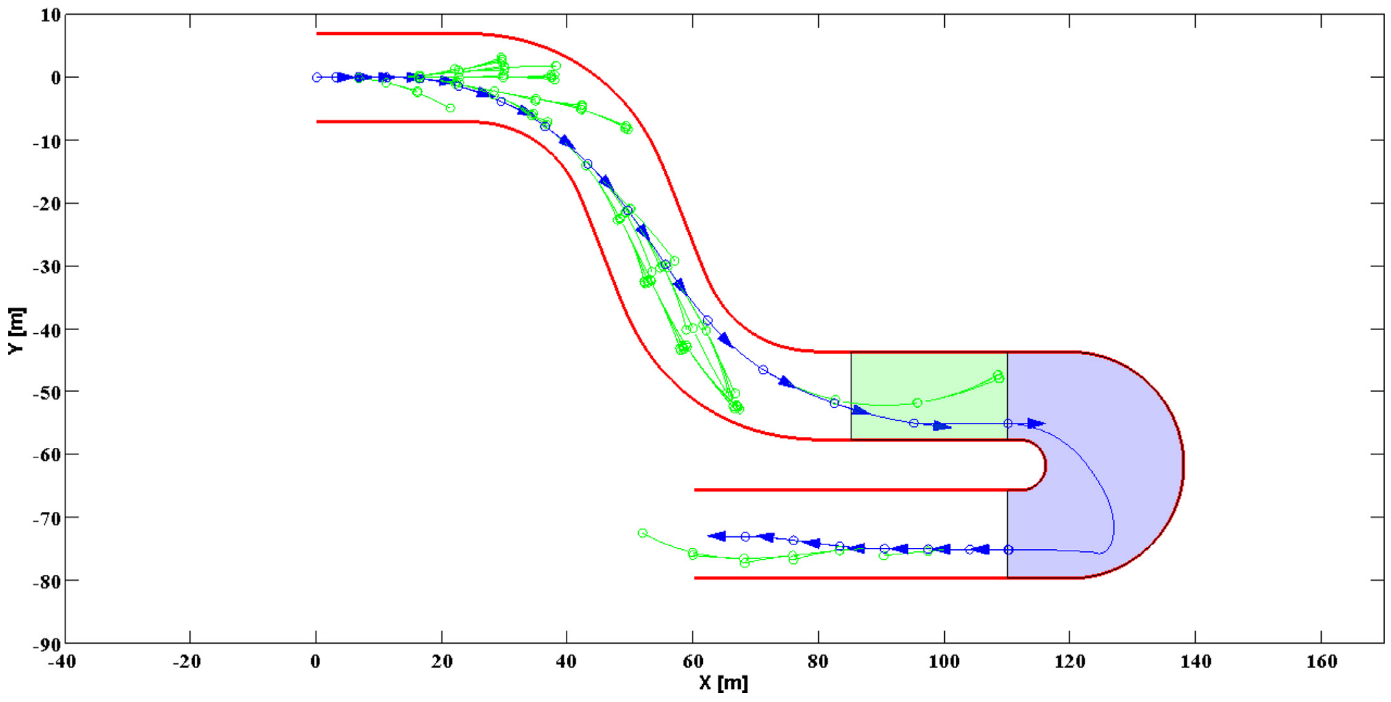

Fig. 3. The path planner divides the path horizon into three types of regions : free (white), drift (purple), transit (green). The path planning algorithm returns the blue trajectory. (For interpretation of the references to colour in this figure legend, the reader is referred to the web version of this article.)

Table 1
Pseudocode of the modified RRT planning.

Input:
Initial vehicle state z_{ini}, maximum number of nodes in the tree K, admissible inputs set $\mathscr{U}$, vehicle model $\dot{z}(t) = f^{2w}(z(t), u(t))$, incremental time Δt
Output: RRT graph G

```
1: $G$.initialize($z_{\text{ini}}$)
2: for $i = 1$ to $K$
3:     $X_{\text{rand}}, Y_{\text{rand}} \leftarrow$ GetRandomSample()
4:     $z_{\text{near}} \leftarrow$ FindNearestNode($X_{\text{rand}}, Y_{\text{rand}}, G$)
5:     $u \leftarrow$ GetRandomInput($\mathscr{U}$)
6:     $z_{\text{new}} \leftarrow$ Forward-model($z_{\text{near}}, u, f^{2w}$)
7:     if IsCollisionFree($z_{\text{new}}$) and IsLatAccValid($z_{\text{new}}$)
8:         $G$.add_node($z_{\text{new}}$)
9:         $G$.add_edge($z_{\text{near}} \rightarrow z_{\text{new}}, u$)
10:        if $z_{\text{new}} \in \mathscr{R}_{\text{transit}}$
11:            Exit SUCCESS
12:        end if
13:        if $i = =K$
14:            Exit FAILURE
15:        end if
16:    end if
17: end for
```

the tree.

$$\frac{a_y}{a_y^{\max}} \leq 0.3 \tag{15}$$

$$a_y = \frac{U_x^2}{R_s} = \frac{U_x^2 \delta}{a+b} \tag{16}$$

$$a_y^{\max} = \sqrt{(\mu g)^2 - a_x^2} \tag{17}$$

where $a_y^{\max}$ is maximum available lateral acceleration, R_s is the vehicle's turning radius, and μ is the tire-road friction coefficient. The algorithm terminates once a state is produced that is inside the transit region.

3.2. Transit region

In the transit region, the path planner applies two PI controllers to connect the final state of the free region and the initial state of the drift region. Both controller activate once the vehicle enters the transit region. The steering PI controller reduces the error between current yaw angle and initial yaw angle in the drift region. The rear-drive torque PI controller reduces the error between the current longitudinal speed and the initial one at the start of the drift region. The gains of controllers are tuned to guarantee fast converge and small overshoot. The planner in the transit region also uses a high fidelity vehicle model.

3.3. Drift region

Next comes the drift region. As mentioned above, drift maneuvers result from simple combinations of control inputs, which can be easily parameterized. The key idea of drift maneuvers is to saturate the rear tires through either a forward torque or a braking torque. Intuitively, tire saturation from a large rear longitudinal force reduces the maximum available lateral force that friction can provide, allowing the vehicle to easily enter a drift state. The drift maneuver is described by three basic phases:

Turn-in phase: The driver turns in the corner and simultaneously applies a large positive rear-drive torque. The vehicle rotates about its vertical axis and starts to slide.

Counter-steering phase: The driver counter steers and simultaneously decreases the rear-drive torque, which is necessary to prevent the vehicle from spinning out.

Exit phase: The driver stabilizes the vehicle, reducing the side-slip angle and de-saturating tire forces.

We use a rule-based algorithm to obtain an input sequence for steering and torque that causes the vehicle to enter drift. Each input sequence is a piece-wise constant function characterized by a set of parameters for time and magnitude. The vehicle model inside the algorithm is a high-fidelity Carsim model for simulation and a 1/10-scale RC car for experimentation. The shape of each input sequence is shown in Fig. 4. The parameters are t_{turn}, δ_{turn}, F_{turn} for the time duration,

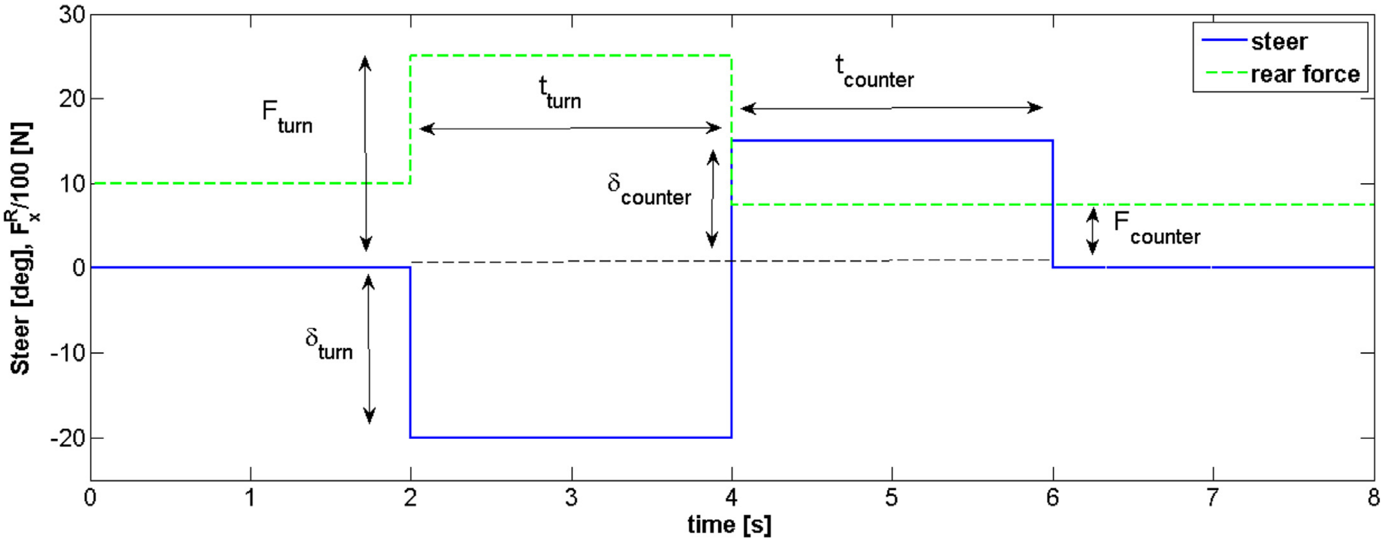

Fig. 4. The input profiles for the steering and rear tire force during drift cornering are often simple combinations of step functions that can be easily parameterized.

steering angle, and rear force of the turn-in phase, respectively; t_{counter}, δ_{counter}, F_{counter} for the time duration, steering angle, and rear force of counter-steering phase, respectively.

The proposed rule-based algorithm is outlined in Table 2. The sample space for each parameter is obtained from observing expert drivers (e.g. at which moment they counter steer, engage the gas, etc.). The demonstration allows us to construct sample space that are more likely to produce a feasible drift trajectory. Inside the algorithm, the planner first uniformly samples each parameter in sample space to generate an input sequence $\hat{u}$. Next, we run an experiment with either a high fidelity vehicle model or an actual vehicle, using the sampled input sequence. The algorithm terminates if the trajectory stays within the track, and also if the final yaw angle is near a target yaw angle ψ_{target} within some tolerance e_ψ. The target angle ψ_{target} is tangent center line path at the end of the drift region.

4. Control law

The proposed controller attempts to track the reference drift trajectory from Section 3 by using a mixed open-loop and closed-loop scheme. As mentioned earlier, it is difficult to track transient drift using a model-based controller due to the complex dynamics that arises from tire saturation and high side-slip angles. Over a short period of time, however, it is observed that a fixed open-loop input sequence produces a repeatable response [18]. With this observation in mind, we design a mixed open-loop and closed-loop controller. In general, the controller switches from closed-loop to open-loop when the closed-loop controller does not track the reference trajectory well.

Table 2
Pseudocode of the proposed rule-based planning.

Input: Initial vehicle states $\hat{z}_{\text{ini}}$, maximum number of iteration $\hat{K}$, sample space, vehicle model (Carsim model for simulation, RC car for experimentation)
Output: The values of t_{turn}, δ_{turn}, F_{turn}, t_{counter}, δ_{counter}, F_{counter}

```
1: for $i = 1$ to $\hat{K}$
2:     $\hat{u} \leftarrow$ GenerateInputSequence()
3:     $(\hat{z}_{\text{ini}} \rightarrow \hat{z}_{\text{end}}) \leftarrow$ ConductExperiment($\hat{z}_{\text{ini}}, \hat{u}$)
4:     if IsCollisionFree($\hat{z}_{\text{ini}} \rightarrow \hat{z}_{\text{end}}$)
5:         if $\|\psi_{\text{end}} - \psi_{\text{target}}\| < e_\psi$
6:             save $\hat{u}$ as $u^{\text{drift}}$
7:             Exit SUCCESS
8:         end if
9:         if $i == \hat{K}$
10:            Exit FAILURE
11:        end if
12:    end if
13: end for
```

The closed-loop part of mixed controller is based on LQR approach. We compute a sequence of LQR feedback control policies for each state and input pair from the reference trajectory z^{ref} and reference input u^{ref} in Section 3.

First, we analytically linearize the vehicle model f^{2w} with matrices $A = \mathrm{d}f^{2w}/\mathrm{d}z$ and $B = \mathrm{d}f^{2w}/\mathrm{d}u$. Next, we numerically evaluate the matrices A and B at each reference state and input pair ($z^{\text{ref}}(i)$, $u^{\text{ref}}(i)$), where i indexes a single state/input pair.

$$A(i) = \left.\frac{\mathrm{d}f^{2w}}{\mathrm{d}z}\right|_{\substack{z=z^{\text{ref}}(i)\\u=u^{\text{ref}}(i)}} \qquad B(i) = \left.\frac{\mathrm{d}f^{2w}}{\mathrm{d}u}\right|_{\substack{z=z^{\text{ref}}(i)\\u=u^{\text{ref}}(i)}} \tag{18}$$

For each reference pair, the error dynamics of the system $f^{\text{err}}$are given by

$$\Delta\dot{z}(i) = A(i)\Delta z(i) + B(i)\Delta u(i) \tag{19}$$

where $\Delta z(i) = z - z^{\text{ref}}(i)$ and $\Delta u(i) = u - u^{\text{ref}}(i)$.

Then we compute the closed-loop control policy based on an LQR with the following quadratic cost function:

$$J = \int_{t=0}^{\infty} (\Delta z^T Q \Delta z + \Delta u^T R \Delta u) dt \tag{20}$$

where $Q = Q^T > 0$, $R = R^T > 0$. The control input Δu that minimizes the cost function is given by

$$\Delta u = -R^{-1}B^T P \Delta z = -K^{\text{LQR}}\Delta z \tag{21}$$

where $P = P^T > 0$, which can be obtained by solving algebraic Riccati equation. The closed-loop control policy is given by equation (22). For *each* reference state and input pair, we now have a feedback matrix $K^{\text{LQR}}(i)$.

$$u^{\text{cl}}(i) = u^{\text{ref}}(i) + \Delta u(i) \tag{22}$$

4.1. *Online control design*

The online controller performs two actions: (a) search for the nearest reference trajectory point, and (b) apply either an open-loop (the reference input)or closed-loop input command. The structure is shown in Fig. 5.

First, at each time step, we find the nearest reference position (X^{ref}, Y^{ref}, ψ^{ref}) by performing the optimization routine below

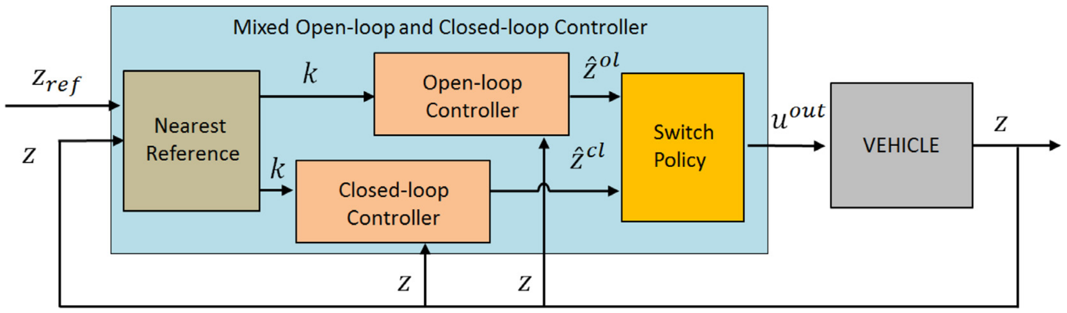

Fig. 5. The online mixed control scheme evaluates both open-loop and close-loop polices and then selects the optimal one according to the cost function in the switch policy block.

$$\min_{k} [\omega_X, \omega_Y, \omega_\psi]^T \begin{bmatrix} \|X_t - X^{\text{ref}}(k)\| \\ \|Y_t - Y^{\text{ref}}(k)\| \\ \|\psi_t - \psi^{\text{ref}}(k)\| \end{bmatrix} \tag{23}$$

where (X_t, Y_t, ψ_t) is the current position estimate and $(\omega_X, \omega_Y, \omega_\psi)$ is a set of weight parameters.

Next the online controller applies either a closed-loop or open-loop input. To decide among the two, we propagate the vehicle model f^{2w} forward in time for n steps starting from current state z^t. The controller propagates the dynamic twice, once using the open-loop inputs and once using the closed-loop policy, as shown below

$$\hat{z}^{\text{mode}}(i+1) = \hat{z}^{\text{mode}}(i) + T_s f^{2w}(\hat{z}^{\text{mode}}(i), u^{\text{mode}}(i)) \tag{24a}$$

$$u^{\text{mode}}(i) = \begin{cases} u^{\text{ref}}(i+k) + K^{\text{LQR}}(i+k)(\hat{z}^{\text{cl}}(i) - z^{\text{ref}}(i+k)) & : \text{CL} \\ u^{\text{ref}}(i+k) & : \text{OL} \end{cases} \tag{24b}$$

$$\hat{z}^{\text{mode}}(0) = z^t \tag{24c}$$

where mode = {ol, cl}. $\hat{z}^{\text{mode}}(i)$ is the predicted states by applying closed-loop or open-loop control. The predicted time horizon is 2 s. The controller selects the control command u^{out} that results in a smaller cumulative tracking error, which is expressed as below

$$J^{\text{mode}} = \sum_{i=0}^{i=n} (\hat{z}^{\text{mode}}(i) - z^{\text{ref}}(i+k))^T \hat{Q} (\hat{z}^{\text{mode}}(i) - z^{\text{ref}}(i+k)) \tag{25}$$

The underlying motivation of the proposed method is that over a short period of time, a fixed open-loop input sequence produces a repeatable response, even for complex maneuvers like transient drift. In general, the controller selects closed-loop command in well-modeled regions and selects open-loop maneuvers in poorly-modeled region (i.e. drifting).

5. Results and discussion

5.1. Simulation result

We first validated the proposed planning and control strategy in simulation using CarSim. We selected an A-class rear-drive vehicle as the plant since the software suite does not provide any vehicle models at the scale of an RC car. The parameters of the vehicle and tires while stationary are summarized in Table 3.

The simulation results of the proposed algorithm are shown in Fig. 6, using a 2 degree offset for steering angle as a measure of model mismatch. In Fig. 6, the black dash line represents the track boundary, the blue path represents the reference trajectory, and the red path outlines the trajectory using the proposed control algorithm. The green line and cyan line show the path under pure open-loop and pure closed-loop command with a steering angle offset. The simulation results suggest that the vehicle cannot track the designed trajectory closely by following the open-loop command because of model mismatch. By following only closed-loop commands, the vehicle closely tracks the reference trajectory in the low side-slip region, since the vehicle dynamics are well-modeled, but begins to diverge from the reference trajectory in the drift region where the tire forces saturate. The proposed strategy tracks the reference trajectory more closely than either method because of the switching policy. When the tracking error of predicted states under closed-loop control is high, the controller no longer relies on the LQR policy and instead applies open-loop inputs.

Table 3
Vehicle parameters.

Parameter	Carsim	RC-car	Parameter	Carsim	RC-car
m [kg]	1830	1.95	b [m]	1.65	0.125
I_z [kg·m^2]	3287	0.24	C_s^F [N/rad]	36000	1.76
a [m]	1.4	0.125	C_s^R [N/rad]	36000	1.76

5.2. Experimental result

The authors used a 1/10 scale rear-drive RC vehicle platform, which is a low-cost development platform for autonomous driving to achieve complex maneuvers such as drifting and lane changing[1]. The model identification and state estimation methods are based on previous work from [12]. The parameters of RC car and tires are summarized in Table 3. After obtaining the reference trajectory as shown in Section 3, we ran two sets of experiments, one using the proposed control algorithm, the other using pure open-loop control command. The blue path in Fig. 7 denotes the reference trajectories, and the green and red paths denote the experimental results using open-loop control and proposed control method, respectively. The snapshot of experiment is shown in Fig. 8. The experimental results are consistent with those of the simulations. For experimentation, the performance of pure open-loop control is poor even in the straight segments due to disturbances in the environment. The proposed mixed open-loop and closed-loop strategy tracks the reference path more closely than either pure open-loop or pure-closed loop strategies. The vehicle follows the reference trajectory well in the straight segment. As the vehicle drifts around the corner, the difference between actual trajectory and reference trajectory gets larger. Because the open-loop control is applied in the cornering segment. Then the vehicle recovers to the reference path again in the second straight segment as the LQR is applied. We tested the repeatability of the proposed control algorithm by tracking the same reference trajectory multiple times. The experimental result shows that the proposed control algorithm is applicable in practice by using only low-cost sensors. The limitation of proposed method is that the duration of the drifting must be short, otherwise, the vehicle cannot recover from the drifting reference errors in well-modeled region after exiting the drift region.

[1] More information at the project site barc-project.com.

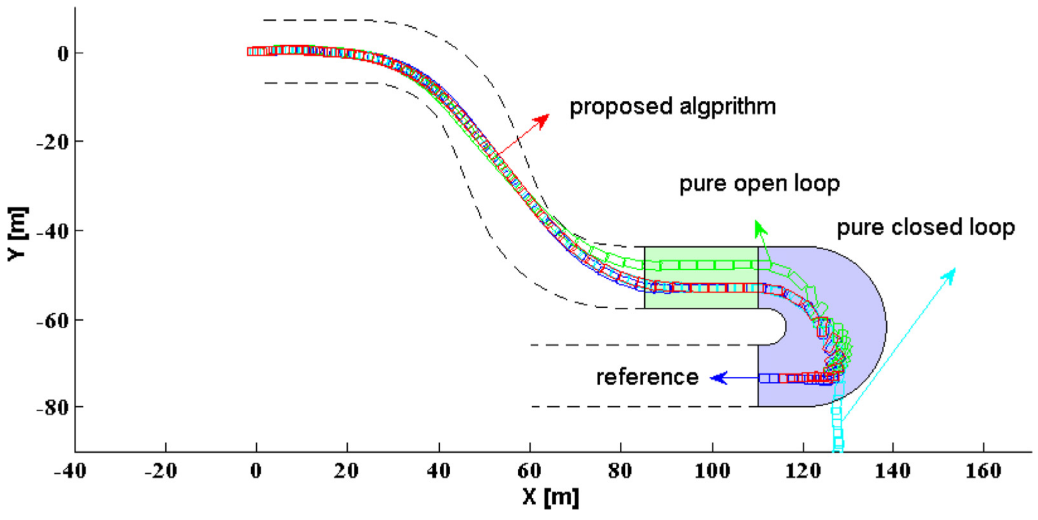

Fig. 6. The plot above illustrates the tracking performance in simulation of the proposed algorithm (red), against a pure open-loop strategy (green) and pure closed-loop strategy (light blue). The proposed algorithm most closely tracks the reference trajectory (blue). (For interpretation of the references to colour in this figure legend, the reader is referred to the web version of this article.)

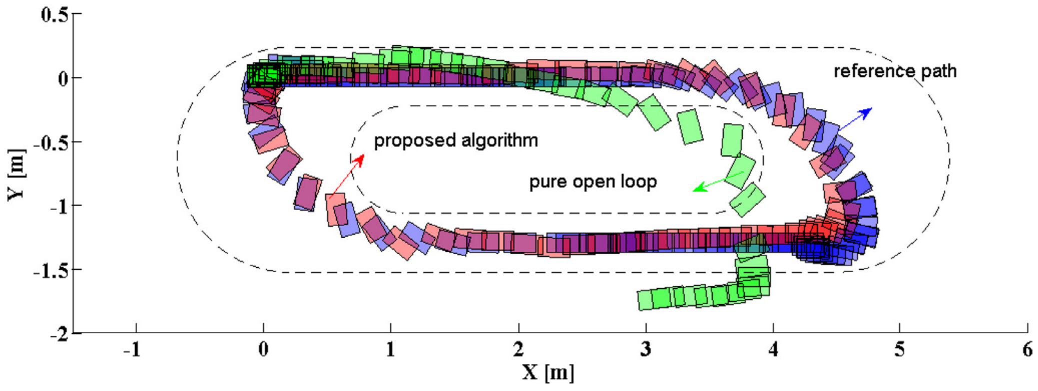

Fig. 7. The plot above illustrates the tracking performance from experimentation of the proposed algorithm (red), against a pure open-loop strategy (green). The proposed algorithm most closely tracks the reference trajectory (blue). (For interpretation of the references to colour in this figure legend, the reader is referred to the web version of this article.)

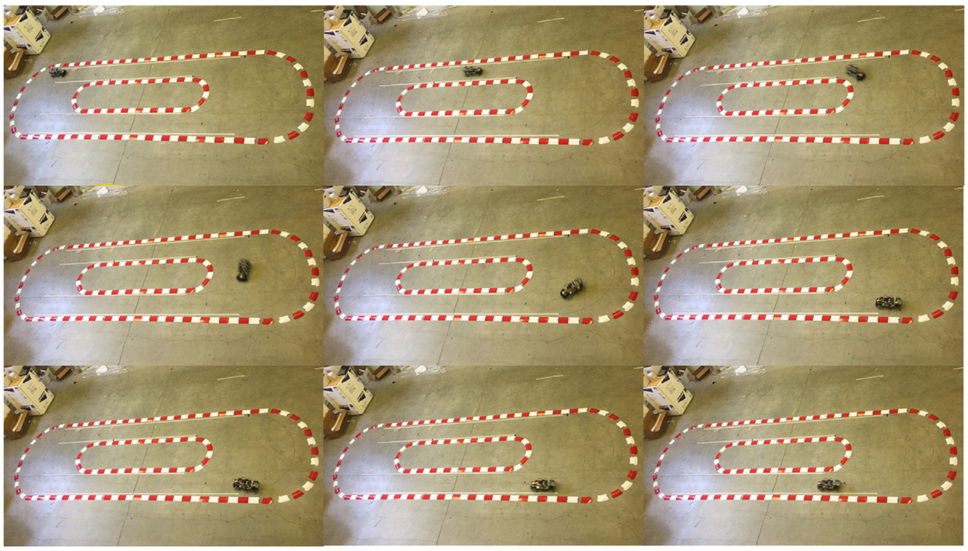

Fig. 8. The sequences of snap shots illustrate that the RC successfully tracks the drift cornering trajectory.

6. Conclusion

In this paper, the authors propose a path planning strategy and control scheme to plan and track a reference drift trajectory along a sharp bend in a track. The path planner combines the RRT method and a rule-based method to plan a trajectory for both low side-slip and high side-slip motion. The controller then applies a mixed open-loop and closed-loop scheme with three-state vehicle model to track the reference trajectory. We validate the proposed planning and control algorithms in simulation using a high-fidelity model in Simulink/Carsim, and through experimentation using a 1/10 scale RC vehicle. The proposed path planner and controller constitute a complete framework to control a vehicle to track a reference drift trajectory. Future work includes achieving better drift performance, like reducing tap time or extending the drifting duration.

Acknowledgements

This study is partially supported by NSF China with 51575293 and 51622504, National Key R&D Program in China with 2016YFB0100906, and International Sci&Tech Cooperation Program of China under 2016YFE0102200.

Supplementary material

Supplementary material associated with this article can be found, in the online version, at doi:10.1016/j.mechatronics.2018.05.009.

References

[1] Osborn RP, Shim T. Independent control of all-wheel-drive torque distribution. Vehic Syst Dyn 2006;44(7):529–46.
[2] Yamakawa J, Kojima A, Watanabe K. A method of torque control for independent wheel drive vehicles on rough terrain. J Terramech 2007;44(5):371–81.
[3] Velenis E, Tsiotras P, Lu J. Modeling aggressive maneuvers on loose surfaces: the cases of trail-braking and pendulum-turn. Control conference (ECC), 2007 European. IEEE; 2007. p. 1233–40.
[4] Velenis E, Tsiotras P. Optimal velocity profile generation for given acceleration limits: The half-car model case,g. 2005 IEEE international symposium on industrial electronics. 2005. p. 355–60.
[5] Ono E, Hosoe S, Tuan HD, Doi S. Bifurcation in vehicle dynamics and robust front wheel steering control. IEEE Trans Control Syst Technol 1998;6(3):412–20.
[6] Voser C, Hindiyeh RY, Gerdes JC. Analysis and control of high sideslip manoeuvres. Vehic Syst Dyn 2010;48(S1):317–36.
[7] Velenis E, Frazzoli E, Tsiotras P. On steady-state cornering equilibria for wheeled vehicles with drift. Decision and control, 2009 held jointly with the 2009 28th Chinese control conference. CDC/CCC 2009. Proceedings of the 48th IEEE conference on. IEEE; 2009. p. 3545–50.
[8] Velenis E, Katzourakis D, Frazzoli E, Tsiotras P, Happee R. Steady-state drifting stabilization of rwd vehicles. Control Eng Pract 2011;19(11):1363–76.
[9] Hindiyeh RY, Gerdes JC. Design of a dynamic surface controller for vehicle sideslip angle during autonomous drifting. IFAC Proc Vol 2010;43(7):560–5.
[10] Velenis E, Frazzoli E, Tsiotras P. Steady-state cornering equilibria and stabilisation for a vehicle during extreme operating conditions. Int J Veh Autonom Syst 2010;8(2-4):217–41.
[11] Hindiyeh RY, Gerdes JC. A controller framework for autonomous drifting: Design, stability, and experimental validation. J Dyn Syst Measure Control 2014;136(5):051015.
[12] Gonzales J, Zhang F, Li K, Borrelli F. Autonomous drifting using onboard sensors. 13th international symposium on advanced vehicle control. AVEC; 2016.
[13] Cutler M, Walsh TJ, How JP. Reinforcement learning with multi-fidelity simulators. 2014 IEEE international conference on robotics and automation (ICRA). IEEE; 2014. p. 3888–95.
[14] Cutler M, Walsh TJ, How JP. Real-world reinforcement learning via multifidelity simulators. IEEE Trans Robot 2015;31(3):655–71.
[15] Cutler M, How JP. Autonomous drifting using simulation-aided reinforcement learning. 2016 IEEE international conference on robotics and automation (ICRA). IEEE; 2016. p. 5442–8.
[16] Chakraborty I, Tsiotras P, Lu J. Vehicle posture control through aggressive maneuvering for mitigation of t-bone collisions. 2011 50th IEEE conference on decision and control and European control conference. IEEE; 2011. p. 3264–9.
[17] Chakraborty I, Tsiotras P, Diaz RS. Time-optimal vehicle posture control to mitigate unavoidable collisions using conventional control inputs. 2013 American control conference. IEEE; 2013. p. 2165–70.
[18] Kolter JZ, Plagemann C, Jackson DT, Ng AY, Thrun S. A probabilistic approach to mixed open-loop and closed-loop control, with application to extreme autonomous driving. Robotics and automation (ICRA), 2010 IEEE international conference on. IEEE; 2010. p. 839–45.
[19] Kolter JZ. Learning and control with inaccurate models. Stanford University; 2010. Ph.D. thesis.
[20] Velenis E, Tsiotras P. Minimum time vs maximum exit velocity path optimization during cornering. 2005 IEEE international symposium on industrial electronics. 2005. p. 355–60.
[21] Velenis E, Tsiotras P, Lu J. Optimality properties and driver input parameterization for trail-braking cornering. Eur J Control 2008;14(4):308–20.
[22] Zhang F, Gonzales J, Li K, Borrelli F. Autonomous drift cornering with mixed open-loop and closed-loop control. IFAC Proc Vol 2017.
[23] Falcone P, Eric Tseng H, Borrelli F, Asgari J, Hrovat D. Mpc-based yaw and lateral stabilisation via active front steering and braking. Vehic Syst Dyn 2008;46(S1):611–28.
[24] LaValle SM. Rapidly-exploring random trees: a new tool for path planning. Tech rep 98-11. Computer Science Department, Iowa State University, 1998; 1998.