Multi-robot trajectory optimization with collisions for the kinematic bicycle model

Tom Tseng
Computer Science and Artificial Intelligence Laboratory
Massachusetts Institute of Technology
Cambridge, MA, United States of America
tomtseng@mit.edu

***Abstract*—Existing multi-vehicle planning systems usually avoid collisions between robots. What happens if the systems instead allow collisions? This work provides a way to extend the kinematic bicycle model to allow collisions and describes the result of applying direct collocation to perform trajectory optimization on this model. Although optimizing over trajectories that allow collisions may produce faster trajectories in special cases, it greatly increases the complexity of the trajectory optimization problem.**

***Index Terms*—trajectory optimization, multi-robot systems, collision mitigation, cooperative systems**

I. Introduction

Mobile robots are common in industrial settings like factories and warehouses, and they are even appearing in consumer settings in the form of autonomous cars and unmanned aerial vehicles. When many mobile robots are in close proximity, however, coordination among them becomes a difficult task. Trajectories for several cooperative vehicles should not, for instance, cause the vehicles to crash into each other so violently that they damage each other, nor should they cause the vehicles to obstruct one other in narrow passageways.

In typical problem formulations for vehicle planning, the robots altogether avoid colliding into one another and into obstacles in the environment. This work explores whether there is benefit to relaxing this collision constraint.

We focus in particular on trajectory optimization for car-like vehicles in the two-dimensional plane. The goal is to find control inputs that propel each vehicle from a given start state to a given final position. Vehicles may collide into each other, causing a discontinuity in their trajectories. We choose to model the vehicles using the kinematic bicycle model, and we treat each vehicle body as a disk to make collisions easier to analyze. More complex models are certainly possible, but even these simple model choices illustrate the complexity of planning with collisions.

The contributions of this work are

- a extended version of the kinematic bicycle model that allows collisions,
- a brief analysis of using direct collocation for trajectory optimization over several bicycle vehicles that may collide into one another,
- a discussion of why for most vehicle planning situations, avoiding collisions like most other existing literature does is likely still the right choice.

II. Related work

Avoiding collisions is the sensible choice for many vehicle motion planning applications like highway driving in which a collision is prone to damage vehicles. Accordingly, most existing work in this area avoids collisions by enforcing some condition that is sufficient to keep the trajectories collision-free. For instance, [1] focuses on planning for a single robot. The robot only searches over choices of velocity that allow the robot to stop completely without hitting any obstacles. In [2], the vehicle drives on a highway and must stay within its lane. The vehicle's trajectory optimization procedure avoids obstacles in the vehicle's lane by conservatively avoiding a half-space containing the obstacle, and the procedure avoids obstacles in adjacent lanes by pretending that the lane is narrower when such obstacles are nearby.

In multi-robot systems, several robots work to achieve some goal. These powerful systems exist for a wide variety of tasks ranging from soccer [3] to military scouting [4] to construction [5]. Again focusing in particular on vehicle motion planning, in [6], several robots avoid collisions in a distributed, independent fashion by only choosing velocities that cannot cause a collision for a fixed time period. In [7], several aircraft avoid collisions in a centralized fashion by drawing bounding boxes around each aircraft body and enforcing with integer linear constraints that no aircraft enters another aircraft's bounding box.

(For brevity, this section is only a small sampling of existing work in motion planning and makes no attempt to be comprehensive.)

III. Vehicle model

We begin by describing the vehicle model used in this work.

A. Standard kinematic bicycle model

The kinematic bicycle model is a common choice for modeling car-like vehicles on the two-dimensional plane because it is analytically simple while still embodying

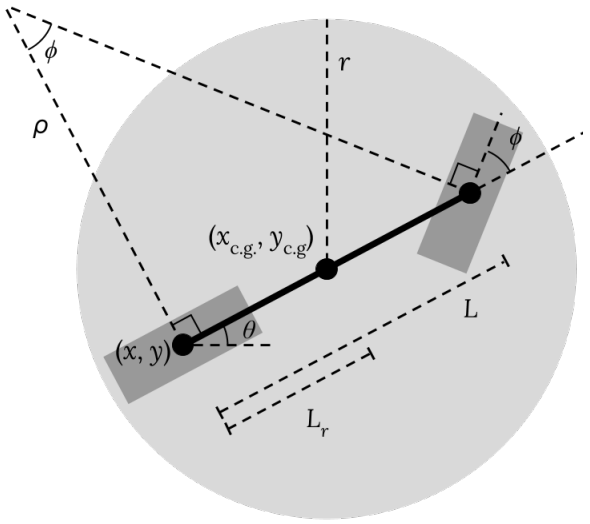

Fig. 1. An illustration of the bicycle model that this work uses. The rear wheel is on the left side, and the front wheel is on the right side.

some of the behavior of actual car dynamics. In the bicycle model, the wheels of a car are combined into a single front wheel and a single rear wheel. Figure 1 displays a diagram of the bicycle model.

Consider one particular vehicle under the bicycle model. Assume front-wheel-only steering so that the rear wheel cannot turn independently. Use the rear axle of the vehicle as the reference frame. Let (x, y) represent the position of the center of the rear axle of the vehicle, let θ represent the heading (the orientation of the vehicle), let ϕ represent the steering angle of the front axle relative to the heading, let $v_{\parallel,\theta}$ represent the velocity of the rear axle in the direction of the heading, and let L be the fixed distance between the rear axle and the front axle. Under the assumption that there is zero slip angle at the wheels — that is, that the velocity vector at the point at which a wheel meets the ground points in the same direction that the wheel does — the following kinematic equations emerge for the vehicle (c.f. chapter 2.2 of [8]):

$$\begin{aligned}\dot{x} &= v_{\parallel,\theta}\cos\theta,\\ \dot{y} &= v_{\parallel,\theta}\sin\theta,\\ \dot{\theta} &= \frac{v_{\parallel,\theta}\tan\phi}{L}.\end{aligned}$$

The no-slip assumption is reasonably accurate when the vehicle is moving at slow speeds.

There is an additional constraint that

$$|\phi| \le \phi_{\max} < \pi/2$$

for some constant max steering angle $\phi_{\max}$. This constraint represents the inability of the vehicle to turn infinitely quickly.

To control the vehicle, in this work, there are two control inputs. The first input applies a torque to the rear wheel to make it accelerate forwards. The second input controls the steering rate $\dot{\phi}$ at the front wheel.

Let $\mathbf{x}_\dagger$ and $\mathbf{u}_\dagger$ represent the state vector and the control vector of the vehicle respectively as follows:

$$\mathbf{x}_\dagger = \begin{bmatrix} x \\ y \\ \theta \\ v_{\parallel,\theta} \\ \phi \end{bmatrix}, \quad \mathbf{u}_\dagger = \begin{bmatrix} u_1 \\ u_2 \end{bmatrix}.$$

Then the dynamics equation $\mathbf{f}_\dagger$ of the vehicle is

$$\begin{bmatrix} \dot{x} \\ \dot{y} \\ \dot{\theta} \\ \dot{v}_{\parallel,\theta} \\ \dot{\phi} \end{bmatrix} = \dot{\mathbf{x}}_\dagger = \mathbf{f}_\dagger(\mathbf{x}_\dagger, \mathbf{u}_\dagger) = \begin{bmatrix} v_{\parallel,\theta}\cos\theta \\ v_{\parallel,\theta}\sin\theta \\ (v_{\parallel,\theta}\tan\phi)/L \\ u_1 \\ u_2 \end{bmatrix}.$$

B. Enabling collisions by extending the bicycle model

The standard kinematic bicycle model that section III-A presents is not rich enough to express the dynamics with collisions, however. The standard model assumes that the velocity of the rear axle is parallel to the vehicle heading. This is no longer necessarily true after a collision, which may give the vehicle an arbitrary velocity. (Depending on the geometry of the vehicle, a collision may also apply an angular impulse to the vehicle, although this work does not address this.)

To account for this, we extend the vehicle state by adding a lateral velocity coordinate $v_{\perp,\theta}$ that represents the component of the rear axle velocity that is 90° counterclockwise to the heading direction. The dynamics then must incorporate this lateral velocity.

For an angle γ, let $\mathbf{R}_\gamma$ denote a matrix that rotates points in the two-dimensional plane by angle γ:

$$\mathbf{R}_\gamma = \begin{bmatrix} \cos\gamma & -\sin\gamma \\ \sin\gamma & \cos\gamma \end{bmatrix}.$$

Then the position of the vehicle changes as follows:

$$\begin{bmatrix} \dot{x} \\ \dot{y} \end{bmatrix} = \mathbf{R}_\theta \begin{bmatrix} v_{\parallel,\theta} \\ v_{\perp,\theta} \end{bmatrix}.$$

For the dynamics of the heading θ and the velocity components $v_{\parallel,\theta}$ and $v_{\perp,\theta}$, first suppose that the only acceleration that the vehicle experiences is the centripetal acceleration a_c that results from having a non-zero steering angle ϕ. If $\rho = L/\tan\phi$ is the turning radius of the vehicle, then the centripetal acceleration is

$$a_c = \frac{v_{\parallel,\theta}^2}{\rho} = \frac{v_{\parallel,\theta}^2\tan\phi}{L} = v_{\parallel,\theta}\dot{\theta}$$

in the direction of $v_{\perp,\theta}$. Moreover, assume that the entire vehicle body experiences the added lateral velocity $v_{\perp,\theta}$ uniformly, which holds if no collision causes a change in angular momentum. Then the change in heading remains the same:

$$\dot{\theta} = \frac{v_{\parallel,\theta}\tan\phi}{L}.$$

Consider how the heading and velocity change in a small positive timestep Δt. The heading change is $\Delta\theta \approx \dot{\theta}\Delta t$. The velocity changes by adding $a_c\Delta t$ to the lateral velocity component and then rotating the velocity components to be relative to the new heading $\theta + \Delta\theta$:

$$\begin{bmatrix} v_{\parallel,\theta} + \Delta v_{\parallel,\theta} \\ v_{\perp,\theta} + \Delta v_{\perp,\theta} \end{bmatrix} \approx \mathbf{R}_{-\Delta\theta} \begin{bmatrix} v_{\parallel,\theta} \\ v_{\perp,\theta} + a_c\Delta t \end{bmatrix}.$$

As Δt approaches zero, so does $\Delta\theta$, at which point we may let $\cos(\pm\Delta\theta) \approx 1$ and $\sin(\pm\Delta\theta) \approx \pm\Delta\theta$. Then the right-hand side of the above equation is

$$\begin{aligned} &\mathbf{R}_{-\Delta\theta} \begin{bmatrix} v_{\parallel,\theta} \\ v_{\perp,\theta} + a_c\Delta t \end{bmatrix} \\ &= \begin{bmatrix} \cos(-\Delta\theta) & \sin(\Delta\theta) \\ \sin(-\Delta\theta) & \cos(-\Delta\theta) \end{bmatrix} \begin{bmatrix} v_{\parallel,\theta} \\ v_{\perp,\theta} + v_{\parallel,\theta}\dot{\theta}\Delta t \end{bmatrix} \\ &\approx \begin{bmatrix} 1 & \dot{\theta}\Delta t \\ -\dot{\theta}\Delta t & 1 \end{bmatrix} \begin{bmatrix} v_{\parallel,\theta} \\ v_{\perp,\theta} + v_{\parallel,\theta}\dot{\theta}\Delta t \end{bmatrix} \\ &= \begin{bmatrix} v_{\parallel,\theta} + v_{\perp,\theta}\dot{\theta}\Delta t + v_{\parallel,\theta}\dot{\theta}^2(\Delta t)^2 \\ v_{\perp,\theta} \end{bmatrix}. \end{aligned}$$

Let Δt approach zero and remove second-order $(\Delta t)^2$ terms to get

$$\begin{bmatrix} \dot{v}_{\parallel,\theta} \\ \dot{v}_{\perp,\theta} \end{bmatrix} = \begin{bmatrix} v_{\perp,\theta}\dot{\theta} \\ 0 \end{bmatrix}.$$

This equation assumes that the only source of acceleration for the vehicle is the centripetal acceleration a_c. The last missing piece for the dynamics is to account for outside forces on the vehicle. One outside force is the rear wheel torque control input, which directly adds an additional term to $\dot{v}_{\parallel,\theta}$. The trickier outside force to account for is the cornering force: for the cylindrical tires typical for car-like vehicles, having a lateral velocity component implies that the tire experiences a cornering force from the resulting slip angle.

To account for the cornering force, we make a few more simplifying assumptions. First, assume that the only source of slip angle is the lateral velocity $v_{\perp,\theta}$ that comes from collisions. Therefore, the original zero-slip-angle assumption of the standard kinematic bicycle model still applies when $v_{\perp,\theta}$ is zero. Second, assume that the acceleration experienced at the rear wheel as a result of its cornering force is approximately equal to the cornering acceleration experienced at the front wheel. This is accurate when the steering angle is small, and it simplifies the vehicle dynamics by making the lateral velocity not affect steering. Third, assume that the cornering force is proportional to the slip angle, which is accurate when the slip angle is small.

(Like with all modeling assumptions, these assumptions trade realism for analytical simplicity and ease of exposition. A more accurate dynamics model would analyze the slip at each tire more carefully (c.f. chapter 2.3 of [8]) and use a more accurate conversion from slip angle to cornering force.)

The slip angle at the rear wheel is $\arctan\left(v_{\perp,\theta}/\left|v_{\parallel,\theta}\right|\right)$. Let u_1 be the rear wheel torque control input and $C_{\text{corner}} > 0$ be a fixed cornering coefficient. Then the full dynamics for the velocity is

$$\begin{bmatrix} \dot{v}_{\parallel,\theta} \\ \dot{v}_{\perp,\theta} \end{bmatrix} = \begin{bmatrix} v_{\perp,\theta}\dot{\theta} + u_1 \\ -C_{\text{corner}} \arctan \frac{v_{\perp,\theta}}{|v_{\parallel,\theta}|} \end{bmatrix}.$$

Putting all of this together gives the dynamics of the full system. Let $\mathbf{x}$ and $\mathbf{u}$ represent the state vector and the control vector of the vehicle respectively as follows:

$$\mathbf{x} = \begin{bmatrix} x \\ y \\ \theta \\ v_{\parallel,\theta} \\ v_{\perp,\theta} \\ \phi \end{bmatrix}, \quad \mathbf{u} = \begin{bmatrix} u_1 \\ u_2 \end{bmatrix}.$$

Then the dynamics equation $\mathbf{f}$ of the vehicle is

$$\begin{bmatrix} \dot{x} \\ \dot{y} \\ \dot{\theta} \\ \dot{v}_{\parallel,\theta} \\ \dot{v}_{\perp,\theta} \\ \dot{\phi} \end{bmatrix} = \dot{\mathbf{x}} = \mathbf{f}(\mathbf{x}, \mathbf{u}) = \begin{bmatrix} v_{\parallel,\theta}\cos\theta - v_{\perp,\theta}\sin\theta \\ v_{\parallel,\theta}\sin\theta + v_{\perp,\theta}\sin\theta \\ (v_{\parallel,\theta}\tan\phi)/L \\ (v_{\perp,\theta}v_{\parallel,\theta}\tan\phi)/L + u_1 \\ -C_{\text{corner}} \arctan \frac{v_{\perp,\theta}}{|v_{\parallel,\theta}|} \\ u_2 \end{bmatrix}.$$

C. Collisions

In order to determine when and how vehicle collisions occur, there needs to be a model of the vehicle body. For this work, we assume that the vehicle body is circular and that the center of gravity of the vehicle is at the center of the circular body. This makes collision behavior simple and predictable.

Recall that for a vehicle at state $\mathbf{x} = \begin{bmatrix} x & y & \theta & v_{\parallel,\theta} & v_{\perp,\theta} & \phi \end{bmatrix}^\top$, the (x, y) coordinates are the coordinates of the rear axle of the vehicle. Let the constant $L_r \in [0, L]$ be the distance from the rear axle to the center of gravity of the vehicle. Then the center of gravity is located at

$$\begin{bmatrix} x_{\text{c.g.}} \\ y_{\text{c.g.}} \end{bmatrix} = \begin{bmatrix} x + L_r\cos\theta \\ y + L_r\sin\theta \end{bmatrix}.$$

With two vehicles with bodies of radius R, a collision occurs between the two vehicles when the distance between the centers of gravity of the vehicles is $2R$.

Suppose two circular vehicles collide with one vehicle in state $\mathbf{x} = \begin{bmatrix} x & y & \theta & v_{\parallel,\theta} & v_{\perp,\theta} & \phi \end{bmatrix}^\top$ and the other vehicle in state $\mathbf{x}' = \begin{bmatrix} x' & y' & \theta' & v'_{\parallel,\theta} & v'_{\perp,\theta} & \phi' \end{bmatrix}^\top$ prior to the collision. Due to the assumption that the center of gravity of each vehicle is located at the center of its circular body and due to the symmetry of circles, the collision does not change the angular momentum of either vehicle and is purely an exchange of linear momentum. To be more explicit, let $(x_{\text{c.g.}}, y_{\text{c.g.}})$ and $(x'_{\text{c.g.}}, y'_{\text{c.g.}})$ be the coordinates

of the center of gravity of each vehicle. The angle α of the line between the two centers of gravity is

$$\arctan\left(\frac{y'_{\text{c.g.}} - y_{\text{c.g.}}}{x'_{\text{c.g.}} - x_{\text{c.g.}}}\right).$$

This line is orthogonal to the tangent line of the point of collision, so the vehicles only exchange linear momentum along this line assuming that the vehicle bodies are perfectly smooth. The velocity components of the first vehicle relative to this angle α are

$$\begin{bmatrix} v_{\parallel,\alpha} \\ v_{\perp,\alpha} \end{bmatrix} = \mathbf{R}_{\theta-\alpha} \begin{bmatrix} v_{\parallel,\theta} \\ v_{\perp,\theta} \end{bmatrix}$$

and likewise for the second vehicle's velocity components $v'_{\parallel,\alpha}$ and $v'_{\perp,\alpha}$. Let $V_{\parallel,\alpha}$ and $V'_{\parallel,\alpha}$ denote the post-collision velocity components of the vehicles along angle α, let $C_{\text{restitution}} \in [0,1]$ be the coefficient of restitution between the two vehicles, and let m and m' denote the masses of the two vehicles. Then

$$V_{\parallel,\alpha} = \frac{C_{\text{restitution}} m'\left(v'_{\parallel,\alpha} - v_{\parallel,\alpha}\right) + m v_{\parallel,\alpha} + m' v'_{\parallel,\alpha}}{m + m'}$$

$$V'_{\parallel,\alpha} = \frac{C_{\text{restitution}} m\left(v_{\parallel,\alpha} - v'_{\parallel,\alpha}\right) + m v_{\parallel,\alpha} + m' v'_{\parallel,\alpha}}{m + m'}.$$

We assume that the vehicle bodies are smooth so that the velocity component perpendicular to angle α is left unchanged for each vehicle, but a more realistic model may also choose to change the perpendicular velocity components to account for friction.

Now the post-collision velocity of the first vehicle relative to its heading θ is

$$\begin{bmatrix} V_{\parallel,\theta} \\ V_{\perp,\theta} \end{bmatrix} = \mathbf{R}_{\alpha-\theta} \begin{bmatrix} V_{\parallel,\alpha} \\ v_{\perp,\alpha} \end{bmatrix},$$

and likewise for the second vehicle.

(If the vehicle body does not have its center of gravity at its centroid or is not circular, collision behavior is considerably more complex, c.f. chapter 2 of [9].)

IV. Trajectory optimization

A. Problem formulation

Fix the number of vehicles. Fix some start state $\mathbf{x}_{\text{start}}$ and goal state $\mathbf{x}_{\text{goal}}$ for each vehicle. The goal is choose a duration T and control inputs $\mathbf{u}(t)$ across time $t \in [0, T]$ so that a trajectory $\mathbf{x}(\cdot)$ starting at the start state $\mathbf{x}(0) = \mathbf{x}_{\text{start}}$ and following the dynamics $\dot{\mathbf{x}}(t) = \mathbf{f}(\mathbf{x}(t), \mathbf{u}(t))$ ends at the goal state $\mathbf{x}(T) = \mathbf{x}_{\text{goal}}$. We prefer trajectories with a lower duration T.

The control inputs $\mathbf{u}(t) = \begin{bmatrix} u_1(t) & u_2(t) \end{bmatrix}^\top$ are limited by some maximum values $u_{1,\max}$ and $u_{2,\max}$ so that for all $t \in [0, T]$,

$$|u_1(t)| \le u_{1,\max}, \quad |u_2(t)| \le u_{2,\max}.$$

If the vehicles collide, their velocities must be updated as described in section III-C. We assume that there is nothing else in the environment besides the vehicles, though this assumption could be relaxed by adding more constraints and defining any relevant collision behavior with environment obstacles.

B. Direct collocation approach

Direct collocation (c.f. [10]) can solve for control inputs for this problem. (The resulting trajectories only follow the dynamics constraint and other constraints approximately — in practice, we would want to use something like model predictive control [11], [12] to actually execute these trajectories.)

To apply direct collocation in a straightforward manner, we assume that we a priori know a (possibly empty) collision sequence that specifies which pairs of vehicles collide in the trajectory and in what order they collide. We later manually search over different settings of this collision sequence. (This is acceptable when the number of vehicles is small but becomes unwieldy when there are many vehicles.)

We control the vehicles in a centralized manner by solving for the trajectories of all vehicles at once. Given the collision sequence of k collisions, we represent the full trajectory as $k+1$ collision-free sub-trajectories. For each sub-trajectory, we have collocation constraints that enforce a cubic spline for the sub-trajectory. (See chapter 10.3.2 of [13] for a more detailed discussion of direct collocation constraints). For $i \in [1, k]$, the i-th sub-trajectory is related to the next one by a constraint that the initial state of sub-trajectory $i+1$ matches the final state of sub-trajectory i with the velocities adjusted according to the i-th collision in the collision sequence. (See chapter 17.1.1 of [13] for an overview of using direct collocation for hybrid trajectory optimization in this manner.)

Finally, with all the appropriate constraints in place, we hand everything over to a mathematical program solver and let it solve for a trajectory.

V. Experiments

A. Implementation

We implement the direct collocation approach in Python with the help of the Drake library [14] and use Drake's SNOPT [15] bindings as the solver. The code is publicly available[1] and displays an animation of the resulting trajectory (screenshot in figure 2).

[1] https://github.com/tomtseng/colliding-multi-vehicle-trajectories

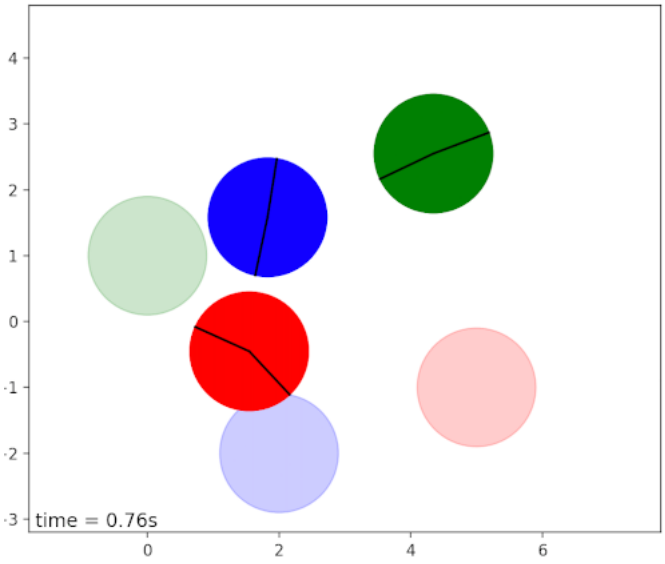

Fig. 2. A screenshot of the animation produced by the direct collocation implementation.

We make several arbitrary choices of constants for each car, all of which are adjustable:

$$\begin{aligned} C_{\text{corner}} &= 5 \text{ meters}/(\text{second}^2 \cdot \text{radian}), \\ C_{\text{restitution}} &= 0.1, \\ L_r &= 0.6 \text{ meters}, \\ \text{car radius} = R &= 0.9 \text{ meters}, \\ \text{car mass} = m &= 1300 \text{ kg}, \\ \phi_{\max} &= \pi/4 \text{ radians}, \\ u_{1,\max} &= 3.9 \text{ meters/second}^2, \\ u_{2,\max} &= \pi/2 \text{ radians/second}. \end{aligned}$$

We modify the goal condition so that the goal for each vehicle is to place its center of gravity on a given goal position and to have no velocity at the goal position. The original problem formulation in section IV-A would enforce that the goal state has some fixed heading θ_{goal}, which is undesirable since headings are indistinguishable modulo 2π. The modified goal condition avoids this issue.

B. Qualitative observations

The implementation can handle an arbitrary number of vehicles in theory, but in practice, once there are more than two vehicles, solving for trajectories takes an inconveniently long time. Even for two vehicles, the solve time is on the order of tens of seconds on a modern consumer laptop. In addition, the solver sometimes fails due to numerical difficulties, and this issue seems to worsen when there are more vehicles.

Because the constraints are non-linear, the resulting trajectories may be local minima that clearly are not global minima. For instance, the output trajectory may have one vehicle drive in reverse towards the goal even though driving forwards may be a more direct path.

Situations in which collisions improve the optimal trajectory are uncommon but do exist. Such situations include the following:

- The vehicles may approach their goal locations at high velocities and then ram into each other to reduce their velocities instantaneously.
- If one vehicle has a goal position that is particularly far away and hence takes longer than other vehicles to reach its goal, then other vehicles may collide with it to push it towards its goal.
- If the start state of the vehicles makes a collision inevitable (i.e., two vehicles start close to each other and have velocity vectors pointing towards each other), then the only feasible trajectories involve collisions. Collisions may also be helpful if the start state would force vehicles to swerve to avoid a collision.
- If one vehicle is somehow more "powerful" than another (e.g., one vehicle has significantly more mass or has a higher rear wheel torque limit $u_{1,\max}$), then collisions may helpfully transfer momentum to the less powerful vehicle.

C. A brief comparison study

We perform a small, limited experiment to compare solver performance under various conditions. To generate a scenario for this experiment, we fix the number of cars to be two and draw the start and goal positions (x, y) uniformly at random from $[-5, 5]^2$, the start headings from $[-2\pi, 2\pi]$, start forward velocity from $[-5, 5]$, and the start turn angle from $[-\phi_{\max}, \phi_{\max}]$. We generate 30 scenarios in this manner, replacing any scenario in which the two cars have overlapping start positions or overlapping goal positions with another newly generated one.

We set the number of direction collocation time samples to be 60, and we restrict the number of seconds between time samples to be in the range $[0.005, 0.2]$. We set the SNOPT iteration limit to be 100,000 and the SNOPT major iteration limit to be 10,000. (See [16] for a description of these SNOPT settings.)

We run the code on an Amazon Web Services c5.4xlarge instance, which has the Intel Xeon Platinum 8000 series processor, a clock speed of 3.6 GHz, 16 vCPUs, and 32 GiB of RAM. The instance uses Ubuntu 18.04 as its operating system and has Python 3.6.9 installed.

We solve for a trajectory under three different setups:

(A) No collisions — the collision sequence is empty,
(B) One collision — the collision sequence specifies that there is one collision between the two cars,
(C) Standard model — we solve for a collision-free trajectory using the standard bicycle model from section III-A instead.

Solving for trajectories, even with such simple scenarios, is expensive. For setup (A), for example, the mean time

for the solver to compute an output is 41.3 seconds with a standard deviation of 30.1 seconds.

1) The effect of having a collision: How often does the one-collision condition (setup (B)) outperform the no-collision condition (setup (A))? Not often; in only six scenarios does the one-collision condition result in a trajectory that is more than 1% better than the no-collision condition. Moreover, in some of these scenarios, the improvement comes from the output trajectory under no-collision condition being a local minima rather than from the collision itself being a helpful action.

There are three scenarios in which the one-collision setup hits the solver iteration limit and fails to output a trajectory but the no-collision setup successfully finds a trajectory. There is also one scenario in which the solver experiences numerical difficulties and fails to find a solution under the one-collision setup. If we compare the percentage difference in solve time on a per-scenario basis, regarding the four aforementioned scenarios as infinitely bad for the one-collision condition, the solve time under the one-collision condition is a median of 32.7% slower compared to the no-collision condition.

These results suggest that having a collision is only occasionally helpful and tends to increase the computational difficulty of finding a trajectory.

2) The computational burden of the extended bicycle model: Section III-B described how to extend the standard kinematic bicycle model to allow collisions. Suppose that we use this extended bicycle model but disallow collisions anyway. Is this more difficult for the solver than simply using the standard kinematic bicycle model from section III-A? To find out, compare the performance under the no-collision extended-model condition (setup (A)) with the performance under the no-collision standard-model condition (setup (C)).

There is one scenario that the solver regards as infeasible under both models. There is one scenario in which the solver hits its iteration limit under the extended model but not the standard model, and there are four scenarios in which the solver hits its iteration limit under the standard model but not the extended model.

Iteration limits aside, the solver is usually significantly faster at finding trajectories for the standard model. When we consider the percentage difference in solve time on a per-scenario basis, regarding situations in which the solver hits the iteration limit under one model but not the other as infinitely good or bad, the solve time with the standard model is a median of 43.3% faster than the solve time with the extended model.

The duration of the trajectories also tends to be better under the standard model, but only slightly. Of the 24 scenarios in which the solver finds a trajectory for both models, there are 6 scenarios for which the trajectory is faster for the extended model, 4 scenarios for which the trajectory duration is the same for both models, and 14 scenarios for which the trajectory is faster for the standard model. However, the median percentage difference in trajectory duration is only .17%, and the mean percentage difference is .35%.

The results of this experiment suggest that using the extended bicycle model described in section III-B instead of the standard kinematic bicycle model leads to substantially worse solve times, even though the behavior of the two models is theoretically the same when there are no collisions. Users of the extended bicycle model should be wary of this performance cost.

VI. Discussion

There is much more work needed to actually solve for trajectories in practical scenarios. The direct collocation approach that section IV-B describes is inconvenient because it requires the user to either guess the sequence of collisions up front or search the space of plausible collision sequences. We only consider the straightforward direct collocation approach in this work because of the relative simplicity of direct collocation, but people who wish to solve for collision-allowed trajectories in more complex environments should consider other approaches that remove this inconvenience (for example, see [17]). Moreover, solving for trajectories that exactly follow the vehicle dynamics is likely computationally infeasible — indeed, the direct collocation approach only satisfies the dynamics approximately. As a result, someone who wishes to actually execute trajectories according to the dynamics would want to use something like model predictive control, which would require the ability to solve for approximate trajectories much more quickly than the implementation from section V-A allows.

Even if all that work is done, there are several major reasons why people should prefer only solving for collision-free trajectories like existing vehicle planning work usually does instead of allowing collisions:

- Accounting for collisions when the vehicle bodies are complicated shapes is difficult. Computing the result of a collision requires determining the tangent line to the contact point and computing an exchange of linear momentum and angular momentum. (See chapter 2 of [9] for a reference on the behavior of collisions in the two-dimensional plane.) And even just checking for a collision between two regular polygons, for example, is a non-trivial task that solvers might not handle well. Collision avoidance is easier than true collision detection because collision avoidance only requires avoiding a necessary condition for collision (e.g., avoiding a bounding sphere around other obstacles) whereas collision detection requires specifying a condition that is both necessary and sufficient for collision.
- Accounting for collisions may require a more complicated vehicle dynamics model. In this work, we had to modify the vehicle model in section III-B to allow collisions, and the results of section V-C2 suggest that

modifying the model in this manner adds significant computational burden.

- Although there are some opportunities for collisions to lead to faster trajectories (e.g., vehicles can use collisions to instantaneously decelerate), such opportunities tend to be infrequent. This of course depends on the problem and context, though.
- Collisions may damage vehicles and change their dynamics. For example, a high-speed collision may ruin the steering alignment for a vehicle. A high-speed collision could also deform the body of a vehicle, making future collision calculations incorrect.
- Computing trajectories that collide with other moving objects requires either centralized control over those moving objects (like what we do in this work) or good prediction of the movement of those objects. Centralized control is computationally expensive when there are many objects, and prediction is difficult and prone to error.
- Modeling collisions accurately for physical systems requires a lot of experimental work to confirm that the model actually conforms well to real-world dynamics. For instance, the coefficient of restitution $C_{\text{restitution}}$ is not actually a constant, fundamental property of the colliding materials and can depend on the velocity and geometry of the impact. Physical experiments to determine the coefficient of restitution under a variety of situations may be expensive in both time and resources, especially if collisions could physically damage the vehicles of interest.

If collisions are computationally burdensome to account for (which is almost certainly true) and have limited helpfulness for the problem at hand, then it is likely better to optimize for the common case by focusing on computing trajectories that avoid collisions altogether.

References

[1] D. Fox, W. Burgard, and S. Thrun, "The dynamic window approach to collision avoidance," *IEEE Robotics & Automation Magazine*, vol. 4, no. 1, pp. 23–33, 1997.

[2] M. Jalalmaab, B. Fidan, S. Jeon, and P. Falcone, "Model predictive path planning with time-varying safety constraints for highway autonomous driving," in *2015 International Conference on Advanced Robotics (ICAR)*. IEEE, 2015, pp. 213–217.

[3] H. Kitano, M. Asada, Y. Kuniyoshi, I. Noda, and E. Osawa, "Robocup: The robot world cup initiative," in *Proceedings of the first international conference on autonomous agents*, 1997, pp. 340–347.

[4] T. Balch and R. C. Arkin, "Behavior-based formation control for multirobot teams," *IEEE transactions on robotics and automation*, vol. 14, no. 6, pp. 926–939, 1998.

[5] A. Stroupe, T. Huntsberger, A. Okon, H. Aghazarian, and M. Robinson, "Behavior-based multi-robot collaboration for autonomous construction tasks," in *2005 IEEE/RSJ International Conference on Intelligent Robots and Systems*. IEEE, 2005, pp. 1495–1500.

[6] J. Van Den Berg, S. J. Guy, M. Lin, and D. Manocha, "Reciprocal n-body collision avoidance," in *Robotics research*. Springer, 2011, pp. 3–19.

[7] A. Richards and J. P. How, "Aircraft trajectory planning with collision avoidance using mixed integer linear programming," in *Proceedings of the 2002 American Control Conference (IEEE Cat. No. CH37301)*, vol. 3. IEEE, 2002, pp. 1936–1941.

[8] R. Rajamani, *Vehicle dynamics and control*. Springer Science & Business Media, 2011.

[9] W. Goldsmith, *Impact: The theory and physical behavior of colliding solids*. E. Arnold, 1960.

[10] C. R. Hargraves and S. W. Paris, "Direct trajectory optimization using nonlinear programming and collocation," *Journal of guidance, control, and dynamics*, vol. 10, no. 4, pp. 338–342, 1987.

[11] M. Morari, C. E. Garcia, and D. M. Prett, "Model predictive control: theory and practice," *IFAC Proceedings Volumes*, vol. 21, no. 4, pp. 1–12, 1988.

[12] E. F. Camacho and C. B. Alba, *Model predictive control*. Springer Science & Business Media, 2013.

[13] R. Tedrake, "Underactuated robotics: Algorithms for walking, running, swimming, flying, and manipulation (course notes for MIT 6.832)," Downloaded on 12 May 2020. [Online]. Available: http://underactuated.mit.edu

[14] R. Tedrake and the Drake Development Team, "Drake: Model-based design and verification for robotics," 2019. [Online]. Available: https://drake.mit.edu

[15] P. E. Gill, W. Murray, and M. A. Saunders, "SNOPT: An SQP algorithm for large-scale constrained optimization," *SIAM review*, vol. 47, no. 1, pp. 99–131, 2005.

[16] ——, "User's guide for SNOPT version 7: Software for large-scale nonlinear programming, systems optimization laboratory (SOL)," 2006.

[17] M. Posa, C. Cantu, and R. Tedrake, "A direct method for trajectory optimization of rigid bodies through contact," *The International Journal of Robotics Research*, vol. 33, no. 1, pp. 69–81, 2014.