Autonomous Drift Cornering with Mixed Open-loop and Closed-loop Control

F.Zhang * **J. Gonzales** ** **K.Li** * **F. Borrelli** **

* *Department of Automotive Engineering Tsinghua University, Beijing, China (e-mail: zhangfang12@ mails.tsinghua.edu.cn likq@tsinghua.edu.cn).*
** *Department of Mechanical Engineering, University of California, Berkeley, USA (e-mail: jon.gonzales@berkeley.edu fborrelli@berkeley.edu)*

Abstract: Expert drivers have the skill to perform high side slip maneuvers, like drifting, during racing events to minimize lap time. Due to the complex dynamics of these maneuvers, transient drift is difficult to model and to control in autonomous vehicles.
In this paper, the authors present a mixed open-loop and closed-loop control strategy to perform a transient drift-corning trajectory. The reference trajectory is generated from a ruled-based algorithm. We validate the planning and control techniques in simulation using Simulink/Carsim, and experimentally using an open source, low cost 1/10 scale RC car.

Keywords: Autonomous vehicles, Vehicle dynamics, Rule-based path planning, Drift control

1. INTRODUCTION

High side slip cornering, or drifting, represents one of the most challenging maneuvers for expert drivers. In contrast to normal cornering, high-slip concerning involves large side slip angle β and near full saturation of the rear tires. The side slip angle β measures the angle between vehicle's direction of travel and longitudinal velocity vector, as shown in equation 1.

$$\beta = \tan^{-1} \frac{U_y}{U_x}. \tag{1}$$

Controlling drifting maneuvers has not been a focus for the majority of the automotive controls community. In fact, current driver assist controls systems like ABS and ESC try to prevent drifting conditions from ever arising. These systems are critical for user safety. In racing instead, drivers often bring the vehicle into this unstable state in order to reduce lap time, while still maintaining control of the vehicle. By studying drift, we can design controllers that not only respond quickly to unintentional drift conditions for the typical driver, but also exploit the drift dynamics to avoid collisions or minimize lap time for racing applications.
Drift maneuvers fall into one of two categories: sustained drift and transient drift. Sustained drift focuses on stabilizing the vehicle about an unstable equilibrium (e.g. steady state circular drift), while transient drift focuses on tracking a set of non-steady drift states (e.g. drift corning). For sustained drifting, different kinds of control techniques have been validated by various researchers. Hindiyeh and Gerdes (2014) utilized a variant of the dynamic surface control technique based on a nested-loop structure. Velenis et al. (2011) designed a combination of linear and backstep control scheme to stabilize rear-wheel-drive vehicles using a seven-state vehicle model with rear axes differentials. Cutler (2015) used reinforcement learning with a motion caption system to achieve sustained drift.
Work on transient drift has also emerged as a research topic for vehicle applications. Velenis and Tsiotras (2005) and Velenis et al. (2008) introduced a bicycle model with suspension dynamics and applied numerical optimization to analyze the drift cornering behavior. Their proposed algorithm achieves maximum corner exit velocity or equivalently minimal time. They validated the algorithm in simulation. Kolter et al. (2010) applied a mixed open-loop and closed-loop maneuver using a probabilistic method called multi-model LQR. Tavernini et al. (2014) utilized nonlinear optimal control theory to investigate the optimality of the handbrake cornering technique for a front wheel drive vehicle.
To the authors' knowledge, all experimental validation for drift control algorithms have used a motion capture system and/or a differential GPS system. While these systems provide very accurate measurements of position and velocity, many of these commercial sensors are expensive. This paper explores a mixed control technique to achieve drift cornering using a set of low-cost sensors and an open source RC platform.
The remainder of this paper is organized as follows. The vehicle and tire models are discussed in Section 2. Then, path planning and control algorithm for drift cornering are presented in Section 3 and Section 4. Model identification and state estimation are discussed in Section 5. Simulation and experimental results are summarized in Section 6 and Section 7, respectively. Concluding remarks are given in Section 8.

2. SYSTEM MODEL

We capture the vehicle dynamics using a six-state bicycle model with linear front and rear-wheel tire forces. The

state vector consists of longitudinal velocity U_x (body-fixed frame), lateral velocity U_y (body-fixed frame), yaw rate r, longitudinal position X (ground-fixed frame), lateral position Y (ground-fixed frame) and yaw angle ψ. The input vector consists of steering angle δ and rear longitudinal force F_{xR}. The effects of aerodynamic drag and rolling drag are not taken into consideration.

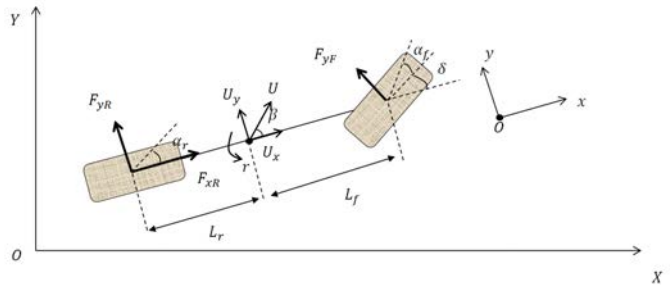

Fig. 1. Vehicle model schematic

This single-track rear-wheel-drive model is governed by the following set of differential equations.

$$\dot{U}_x = \frac{1}{m}\sum F_x = \frac{1}{m}F_{xR} \tag{2}$$

$$\dot{U}_y = \frac{1}{m}\sum F_y - U_x r = \frac{1}{m}(F_{yF} + F_{yR}) - U_x r \tag{3}$$

$$\dot{r} = \frac{1}{I_z}\sum M_z = \frac{1}{I_z}(L_f F_{yF} - L_r F_{yR}) \tag{4}$$

$$\dot{X} = U_x \cos\psi - U_y \sin\psi \tag{5}$$

$$\dot{Y} = U_x \sin\psi + U_y \cos\psi \tag{6}$$

$$\dot{\psi} = r \tag{7}$$

where F_{yF} and F_{yR} are the lateral forces on the front wheel and real wheel. The parameters m and I_z are the mass and moment of inertia about z-axis. The quantities F_x, F_y, M_z are the total longitudinal force, total lateral force, and total moment about the z-axis, respectively. L_f and L_r represent the distance from the front and rear axles to the CoG respectively, as shown in Figure 1.

For the tires, we use a linear model to estimate the lateral force. The following equations determine the lateral forces on each wheel.

$$F_{yF} = C_{\alpha f}\alpha_f \tag{8}$$

$$F_{yR} = C_{\alpha r}\alpha_r \tag{9}$$

where $C_{\alpha i}$ is the cornering stiffness of front or rear wheel and α_i is the side slip angle of wheels $(i = f, r)$.

$$\alpha_f = \delta - \frac{U_y + L_f r}{U_x} \tag{10}$$

$$\alpha_r = \frac{L_r r - U_y}{U_x}. \tag{11}$$

In state space form, the dynamic equations are expressed as shown below:

$$\dot{\mathbf{z}} = f(\mathbf{z}, \mathbf{u}) \tag{12}$$

where $\mathbf{z} = [U_x, U_y, r, X, Y, \psi]$, and $\mathbf{u} = [\delta, F_{xR}]$.

3. PATH PLANNING

Drift corning behavior is characterized by a large lateral acceleration, a large side slip angle, and tire force saturation. While methods have been proposed to analyze and control drift under steady state conditions (Hindiyeh and Gerdes (2014)), it is difficult to model the vehicle behavior under transient conditions. Model-based path planning algorithms often cannot easily plan such maneuvers. Interestingly, by observing expert drivers, we find that drifting maneuvers often result from simple combinations of control inputs that can be easily parametrized.

There are many kinds of maneuvers that expert drivers apply in real drift cornering, such as 'power oversteer' or 'clutch kick'. The key idea of all these maneuvers is to saturate the rear wheels with a large torque input or braking. Tire saturation from a large rear longitudinal input reduces the maximum possible lateral force that friction can provide, which allows the vehicle to easily enter a drift state.

During a drift cornering process, the driver initially drives straight before the track corner, then he turns the corner and applies a large rear torque. At this point, the vehicle begins to slide, and the driver then both counter-steers and decreases the rear torque. Without this counter steer, the vehicle would spin out. As the driver exits the corner, the vehicle recovers to a stable condition, with a small side slip and non-saturated tires. The general sequence of drift cornering inputs is in illustrated in Figure 2.

The controller proposed in this paper tracks a reference drift cornering trajectory. To obtain this trajectory, we apply input sequences generated by a rule-based algorithm to a high-fidelity Carsim model for simulation, and then to a 1/10-scale RC car for experimentation. The authors use the rule-based algorithm to search for an input sequence that result in transient drift for both simulation and experimentation.

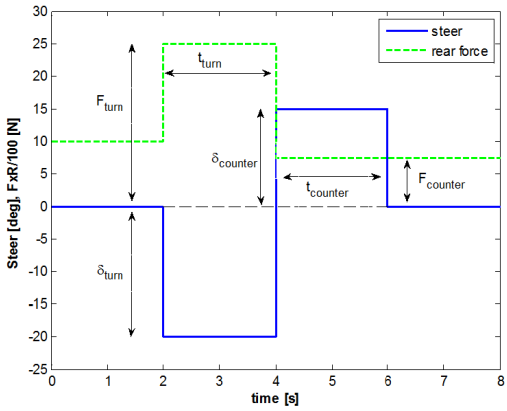

Fig. 2. Parametrized control sequence for drift cornering

The generated input sequences for δ and F_{xR} are parametrized by the following parameters. Sample input sequences are shown in Figure 2.

t_{turn}, time duration of Turning in phase;
δ_{turn}, steering angle of Turning in phase;
F_{turn}, rear force of Turning in phase;
t_{counter}, time duration of Counter-steering phase;
δ_{counter}, steering angle of Counter-steering phase;
F_{counter}, rear force of Counter-steering phase.

Table 1. Rule-based path planning algorithm

1 Define the initial vehicle state and track boundary
2 Sample each parameter from a uniform distribution
3 Conduct experiments using inputs based on the sampled parameter set
4 Check if the resulting trajectory is drifting inside the corner. If unsuccessful, return to step 2.

Table 2. Offline control design

1 Define the reference trajectory (from Section 3) $U_x^{\text{ref}}(i), U_y^{\text{ref}}(i), r^{\text{ref}}(i), X^{\text{ref}}(i), Y^{\text{ref}}(i), \psi^{\text{ref}}(i), \delta^{\text{ref}}(i), F_{xR}^{\text{ref}}(i)$
2 Linearize vehicle model along the designed reference trajectory
3 Define the feedforward-feedback control policy $\Delta\mathbf{u}(i) = -K(i)\Delta\mathbf{z}(i)$ $\mathbf{u}^{\text{cl}}(i) = \mathbf{u}^{\text{ref}}(i) + \Delta\mathbf{u}(i)$
4 Compute errors between predicted and reference states $\mathbf{e}(i) = \hat{\mathbf{z}}(i) - \mathbf{z}^{\text{ref}}(i)$

Table 3. Online control design

1 Find the nearest point in the designed trajectory
2 Propagate vehicle dynamics forward in time for both closed-loop and open-loop policies
3 Compare cost functions of predicted closed-loop and predicted open-loop trajectories
4 Select the control policy command with less cost

The rule-based algorithm uniformly samples each parameter to generate an input sequence. The sampling domain for each parameter is based on observing expert drivers. By doing this, we can reduce the size of the sample domain to quickly generate input sequences that result in drift cornering. These steps are outlined in Table 1.
During a simulation or experiment in which the vehicle drifts the corner successfully, we set the associated states and inputs as the reference trajectory. This reference trajectory ($\mathbf{z}^{\text{ref}}$, $\mathbf{u}^{\text{ref}}$) will guide the design of the feedback controller. For simulation, we run the path planning algorithm on a laptop using a high fidelity vehicle mode from CarSim.

4. CONTROL LAW

The proposed controller attempts to track a reference drift trajectory. As mentioned earlier, a model-based controller is difficult to design due to complex dynamics during transient drift. During a short period of time (relative to the time constant of the system), however, a fixed open-loop input sequence produces a repeatable response (Kolter et al. (2010)). With this observation in mind, we design a mixed open-loop and closed-loop controller. The general steps are outlined in Table 2 and Table 3, details are discussed next.

4.1 Offline control-design

The closed-loop controller is based on an LQR approach. We compute a sequence of LQR feedback control policies for each state and input pair from the reference trajectory ($\mathbf{z}^{\text{ref}}$, $\mathbf{u}^{\text{ref}}$) in Section 3. First, we analytically linearize the vehicle model in equations (2)-(11) with matrices $A = d\mathbf{f}/d\mathbf{z}$ and $B = d\mathbf{f}/d\mathbf{u}$. The analytic expressions for the entries of A, B are available in the technical report [1]. Next, we numerically evaluate the matrices A and B at each reference state and input pair ($\mathbf{z}^{\text{ref}}(i)$, $\mathbf{u}^{\text{ref}}(i)$), where i indexes a single state/input pair.

$$A(i) = \left.\frac{d\mathbf{f}}{d\mathbf{z}}\right|_{\substack{\mathbf{z}=\mathbf{z}^{\text{ref}}(i)\\ \mathbf{u}=\mathbf{u}^{\text{ref}}(i)}} \qquad B(i) = \left.\frac{d\mathbf{f}}{d\mathbf{u}}\right|_{\substack{\mathbf{z}=\mathbf{z}^{\text{ref}}(i)\\ \mathbf{u}=\mathbf{u}^{\text{ref}}(i)}} \tag{13}$$

For each each reference pair, the error dynamics of the system are given by

$$\Delta\dot{\mathbf{z}}(i) = A(i)\Delta\mathbf{z}(i) + B(i)\Delta\mathbf{u}(i) \tag{14}$$

where $\Delta\mathbf{z}(i) = \mathbf{z} - \mathbf{z}^{\text{ref}}(i)$ and $\Delta\mathbf{u}(i) = \mathbf{u} - \mathbf{u}^{\text{ref}}(i)$.
The closed-loop control policy is based an LQR with the following quadratic cost function:

$$J = \int_{t=0}^{\infty} (\Delta\mathbf{z}^T Q \Delta\mathbf{z} + \Delta\mathbf{u}^T R \Delta\mathbf{u}) dt \tag{15}$$

where $Q = Q^T \geq 0$, $R = R^T \geq 0$. The control input $\Delta\mathbf{u}$ that minimizes the cost function is given by

$$\Delta\mathbf{u} = -R^{-1}B^T P \Delta\mathbf{z} = -K(i)\Delta\mathbf{z} \tag{16}$$

where $P = P^T > 0$, which can be obtained by solving Riccati Equation. The closed-loop control policy is given by equation (17). For *each* reference state and input pair, we now have a feedback matrix $K(i)$ and

$$\mathbf{u}^{\text{cl}}(i) = \mathbf{u}^{\text{ref}}(i) + \Delta\mathbf{u}(i). \tag{17}$$

For the last part of the off-line design, we compute the state error between the vehicle model f in equations (2)-(7) and the actual plant [2]. This series of error states will be used in the online controller, discussed shortly. We first propagate the dynamics for the entire duration of the trajectory using the vehicle model in (2)-(7), as shown in equation (18), where T_s is the time step between consecutive state/input pairs.

$$\hat{\mathbf{z}}(i+1) = \mathbf{z}^{\text{ref}}(i) + T_s f(\mathbf{z}^{\text{ref}}(i), \mathbf{u}^{\text{ref}}(i)) \tag{18a}$$

$$\hat{\mathbf{z}}(0) = \mathbf{z}^{\text{ref}}(0) \tag{18b}$$

Then we calculate error $\mathbf{e}$ between the predicted state $\hat{\mathbf{z}}$ and the reference state $\mathbf{z}^{\text{ref}}$ as shown below

$$\mathbf{e}(i) = \hat{\mathbf{z}}(i) - \mathbf{z}^{\text{ref}}(i) \tag{19}$$

The entire offline control process is carried out on a laptop. Calculating the LQR matrix gain computations requires modest computational resources.

4.2 Online control-design

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

[1] Technical Report available barc-project.com/projects/
[2] The plant refers to either Carsim or the physical experiment, depending on which system was used to produce the reference trajectory

the first part, at each time step, the controller finds the 'nearest' reference position $(X^{\text{ref}}, Y^{\text{ref}}, \psi^{\text{ref}})$ by performing the optimization routine below

$$\underset{k}{\text{argmin}} \, [\omega_X, \omega_Y, \omega_\psi]^\top \begin{bmatrix} ||X_t - X^{\text{ref}}(k)|| \\ ||Y_t - Y^{\text{ref}}(k)|| \\ ||\psi_t - \psi^{\text{ref}}(k)|| \end{bmatrix} \tag{20}$$

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, the controller propagates the vehicle model (2)-(11) forward in time for n steps starting from the current state estimate $\mathbf{z}_t$. The controller propagates the dynamics twice, once using an open-loop input sequence and once using a closed-loop one, adding the state error terms at each time step, as shown below

$$\hat{\tilde{\mathbf{z}}}(i+1) = \hat{\tilde{\mathbf{z}}}(i) + T_s f(\hat{\tilde{\mathbf{z}}}(i), \mathbf{u}(i)) + \mathbf{e}(k+i) \tag{21a}$$

$$\mathbf{u}(i) = \begin{cases} \mathbf{u}^{\text{ref}}(i+k) + \\ K(i+k)(\hat{\tilde{\mathbf{z}}}(i) - \mathbf{z}^{\text{ref}}(i+k)) & : \text{CL} \\ \\ \mathbf{u}^{\text{ref}}(i+k) & : \text{OL} \end{cases} \tag{21b}$$

$$\hat{\tilde{\mathbf{z}}}(0) = \mathbf{z}_t \tag{21c}$$

The state errors terms $\mathbf{e}$, computed from equation (19), are added in equation (21) to capture the model mismatch between the vehicle model and the plant, especially during drifting. The controller selects the input sequence that results in a smaller tracking error, expressed in terms of the cost function below

$$J = \sum_{i=0}^{i=n} (\hat{\tilde{\mathbf{z}}}(i) - \mathbf{z}^{\text{ref}}(i+k))^T \hat{Q} (\hat{\tilde{\mathbf{z}}}(i) - \mathbf{z}^{\text{ref}}(i+k)) \tag{22}$$

In general, the controller selects closed-loop command in well-modeled region and selects open-loop maneuver poorly-modeled regions (i.e. drifting).

All computations online are conducted on an multi core Odroid XU-4.

5. SENSORS AND MODEL IDENTIFICATION

This section provides a brief overview of the model identification and state estimation methods used in this paper. These methods are based on previous work from Gonzales et al. (2016).

5.1 Longitudinal Dynamics Identification

We apply an Extended Kalman filter (EKF) to estimate the vehicle state using the vehicle model in (2)-(11). Inside the filter, however, we treat the longitudinal force F_{xR} as a synthetic input composed of the following forces

$$F_{xR} = F_{\text{motor}} + F_f + F_{\text{drag}} \tag{23}$$

where F_{motor} is the actuator input force, F_f is the friction force, $F_{\text{drag}} = C_D U_x^2$ is the drag force with C_D representing the aerodynamic drag coefficient. In the previous section on control design, we ignore the friction and air drag force (i.e. assume $F_{xR} = F_{\text{motor}}$).

In order to solve for the constants F_f and C_D, we perform a nonlinear least squares optimization using the longitudinal model in (2) and longitudinal force in (23) as follows,

$$\begin{aligned} \min_{F_f, C_D} \; & \left|\left| \dot{U}_x - \frac{1}{m}(F_{\text{motor}} + F_f + C_D U_x^2) \right|\right|_2^2 \\ \text{s.t.} \quad & F_f, C_D \geq 0 \end{aligned} \tag{24}$$

To generate experimental data for U_x and F_{motor}, we run short step-input tests with a fixed steering angle, $\delta = 0$.

5.2 Tire Force Identification

To identify the tire model, the RC vehicle is driven in circles, with fixed inputs to reach a steady state condition. After several tests from a range of steering angles δ and motor inputs F_{xR}, we solve the following optimization to estimate the tire stiffness parameters $C_{\alpha f}, C_{\alpha r}$

$$\begin{aligned} \min_{C_{\alpha f}, C_{\alpha r}, \boldsymbol{\beta}} \; & \left|\left| \begin{bmatrix} \dot{\beta} - \frac{1}{mU_x}(F_{yF} + F_{yR}) + r \\ \dot{r} - \frac{1}{I_z}(L_f F_{yF} - L_r F_{yR}) \end{bmatrix} \right|\right|_2^2 \\ \text{s.t.} \quad & \forall i \in 1, 2, 3, ...n \\ & C_{\alpha f} = C_{\alpha r} \\ & \alpha_F, \alpha_R = \text{side slip in Eq (10),(11)} \\ & F_{yF}, F_{yR} = \text{tire force in Eq (8), (9)} \end{aligned} \tag{25}$$

Note that in the above optimization, the parameters $C_{\alpha f}, C_{\alpha r} \in \mathbb{R}$ are scalars, while $\beta \in \mathbb{R}^n$ is a vector.

5.3 Position and Velocity Measurements

We use the onboard camera to estimate longitudinal and lateral velocity using optical flow. Optical flow methods calculate the motion between two image frames by computing spatial and temporal gradients of the brightness, and then solving the following equation

$$I_x v_x + I_y v_y + I_t = 0 \tag{26}$$

where I_x, I_y, I_t are the two spatial derivatives and temporal derivative, respectively, and v_x is the optical flow along the $x-$direction, and v_y is the optical flow along the $y-$direction. We obtain a position measurement using an indoor-gps kit from Marvelmind Robotics, which uses ultrasonic beacons to localize the vehicle.

5.4 State Estimation

We apply an EKF to synthesize measurements from different sensors using the vehicle model from (2)-(11), but with the longitudinal model in (24). The details of the EKF are available in the technical report [1] .

6. SIMULATION RESULTS

We first validated the proposed controller using Simulink and Carsim, a high fidelity simulation software, which uses over 250 state variables to estimate the vehicle dynamics. The simulated dynamics are very close to that of an actual vehicle. We used an A-class vehicle as the control plant since the software suite cannot provide any model at the similar scale with the RC car. The parameters of the vehicle and tires while stationary are summarized in Table 4.

The simulation results are shown in Figure 3 and Figure 4. We set a $\pm$ 2 degree offset for steering angle for each

Table 4. Carsim vehicle parameters

Parameter	Value	Parameter	Value
m [kg]	1830	L_r [m]	1.65
I_z [kg $\cdot$ m^2]	3287	$C_{\alpha f}$ [N/rad]	36000
L_f [m]	1.4	$C_{\alpha r}$ [N/rad]	36000

simulation. In these figures the blue path outlines the reference trajectory, the red path shows the trajectory under the proposed control algorithm, and the green line shows the path under open-loop commands with a steering angle offset. The black dash line is the boundary of the track. The simulation results suggest that by simply following the open-loop command, the vehicle cannot track the designed trajectory successfully. The vehicle collides with the boundaries of the track. The proposed strategy can control the vehicle to track the designed path well in both situations.

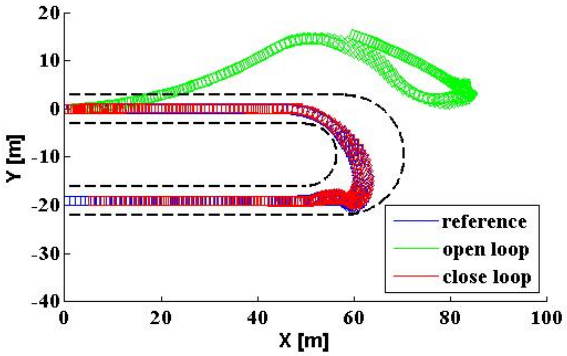

Fig. 3. Drift cornering position with different maneuvers (+2 degree offset)

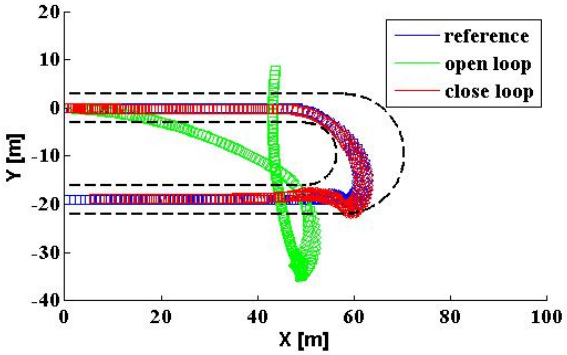

Fig. 4. Drift cornering position with different maneuvers (-2 degree offset)

7. EXPERIMENTAL RESULT

7.1 Hardware Platform

The authors use a low-cost, open-source 1/10 scale RC vehicle platform to validate the control design. The platform

Table 5. RC-car parameters

Parameter	Value	Parameter	Value
m [kg]	1.95	L_r [m]	0.125
I_z [kg $\cdot$ m^2]	0.24	$C_{\alpha f}$ [N/rad]	1.76
L_f [m]	0.125	$C_{\alpha r}$ [N/rad]	1.76

is called Berkeley Autonomous Race Car (BARC) [3]. The RC vehicle is shown in Figure 5, with parameters of the RC-car and tires summarized in Table 5.

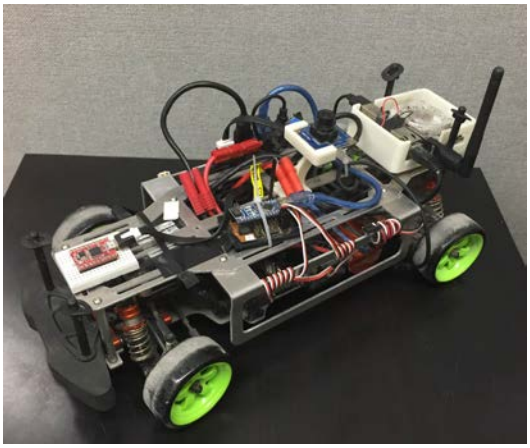

Fig. 5. Berkeley Autonomous Race Car (BARC)

7.2 Results from Experimentation

As with simulation, we applied the mixed open-loop and closed-loop control algorithm for the RC vehicle. The experiments were conducted in an indoor space with the RC vehicle set to the same initial position for each test. We began by running a series of open-loop tests using the rule based algorithm, recording the inputs during each test. After the tests, we selected a drift-cornering trajectory that didn't collide with the track boundaries, and set it as the reference trajectory. Next, we ran another two sets of experiments, one using the proposed algorithm, the other using the recorded inputs from the reference trajectory.
The blue paths in Figure 6 and Figure 7 show the reference trajectories. The green paths in Figure 6 are the experimental results when the control commands are exactly the same with inputs of reference trajectory. The green paths in Figure 7 are the results when the proposed mixed open-loop and closed-loop controller is applied.
These results show that by using the pre-recorded open-loop control commands, the vehicle fails to track the reference trajectory. By using the proposed mixed open-loop and closed-loop strategy, the vehicle can repeatably track the reference path. The vehicle recovers from drifting reference errors during the straight segment of the track.
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 snapshot of the experiment is shown in Figure 8. A video of the experiment can be found at barc-project.com.

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

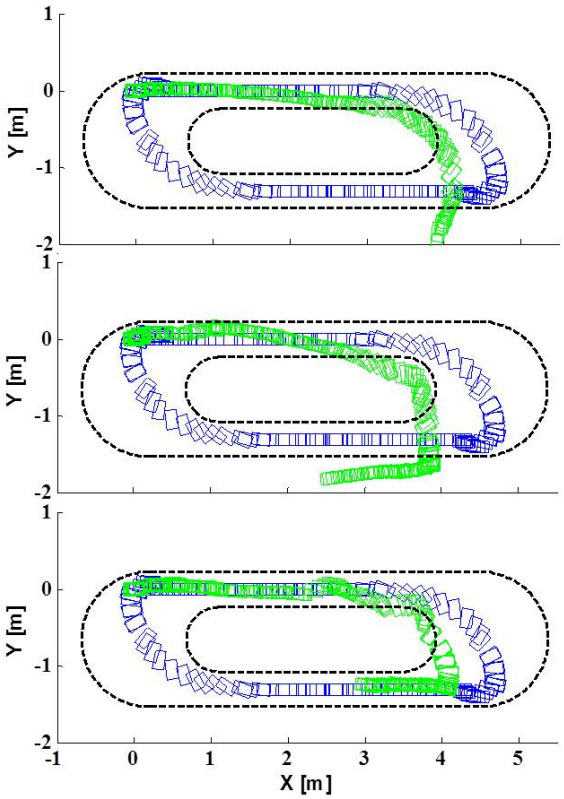

Fig. 6. Tracking performance of open-loop maneuver

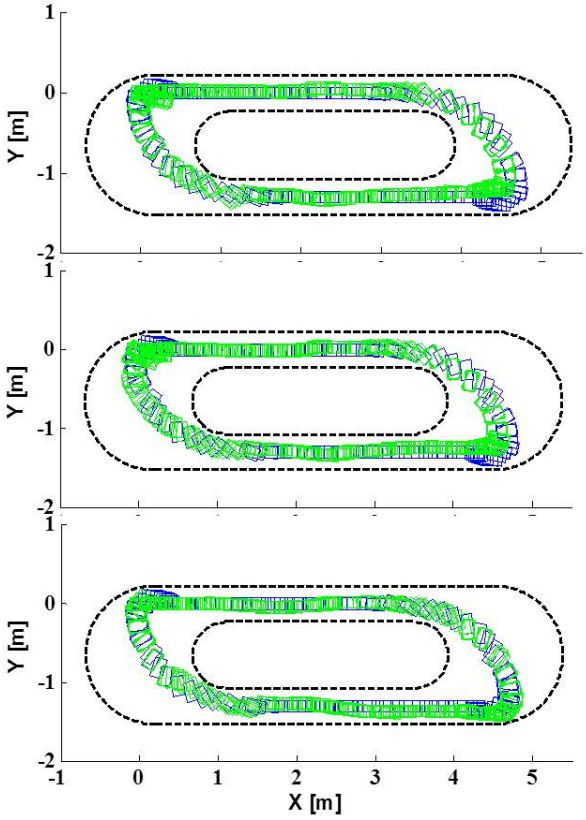

Fig. 7. Tracking performance of mixed open-loop and closed-loop maneuver

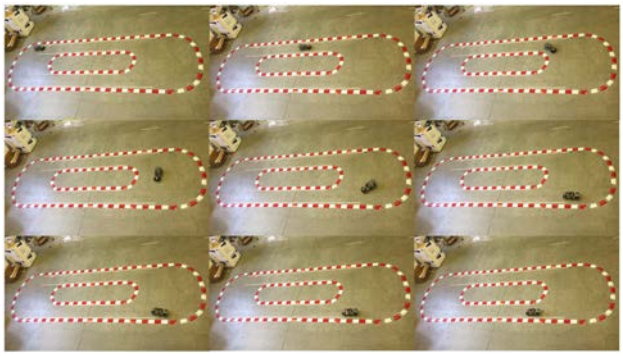

Fig. 8. The snapshot of experiment

8. CONCLUSION

In this paper, we demonstrate a mixed open-loop and closed-loop control scheme capable of achieving drift cornering. The proposed approach is validated in a simulation with a high fidelity model and experimentally with a 1/10 scale RC car. To the authors' knowledge, this is the first work without using the differential GPS or motion capture system. Future work includes achieving better drift performance, focusing on reducing tap time and extending the drifting duration.

ACKNOWLEDGEMENTS

This study is partially supported by NSF China with 51475254 and 51625503.

REFERENCES

Cutler, M.J. (2015). *Reinforcement learning for robots through efficient simulator sampling.* Ph.D. thesis, Massachusetts Institute of Technology.

Gonzales, J., Zhang, F., Li, K., and Borrelli, F. (2016). Autonomous drifting using onboard sensors. In *13th International Symposium on Advanced Vehicle Control*. AVEC.

Hindiyeh, R.Y. and Gerdes, J.C. (2014). A controller framework for autonomous drifting: Design, stability, and experimental validation. *Journal of Dynamic Systems, Measurement, and Control*, 136(5), 051015.

Kolter, J.Z., Plagemann, C., Jackson, D.T., Ng, A.Y., and Thrun, S. (2010). A probabilistic approach to mixed open-loop and closed-loop control, with application to extreme autonomous driving. In *Robotics and Automation (ICRA), 2010 IEEE International Conference on*, 839–845. IEEE.

Tavernini, D., Velenis, E., Lot, R., and Massaro, M. (2014). The optimality of the handbrake cornering technique. *Journal of Dynamic Systems, Measurement, and Control*.

Velenis, E. and Tsiotras, P. (2005). Minimum time vs maximum exit velocity path optimization during cornering. In *2005 IEEE international symposium on industrial electronics*, 355–360.

Velenis, E., Katzourakis, D., Frazzoli, E., Tsiotras, P., and Happee, R. (2011). Steady-state drifting stabilization of rwd vehicles. *Control Engineering Practice*, 19(11), 1363–1376.

Velenis, E., Tsiotras, P., and Lu, J. (2008). Optimality properties and driver input parameterization for trail-braking cornering. *European Journal of Control*, 14(4), 308–320.