AutonoVi: Autonomous Vehicle Planning with Dynamic Maneuvers and Traffic Constraints

Andrew Best[1] and Sahil Narang[1] and Daniel Barber[2] and Dinesh Manocha[1]
http://gamma.cs.unc.edu/AutonoVi/video.avi (video included)

***Abstract*— We present AutonoVi:, a novel algorithm for autonomous vehicle navigation that supports dynamic maneuvers and satisfies traffic constraints and norms. Our approach is based on optimization-based maneuver planning that supports dynamic lane-changes, swerving, and braking in all traffic scenarios and guides the vehicle to its goal position. We take into account various traffic constraints, including collision avoidance with other vehicles, pedestrians, and cyclists using control velocity obstacles. We use a data-driven approach to model the vehicle dynamics for control and collision avoidance. Furthermore, our trajectory computation algorithm takes into account traffic rules and behaviors, such as stopping at intersections and stoplights, based on an arc-spline representation. We have evaluated our algorithm in a simulated environment and tested its interactive performance in urban and highway driving scenarios with tens of vehicles, pedestrians, and cyclists. These scenarios include jaywalking pedestrians, sudden stops from high speeds, safely passing cyclists, a vehicle suddenly swerving into the roadway, and high-density traffic where the vehicle must change lanes to progress more effectively.**

I. INTRODUCTION

Autonomous driving is a difficult and extremely complex task that has immense potential for impacting the lives of billions of people. In order to develop autonomous capabilities to perform the driving task, we need appropriate capabilities to sense and predict the traffic and road obstacles, as well as for planning, control, and coordination of the vehicle [1], [2]. There is considerable research in this area that borrows ideas from different disciplines including computer vision, machine learning, motion planning, mechanical engineering, intelligent traffic simulation, human-factors psychology, etc.

Research into sensing and perception technologies has been progressing considerably and current vehicle sensors seem to have the capability to detect relevant obstacles, vehicles, and other traffic entities including bicycles and pedestrians. However, automatic planning in different scenarios and the computation of the appropriate response to vehicle and non-vehicle entities, such as bicycles and pedestrians, are still the subjects of ongoing research. A key issue is the development of an efficient navigation algorithm for autonomous driving that takes into account the vehicle dynamics, sensor inputs, traffic rules and norms, and the driving behaviors of other vehicles. Moreover, the uncertainties in the sensor data, capability, and response of the autonomous vehicle, typically referred to as the *ego-vehicle* [3], have led to the development of behavior and navigation algorithms that impose conservative limits on the acceleration, deceleration, and steering decisions. For example, algorithms tend to limit hazard responses to either steering [4], [5] or braking [6]. Few algorithms demonstrate combined control of throttle and steering and typically do so in constrained navigation scenarios [7]. In terms of planning the routes and navigating the roads, current algorithms tend to limit the lane-changing behaviors, precluding their use for progressing more quickly to a goal or better utilization of the road conditions. These limitations have led to the perception that autonomous cars behave more like student drivers taking their driving test than actual skilled human drivers [3]. One of the goals is to extend the capabilities of current autonomous vehicles in terms of planning, control, and navigation, making them less conservative but still allowing safe performance during driving.

Main contributions: We present a novel navigation algorithm for autonomous vehicles, AutonoVi, which utilizes a data-driven vehicle dynamics model and optimization-based maneuver planning to compute a safe, collision-free trajectory with dynamic lane-changes. Our approach is general, makes no assumption about the traffic conditions, and plans dynamically feasible maneuvers in traffic consisting of other vehicles, cyclists, and pedestrians. In order to develop an autonomous vehicle planning approach with these capabilities, we present four novel algorithms:

- **Optimization-based Maneuvering:** We describe a novel multi-objective optimization approach for evaluating the dynamic maneuvers. Our algorithm encodes passenger comfort, safe passing distances, maneuver constraints in terms of dynamics, and global route progress in order to compute appropriate trajectories.
- **Data driven Vehicle Dynamics:** We use a data-driven vehicle dynamics formulation that encodes feasible accelerations, steering rates, and decelerations into a set of per-vehicle profile functions, which can be quickly evaluated. These profiles are generated by simulating the ego-vehicle through a series of trials to obtain lateral and longitudinal slip profiles. This data-driven model generalizes to multiple vehicles and configurations.
- **Collision avoidance with kinematic and dynamic constraints:** We present a collision avoidance algorithm that combines collision-free constraints with specific kinematic and dynamic constraints of the autonomous vehicle. Our approach allows the autonomous vehicle to steer away from collisions with other vehicles, pedes-

[1]Andrew Best and Sahil Narang and Dinesh Manocha are at the University of North Carolina, Chapel Hill
[2] Daniel Barber is at the University of Central Florida

trians, and cyclists as well as to apply brakes, or use a combination of steering and braking.

- **Trajectory Planning with Traffic Rules and Behaviors:** We present a trajectory planning algorithm that encodes traffic rules and road behaviors along with lane-following for computing safe trajectories. Our approach is based on computing arcs along the center-line of the current lane to generate an initial trajectory that satisfies all the constraints. This initial trajectory is computed and refined according to collision avoidance and maneuver optimization computations.

We evaluate our algorithm in a set of traffic scenarios generated using a physics-based traffic simulator in both sparse and dense traffic conditions with tens of other vehicles, pedestrians, and cyclists. We demonstrate collision-avoidance events including a vehicle suddenly driving into the road, traffic suddenly stopping ahead of the ego-vehicle while travelling at high speed, and a pedestrian jaywalking in front of the ego-vehicle, representing typical accident scenarios [5]. Our approach enables advantageous of lane changes (e.g., overtaking) and adherence to traffic rules in typical traffic conditions. It also exhibits safe maneuvering in the presence of heavy traffic, pedestrians, and cyclists. To our knowledge, AutonoVi is the first approach that allows the ego-vehicle to follow an arbitrary route, determine appropriate lane changes dynamically during travel, and plan dynamically and kinematically feasible trajectories while following traffic norms and providing collision avoidance for vehicles, pedestrians, and cyclists.

The rest of the paper is organized as follows: we detail relevant related work in section 2. In section 3, we introduce the vehicle kinematic model, define relevant assumptions, and introduce the terminology used in the rest of the paper. In section 4, we present our navigation algorithm, AutonoVi, and its components. We present the details of our simulation benchmarks in section V and highlight the results in section VI.

II. RELATED WORK

The problem of autonomous driving has been widely studied in robotics, computer vision, intelligent transportation systems and related areas. In this section, we give a brief overview of prior methods which address motion planning and navigation, dynamics, behavior generation, and collision avoidance. More detailed surveys are given in [2], [8], [9].

Vehicle Kinematics and Dynamics Modeling: A number of approaches have been developed to model the motion of a moving vehicle. Different models offer a trade-off between simplicity or efficiency of the approach, and physical accuracy. Simpler models are typically based on linear dynamics and analytical solutions to the equations of motion [10]. More accurate models provide a better representation of the physical motion, but require more computational power to evaluate and incorporate non-linear forces in the vehicle dynamics [4]. The Reeds-Shepp formulation is a widely used car model with forward and backward gears [11]. Margolis and Asgari [12] present several representations of a car including the widely used single-track bicycle model. Borrelli et al. [4] extend this model by including detailed tire-forces. Current planning and control algorithms leverage varying levels of detail in the model of the vehicle.

Path Planning and Collision Avoidance: Prior approaches to path planning for autonomous vehicles are based on occupancy grids [13], random-exploration [14], driving corridors [15], potential-field methods [16], etc. Recent approaches seek to incorporate driver behavior prediction in path planning using game-theoretic approaches [17] and Bayesian behavior modeling [18]. In addition, a variety of algorithms have been proposed for planning paths for automobiles for navigation outside of road conditions and traffic rules [19]. Several techniques have been proposed to specifically avoid hazards while remaining in a target lane. These techniques can be coupled with a path planner to avoid vehicles [20] and other hazards in the ego-vehicle's lane [7].

Many continuous approaches for collision-avoidance have been proposed based on spatial decomposition or velocity-space reasoning. Berg et al. [21] apply velocity-space reasoning with acceleration constraints to generate safe and collision-free velocities. Bareiss et al. [22] extend the concept of velocity obstacles into the control space to generate a complete set of collision free control inputs. Ziegler et al. [1] utilize polygonal decomposition of obstacles to generate blockages in continuous driving corridors. Sun et al. [23] demonstrate the use of prediction functions and trajectory set generation to plan safe lane-changes.

Modeling Traffic Rules: Aside from planning the appropriate paths to avoid collisions, autonomous vehicles must also follow applicable laws and traffic norms. Techniques have been proposed to simulate typical traffic behaviors in traffic simulation such as Human Driver Model [24] and data-driven models such as [25]. An extensive discussion on techniques to model these behaviors in traffic simulation can be found in [26].

Autonomous Driving Systems: Many autonomous systems have been demonstrated that are able to navigate an autonomous vehicle in traffic along a specific route. Ziegler et al. [3] demonstrated an autonomous vehicle which drove the historic Bertha Benz route in southern Germany. They use a conservative navigation approach, which specifically encodes *lanelets* for lane changing and does not account for dynamic lane changes. In contrast, our algorithm allows the vehicle to change lanes when our maneuver optimization method deems it appropriate and does not rely on pre-encoded changes. Geiger et al. [27] demonstrate a planning and control framework that won the Grand Cooperative Driving Challenge in 2011. This vehicle was designed for platooning and employed controls over acceleration only. Our navigation algorithm plans maneuvers using both steering and acceleration to operate in more generic traffic scenarios. The DARPA Urban Grand Challenge included a number of autonomous vehicle navigating examples of driving scenarios [28], [29]. While overtaking was allowed as an intended capability in these systems, the vehicles were not evaluated in dense, high-speed traffic conditions where the benefits of lane changes

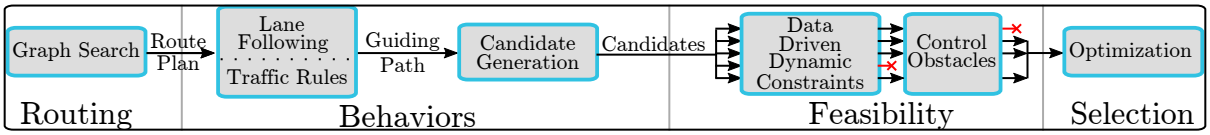

Fig. 1. **Algorithm Pipeline:** Our autonomous vehicle planning algorithm operates in several sequential steps. First, a route is planned using graph-search over the network of roads. Secondly, traffic and lane-following rules are combined to create a guiding trajectory for the vehicle for the next planning phase. This guiding trajectory is transformed to generate a set of candidate control inputs. These controls are evaluated for dynamic feasibility using our data driven vehicle dynamics modeling and collision-free navigation via extended control obstacles. Those remaining trajectories are evaluated using our optimization technique to determine the most-appropriate set of controls for the next execution cycle.

could be demonstrated.

III. Problem Space

In this section, we introduce the notation, the kinematic and dynamics model of the car and the state space of the vehicle in terms of both physical configuration and behavior space.

A. Vehicle State Space

We represent the kinematic and dynamic constraints of the vehicle separately. In terms of trajectory planning, steering and throttle controls that could lead to skidding or a loss of control are first excluded in our dynamics model (see section IV-F) and future trajectories are computed according to our vehicle kinematic model described in equation (1).

We extend the simple-car kinematic model [10], [30]. The vehicle has three degrees of freedom in a planar coordinate space. These are the position of the center of mass $\vec{\boldsymbol{p}} = (p_x, p_y)$, and the current heading or orientation θ. We represent the speed of the vehicle as v and steering as ϕ. L_f and L_r represent the distance from the center of mass to the front and rear axles, respectively. The geometry of the ego-vehicle is represented as $\mathcal{O}_e$.

The vehicle has two degrees of control, throttle (u_t) and steering (u_ϕ). We define throttle $-1 \leq u_t \leq 1$, where -1 indicates maximum braking effort for the vehicle and 1 represents maximum throttle. $-1 \leq u_\phi \leq 1$ describes the steering effort from $-\phi_{max}$ to ϕ_{max}.

We also use acceleration and steering functions, $A(v, u_t)$ and $\Phi(v, u_s)$, respectively, which describe the relationship between the vehicle's speed, steering, and control inputs and its potential for changes in the acceleration and steering (see section IV-F). A and Φ can be chosen to be constants in the simplest model, or may be represented using complex functions corresponding to tire dynamics and load transfer. We describe our choice for A and Φ in section IV-F. The vehicle's motion can be described by:

$$\dot{p}_x = v\cos(\theta) \qquad \dot{p}_y = v\sin(\theta) \qquad \dot{\theta} = \frac{\tan(\phi)}{L_f + L_r} v \tag{1a}$$

$$\dot{v} = A(v, u_t) \qquad \dot{\phi} = \Phi(\phi, u_s) \tag{1b}$$

In addition to the physical state of the vehicle, we describe its behavior b as a label from a set of all behaviors $\mathcal{B}$, such as driving straight, turning left, merging right, etc. The behavior state is used to modify parameters of each stage of the algorithm. Each behavior state can encode a set of weights of the maneuver optimization function, bias the generation of a guiding path, and adjust the sampling bias of our control-obstacle approximation and acceleration when necessary (see section IV-A). The full state of a vehicle is defined as $X_e = \{p_x, p_y, v, \phi, u_t, u_\phi, b\}$. The vehicle updates its plan at a fixed planning rate Δt. At each planning step, the vehicle computes a target speed v' and target steering ϕ' to be achieved by the control system. We refer to equation (1) compactly as the *state evolution function* $X_{t+\Delta t} = q(X_t, u, t)$. We also define a function $S(u, X)$ which determines if a set of controls is feasible. Given the current state of the vehicle, $S(u, X)$ will return false if the given input u will cause a loss of traction or control. We describe this function further in section IV-F.

B. Sensing and Perception

We assume a sensing module is available for the vehicle that is capable of providing information regarding the surrounding environment. For each lane on a road, the sensing module provides an approximation of the center line of the lane, l. The sensing module also provides the closest point on the lane center to the ego-vehicle, $\vec{\boldsymbol{l}}_p$, and a reasonable value of the friction coefficient μ. Recent work has presented approaches to evaluate μ from sensor data [31]. Our navigation algorithm utilizes the set of nearby vehicles, pedestrians, bicycles, or other obstacles, collectively referred to as neighbors, N within the sensing range. For each neighbor $n \in N$, the sensing system provides the neighbor's shape, $\mathcal{O}_n$, position, $\vec{\boldsymbol{p}}_n$, and velocity $\vec{\boldsymbol{v}}_n$. Moreover, the sensing module provides the lane l_n, acceleration $\dot{v}_n$, and rate of turn, $\dot{\theta}$ for the neighbor. We define a set of neighbor types, $\mathcal{T}_n$, including vehicle, pedestrian, cyclist, and obstruction. Each neighbor is assigned a type $\mathcal{T}_n$ corresponding to the detected neighbor type. The complete state of a neighbor is denoted as $X_n = \{\vec{\boldsymbol{p}}_n, \vec{\boldsymbol{v}}_n, l_n, \dot{v}_n, \dot{\theta}_n\}$

IV. Navigation Algorithm

In this section, we describe our navigation algorithm. Our algorithmic approach operates in four sequential stages, shown in Fig. 1. First, a route is constructed over the space of roads in the environment. Secondly, a *Guiding Path* that follows the current lane is computed that provides input to the collision-avoidance and optimization-based maneuver stages. The collision avoidance stage determines the set of feasible *candidate controls* that represent dynamically feasible, collision-free controls for the vehicle. Finally, a new control is chosen for the vehicle based on the optimization-based maneuver function.

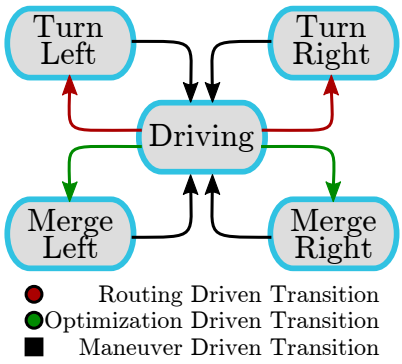

Fig. 2. **Finite State Machine:** We highlight different behavior states that are determined by the routing and optimization algorithms. When executing turns, the routing algorithm transitions the behavior state to a turning state. When the optimization-based maneuver algorithm plans a lane change, the behavior state is transitioned to merging.

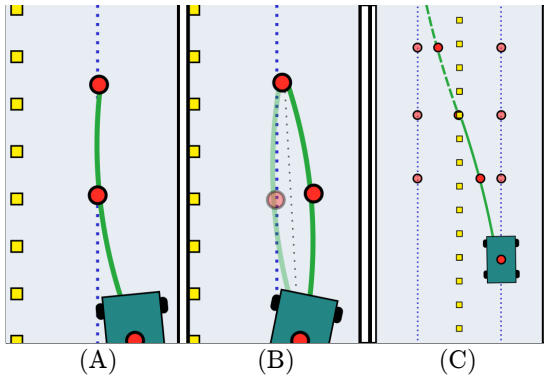

Fig. 3. **Guiding Path Computation:** The vehicle computes a guiding path to the center of its current lane based on a circular arc. **(A):** When the vehicle tracks a path off the center of its current lane, the guiding path leads it smoothly back to center. **(B):** In cases where the guiding path represents abrupt changes to heading, the center point is reflected about the axis formed by the car's position and the final waypoint. **(C):** In the case of lane changes, the guiding path is computed by a weighted average of the waypoints on the departure and destination lanes.

A. Route Choice and Behavior State

Our navigation algorithm performs several steps in a sequential manner. In the first step, a global route for the vehicle to follow to the goal is determined. This step is performed only once unless special conditions (e.g., missing a turn) force the vehicle to recompute a route. The ego-vehicle is provided a connected graph of roads in the environment from a GIS database. Each road in the graph contains information on the number and configuration of lanes in the road and the speed limits. When a destination is chosen, we use A* search to compute the shortest route to the goal and construct a route plan.

Each step of the route plan encodes how the vehicle transitions from one road to the next. We denote these as *road-transition maneuvers*. A road-transition maneuver consists of the valid source lanes, valid destination lanes, the position along the road at which the maneuver begins, denoted $\vec{\boldsymbol{p}}_m$, and the behavior implied by the road transition. The set of behaviors includes merging, right turns, left turns, and driving straight. Once the road-change maneuver is completed, the vehicle navigates along the lanes of the new road until the next maneuver node is reached. Lane changes are not encoded in the maneuver nodes, but they are performed implicitly based on the optimization function described in section IV-E.

The behavior state of the vehicle is described by a finite-state machine shown in figure 2. It is used to restrict potential control decisions and adjust the weight of the cost-function for specific maneuvers, such as turning. This allows our algorithm to force the vehicle to be more conservative when performing delicate maneuvers. For example, the valid steering space is constrained in turns to guarantee that the vehicle moves closely along the center line.

B. Guiding Path

In order to perform trajectory planning, the ego-vehicle computes a set of waypoints along the center-line of its current lane at fixed time intervals, and these waypoints represent the expected positions for a planning horizon, τ. We represent these waypoints as $\vec{\boldsymbol{w}}_1 - \vec{\boldsymbol{w}}_k$. Using its own position, the median point, and the final waypoint $(\vec{\boldsymbol{p}}, \vec{\boldsymbol{w}}_{\frac{k}{2}}, \vec{\boldsymbol{w}}_k)$, we compute a circular arc on the road plane which sets the initial target speed and steering, v' and ϕ' respectively, and acts as the guiding path for the next planning phase. We use circular arc approximations because they implicitly encode the radius of curvature needed for slip computation making it easy to check whether the dynamic constraints are violated.

Absent discontinuities in the center-line of the lane, the guiding arcs exhibit first-order C^1 continuity. Figure 3(a) demonstrates a guiding arc constructed for a sample lane.

We constrain the arcs to lie within the first two quadrants of the circle that is represented by three waypoints. In cases when the vehicle's trajectory tracks away from the center of the lane, e.g. during collision avoidance maneuvers, this constraint may be violated as shown in figure 3(b). In such cases, the point $\vec{\boldsymbol{w}}_{\frac{k}{2}}$ is reflected about the axis formed between $\vec{\boldsymbol{p}}$ and $\vec{\boldsymbol{w}}_k$ to correct the arc angle. In case of lane changing, waypoints are constructed from a weighted average of points sampled ahead on both the departure lane and the destination lane. Figure 3(c) demonstrates a set of lane-change arcs.

Given a guiding path, a target steering, ϕ' is computed from equation (1a). The radius of the arc, r, is substituted into equation (8) to determine the maximum safe speed for the current road curvature. A target speed, v', is computed from the minimum value of the current speed limit and the maximum safe speed. The target steering and speed form the basis of the control-obstacle exploration in the subsequent stage.

1) Traffic Rules: Traffic rules such as stopping at red lights are encoded in our algorithm. When choosing a target speed v', the sensing system is referenced to determine if an intersection is being approached and whether the vehicle needs to stop at the intersection. In cases where the vehicle

must stop, the edge of the intersection is used to compute a stopping point and v' is set to the speed that will reach the stopping point at τ seconds. In case of stoplights, the green light signals v' to return to its original value.

In the case of stops with continuous cross-traffic, the vehicle waits until the collision-avoidance algorithm indicates safety. This is accomplished by limiting the potential speed controls the vehicle may choose. When waiting for cross-traffic, the vehicle will stop until its guiding path is determined to be safe. In the case of all-way stops, the vehicle maintains a queue of vehicle arrival order, but defers to other drivers if they enter the intersection out of turn.

Although merges are not specifically encoded in transitions in the route plan, the vehicle is able to determine when merging is safe through the collision-avoidance and optimization stages of the algorithm. A merge is note determined safe and appropriate unless it provides collision-free guarantees and respects safety and comfort costs detailed in section IV-E.

C. Collision Avoidance

We leverage the theory of *Control Obstacles* for collision avoidance [22]. Control Obstacles construct constraints in the control space and are an extension of acceleration-velocity obstacles [21]. For each neighbor of the ego-vehicle, n, we define the control obstacle for the neighbor as the union of all controls that could lead to collisions with the neighbor within the time horizon, τ. Given t, where $0 \leq t \leq \tau$, the relative position of the ego-vehicle and neighbor $\vec{\boldsymbol{p}}_{en}$ must remain outside the Minkowski Sum given by the formulation, which is defined as

$$\mathcal{O}_{en} = \mathcal{O}_n \oplus -\mathcal{O}_e. \tag{2}$$

The complete derivation for control obstacles can be found in [22].

In order to adapt to the autonomous vehicles, we modify the original control obstacle formulation [22] in the following manner: (1) We do not assume reciprocity in collision avoidance and the ego-vehicle must take full responsibility for avoiding collisions; (2) We do not assume the control inputs of other vehicles are observable, which is consistent with the first point; (3) We do not assume bounding discs for the neighboring entities, but rather a tight bounding rectangle. The Minkowski Sum for two convex polygons can be computed in linear time in the number of edges; (4) The new feasible control chosen does not correspond to the control that minimizes the deviation from v' and ϕ'. Rather it is the control that minimizes the objective function defined in section IV-E.

The union of all control-obstacles and the set of dynamically infeasible controls form the boundary of the space of collision-free controls for the ego-vehicle. As long as a new control set is chosen from outside the union of the control obstacles, the ego-vehicle will be collision free for the next τ seconds. In section VII, we detail how behavior prediction models can be incorporated to assume varying levels of reciprocity. This approach is conservative and it is possible that there may be no feasible solution. In that case, we reduce τ and search for a feasible solution.

D. Trajectory Sampling

Computing the exact boundary of the control obstacle is computationally expensive. Moreover, depending on the choice of A and Φ, the boundary computation will typically not have an analytical solution. In order to ensure that the vehicle can plan within a specific time bound, we use a sampling strategy around ϕ' and v' to determine a feasible control that the vehicle will adopt for the next τ seconds. Each sample is referred to as a *candidate control* and represented as u_c.

First, the closest collision-free velocity to v' is determined where $\phi = \phi'$ by forward projection. This represents the largest speed the vehicle could take without deviating from the center-line of its lane and is always included in the set of candidates. Next, we compute evenly spaced samples around the point (v', ϕ') in the control space. We also choose a set of samples around the prior step solution, ϕ_{t-1} and v_{t-1}, which allows the vehicle to explore minor deviations in trajectory. Samples near the prior solution facilitate lane-keeping and within-lane avoidance maneuvers.

For each neighbor $n \in N$, we compute a set of states for that neighbor for the next τ seconds by forward integration of $q(X_n, \emptyset, t)$. We assume that the neighboring vehicle will follow its current lane at the current speed and acceleration during this time interval. Otherwise, the neighbor is assumed to move along its current velocity $\vec{\boldsymbol{v}}_n$ with the current observed values of turning and acceleration, $\dot{\theta}$ and $\dot{v}_n$, respectively.

For each candidate control, u_c, we determine whether equation (8) is violated by the candidate control inputs and immediately discard it if that is the case. If not, the sample points are computed at even time intervals along $0 \leq t \leq \tau$ by forward integration of $q(X_t, u_c, t)$. For each position in time, $\vec{\boldsymbol{p}}_t$, we compute the relative position with each neighboring position at that time and determine if the relative position lies inside the Minkowski Sum. If so, we discard the candidate controls. After all the candidates are evaluated, the new control sequence is chosen by minimizing the objective function described in the subsequent section.

E. New Trajectory Computation

Once a set of suitable control candidates has been computed, the vehicle selects the valid controls that minimize the following cost function at each sample point $i \in I$:

$$C = \sum_{i=0}^{I} c_{path}(i) + c_{cmft}(i) + c_{mnvr}(i) + c_{prox}(i). \tag{3}$$

This function corresponds to producing paths which are comfortable for passengers, provide safe passing-distances from other vehicles, and respect the constraints of upcoming maneuvers the vehicle must perform. Each term consists of several cost evaluation functions, each with its own weight $e \in W$, which are described in the following sections.

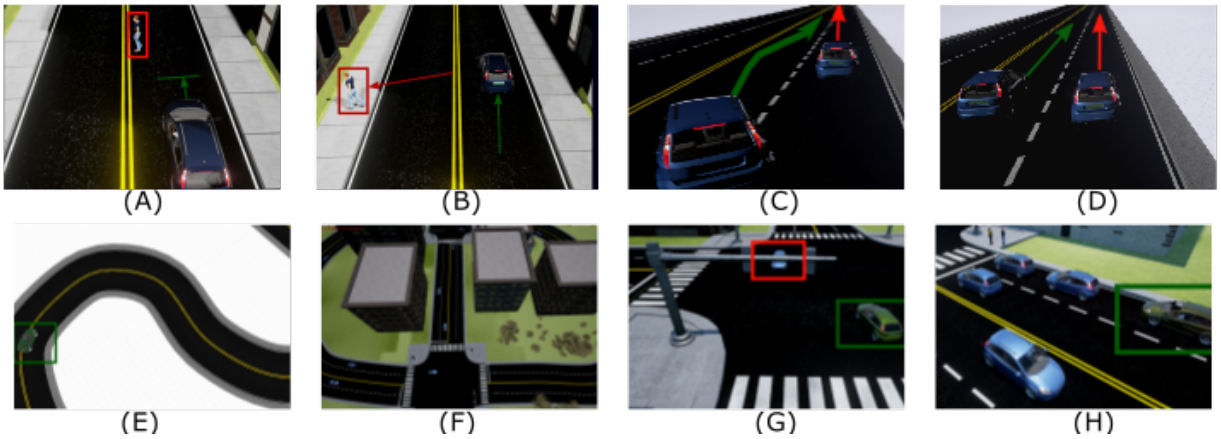

Fig. 4. **Results: (A) and (B)**: The ego-vehicle is forced to stop as a pedestrian enters the roadway during the **Jaywalking** benchmark due to the proximity costs. Once the pedestrian has moved away, the vehicle resumes its course. **(C) and (D)**: The ego-vehicle approaches a slower moving vehicle from behind. The path and maneuver costs drive the ego-vehicle to plan a lane-change around the slower vehicle. The trajectory of the ego-vehicle is shown in green. **(E)**: The Hatchback ego-vehicle during the **S-Turns** benchmark. The vehicle plans the highest speed it can safely maintain during the tight turns. Each ego-vehicle plans a different safe speed based on their data-driven vehicle dynamics functions. **(F)**: An overview of the **Simulated City** benchmark. The ego-vehicle navigates amongst typical traffic to a set of randomly assigned destinations. **(G)**: The ego-vehicle (outlined in green) yields to an oncoming vehicle (outlined in red) during the **Simulated City** benchmark. Once the vehicle clears the intersection, the ego-vehicle proceeds with a left turn. **(H)**: The ego-vehicle (outlined in green) stops in traffic waiting for a stoplight to change during the **Simulated City** benchmark.

1) Path Cost: c_{path} encodes costs associated with the vehicle's success at tracking its path and the global route. To compute this cost, we define two points. The target point, $\vec{\boldsymbol{p}}_{tar}$, represents the relative position the vehicle would achieve following the spline defined by its guiding path exactly at v'. Given the final sample, we project the vehicle's expected position at the final sample point onto $\vec{\boldsymbol{p}}_{tar}$, which we denote $\vec{\boldsymbol{p}}_{I,tar}$. The path cost is given by:

$$\begin{aligned} c_{path} &= c_{vel} + c_{drift} + c_{prog} \\ c_{vel} &= (v' - v)^2 \\ c_{drift} &= ||\vec{\boldsymbol{p}} - \vec{\boldsymbol{l}}_p||^2 \\ c_{prog} &= \frac{||p_{tar} - p_{I,tar}||}{||p_{tar}||} \end{aligned} \tag{4}$$

c_{vel} is the squared difference between desired speed and current speed and c_{drift} is the squared distance between the center line of the vehicle's lane and its current position. If the path crosses a lane boundary, c_{drift} is computed with respect to the new lane. c_{prog} represents the vehicle's desire to maximally progress along its current path. Candidates which reduce progress with respect to the guiding path are penalized. These terms drive the vehicle to choose trajectories that maximally progress the ego-vehicle along its computed route between steps. c_{prog} is only computed at the final sample point.

2) Comfort Costs: Comfort costs are computed similar to [1] and penalize motions which are uncomfortable for passengers in the vehicle. c_{accel} penalizes large accelerations and decelerations. c_{yawr} penalizes large heading changes and discourages sharp turning. The comfort costs are given as:

$$\begin{aligned} c_{cmft} &= c_{accel} + c_{yawr} \\ c_{accel} &= ||\dot{v}_i|| \\ c_{yawr} &= ||\dot{\theta}|| \end{aligned} \tag{5}$$

3) Maneuver Costs: The novel maneuvering cost function discourages lane-changes without excluding them and guides the vehicle to occupy the necessary lane for its next maneuver. The formulation is given as:

$$\begin{aligned} c_{mnvr} &= c_{lane} + c_{mdist} \\ c_{lane} &= 1 \cdot LaneChanged \\ c_{mdist} &= \frac{1}{\vec{\boldsymbol{p}} - \vec{\boldsymbol{p}}_m} \cdot WrongLane \end{aligned} \tag{6}$$

$LaneChanged$ is a boolean variable representing whether a candidate path crosses a lane boundary. $\vec{\boldsymbol{p}}_m$ is the position of the next maneuver change, e.g. the beginning of a right turn. This position is determined by the point of maneuver and starts in the desired lane for the maneuver. $WrongLane$ is a boolean that evaluates to true if the vehicle's lane does not match the lane for the next maneuver. If a candidate control is chosen where for some point $i \in I$, $LaneChanged$ evaluates to true, a lane change behavior is initiated in the finite state machine.

4) Proximity Costs: While the collision avoidance stage prevents the vehicle from colliding with neighbors, the proximity cost term is designed to prevent the vehicle from passing close to neighboring entities based on the identified type of the neighbor, T_n. This cost is represented as a cost distance term with exponential decay based on the relative positions of the ego-vehicle and its neighbor.

$$c_{prox} = \sum_{n=0}^{N} d(N_j, \vec{\boldsymbol{p}}) \tag{7}$$
$$d(N_n, \vec{\boldsymbol{p}}) = C_{type_{\mathcal{T}_n}} \cdot e^{-||\vec{\boldsymbol{p}}_n - \vec{\boldsymbol{p}}_e||}$$

C_{type} is a per-type constant cost value. C_{type} is larger for pedestrians and bicycles than for vehicles, and guides the ego-vehicle to pass those entities with greater distance.

F. Data-driven Vehicle Dynamics Model

In order to determine values for $A(v, u_t)$ and $\Phi(\phi, u_s)$, we use a data-driven approach to model the dynamics of the vehicle. For each ego-vehicle, data is collected by driving the vehicle from $v = 0$ to $v = v_{max}$ at the highest possible throttle without loss of traction. Similarly, for braking, the vehicle is decelerated from $v = v_{max}$ to $v = 0$ using the highest braking effort possible without loss of traction. Data is collected at 60Hz for these values: current speed, acceleration, and throttle/braking values. From these data, piecewise quadratic functions are constructed by least squares fitting to represent the maximum available acceleration and braking given the current vehicle state. These values also define thresholds for the control safety function $S(u, X)$.

We determine $\Phi(\phi, u_s)$ by fixing the vehicle's speed and collecting data for instantaneous changes to the steering angle for a given u_s. We construct a piecewise quadratic function by least-squares fitting to represent the vehicle's steering dynamics. Having the value of μ from the sensors, we determine the maximum feasible speed for a given curvature from the centripetal force equation:

$$v = \sqrt{\mu r g} \tag{8}$$

where r is the radius of curvature that is computed from equation (1a). By substituting equation (1a) into equation (8), and the angular velocity formula $v = \omega \cdot r$, we can determine feasible steering for a given speed as

$$\phi = tan^{-1}(\frac{(L_f + L_r) \cdot \mu \cdot g)}{v^2}). \tag{9}$$

Given the generated functions S, A, and Φ, the future path of the vehicle can be evaluated quickly for planning future controls.

G. Control Input

Once a new set of controls is chosen, they are input to the vehicle using a pair of PID controllers. One of the PID controllers drives the current speed to match the target speed. The second PID controller drives the current steering angle to match the target steering angle chosen by the optimization function. By limiting the choice of candidate controls to kinematically and dynamically feasible controls using our data-driven vehicle dynamics model, the PID controllers are sufficient to achieve the desired values.

V. Experimental Evaluation

In this section, we detail the evaluation scenarios for our navigation algorithm. Each scenario is chosen to test different aspects of the algorithm including response time, safety, and handling different traffic situations.

A. Ego-Vehicles

To demonstrate the generality of our approach, we tested each experimental scenario on each of three vehicles. Vehicle 1, the hatchback, has a mass of 1365 kg, a length of 3.8m, and a maximum steering angle of 60°. Vehicle 2, the sports car, has a mass of 1750 kg, a length of 4.6m, and a maximum steering angle of 63°. Vehicle 3, the SUV, has a mass of 1866 kg, a length of 4.8m, and a maximum steering angle of 55°.

B. Benchmarks

We conducted a series of simulations with each vehicle representing a variety of the challenging traffic scenarios an ego-vehicle will face while navigating city roads and highways.

Passing a bicycle: This scenario involves the ego-vehicle passing a bicycle on a four lane straight road. The vehicle should maintain a safe distance from the bicycle, changing lanes if possible to avoid the cyclist. We perform the evaluation twice, once featuring a vehicle in the adjacent lane preventing the vehicle from moving to avoid the cyclist without first adjusting its speed.

Jaywalking Pedestrian: This scenario features a pedestrian stepping into the road in front of the vehicle. The vehicle must react quickly to safely decelerate or stop to avoid the pedestrian.

Sudden Stop at High Speed: The vehicle must execute an emergency stop on a highway at high speeds when the vehicle in front of it stops suddenly. We evaluate this scenario in two conditions. First, we evaluate performance with no other traffic aside from the ego-vehicle and stopping vehicle. In this condition, swerving can be performed simply. Secondly, we evaluate this scenario with surrounding traffic, complicating any swerving maneuvers as the vehicle must account for nearby traffic.

High Density Traffic Approaching a Turn: This scenario features a four lane road with the ego-vehicle starting in a heavily congested outer lane. The ego-vehicle must make a turn at a stoplight ahead in the outer lane. To make optimal progress, the ego-vehicle must execute a lane change to the inner lane, but must return to the outer lane with sufficient time to execute the turn.

Car Suddenly entering Roadway: This scenario demonstrates the ego-vehicle traveling along a straight road at constant speed when a vehicle suddenly enters the roadway ahead of the vehicle and blocks the vehicle's path. The vehicle must decelerate and swerve to avoid colliding with the blocking vehicle. We demonstrate this scenario with the ego-vehicle travelling at 10, 30, and 50 mph and with the blocking vehicle obstructing either the right lane or both lanes.

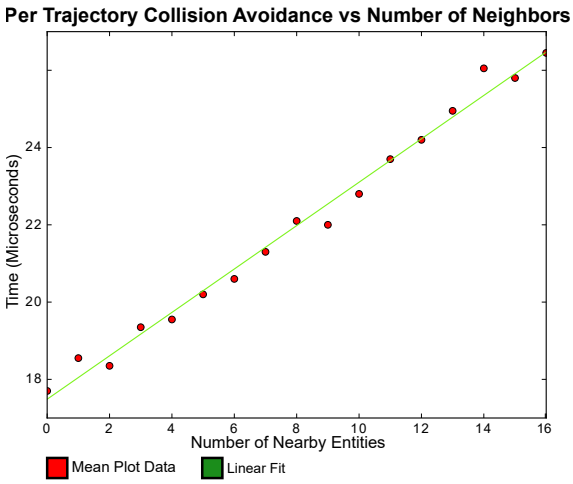

Fig. 5. **Collision Avoidance Timing:** We detail the relationship between the cost of collision checking for a trajectory and the number of nearby entities considered. We observe a linear relationship between collision checking and number of neighbors.

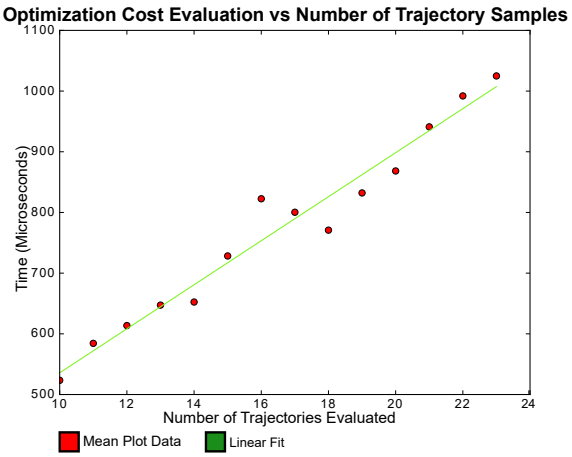

Fig. 6. **Cost Function Evaluation Timing:** The computational cost of computing the optimal trajectory for the vehicle varies linearly with the number of collision-free trajectories evaluated.

S-turns: We demonstrate the ego-vehicle navigating a set of tight alternating turns, or S turns. Each ego-vehicle computes a different safe speed depending on the specific kinematic and dynamic limits of the vehicle.

Simulated City: We demonstrate the ego-vehicle navigating to several key points in a small simulated city. The vehicle must execute lane changes to perform various turns as it obeys traffic laws and navigates to its goals. The vehicle encounters bicycles, pedestrians, and other vehicles as it navigates to its waypoints.

VI. Benchmark Results

We evaluated our navigation algorithm in these simulated scenarios. The algorithm can avoid tens of vehicles at interactive rates. As expected, the sports-car and the hatchback were able to maintain their preferred speeds more effectively in turns, whereas our SUV was forced to reduce speed. Each of the vehicles was able to pass other vehicles, pedestrians, and bicycles safely. In **Car Suddenly entering Roadway** scenario, we observed a greater tendency to swerve. We did not observe the ego-vehicle colliding with any of the simulated vehicles in traffic.

Figure 4 details some of the interesting behaviors we observed while testing our navigation algorithm. As expected, the ego-vehicle utilizes lane-changes to pass slower vehicles when no traffic is imposing. In traffic, the ego-vehicle slows down until it is safe to pass in the adjoining lane. When interacting with pedestrians, the high proximity cost discourages the vehicle from changing lanes as the pedestrian passes, and the vehicle instead waits until the pedestrian has moved considerably.

A. Timing Results

We collected data from a simulation designed to gradually increase the density of other vehicles encountered by our car. Figure 5 demonstrates the relationship between collision avoidance cost for a specific trajectory sample and the number of nearby vehicles and entities. We observe the cost of collision avoidance grows linearly in the number of neighbors.

Figure 6 demonstrates the relationship between the number of trajectories sampled and the computational expense of optimization evaluation. The observed relationship is linear with greater variance than that of collision avoidance computation. The overall computation time for a typical navigation update including guiding path computation, control-obstacle sampling, collision-avoidance and cost evaluation is on the order of milliseconds, typically between 1 and 2 milliseconds. This suggests that the cost of the algorithm is dominated by the optimization time.

VII. Conclusion and Limitations

We present, AutonoVi, a navigation algorithm for autonomous vehicles. Our approach uses a data-driven vehicle dynamics model and optimization-based maneuver planning to compute safe, collision free trajectories with dynamic lane changes under typical traffic conditions. We have demonstrated our algorithm on a varied set of vehicles under varying dense and sparse traffic conditions with pedestrians and cyclists. We have also demonstrated that our vehicles follow traffic laws, and utilize both braking and steering simultaneously when avoiding collisions. We highlight many benefits over prior methods in our simulations.

Our approach has some limitations. First, though our introduction of the data-driven dynamics functions A, Φ, and S generalize to arbitrary levels of underlying dynamics complexity, our current approach requires computing new vehicle dynamics functions for different values of μ. We will address this limitation in future work by learning a transfer function between various road frictions to produce more general data-driven vehicle dynamics functions. In addition, we have assumed perfect sensing in the current technique and it would be useful to take into account sensing errors and uncertainty in our approach. These could be based on relying on predictive behavior models to overcome imperfect state estimations for neighboring entities [18], [23]. With prediction, our control obstacles could anticipate levels of

reciprocity from predictable vehicles. We will also explore whether the use of circular arcs may is appropriate for vehicles with substantially different geometries, such as trucks pulling large trailers. It is unclear if our kinematic models and data-driven profiles will navigate large vehicles safely. We will also like to incorporate real-world driving patterns and cultural norms to improve our navigation algorithm. This will include choosing optimal weights for the navigation algorithm using available training data.

REFERENCES

[1] Julius Ziegler, Philipp Bender, Thao Dang, and Christoph Stiller. Trajectory planning for Bertha - A local, continuous method. *The International Journal of Robotics Research*, 35(April):450–457, 2014.

[2] Pendleton et al. Perception, Planning, Control, and Coordination for Autonomous Vehicles. *Machines*, 5(1):6, 2017.

[3] Ziegler et al. Making bertha drive-an autonomous journey on a historic route. *IEEE Intelligent Transportation Systems Magazine*, 6(2):8–20, 2014.

[4] F. Borrelli, P. Falcone, T. Keviczky, J. Asgari, and D. Hrovat. MPC-based approach to active steering for autonomous vehicle systems. *International Journal of Vehicle Autonomous Systems*, 3(2/3/4):265, 2005.

[5] Andreas Eidehall, Jochen Pohl, Fredrik Gustafsson, and Jonas Ekmark. Toward autonomous collision avoidance by steering. *IEEE Transactions on Intelligent Transportation Systems*, 8(1):84–94, 2007.

[6] Martin Distner, Mattias Bengtsson, Thomas Broberg, and Lotta Jakobsson. City Safety- A System Addressing Rear-End Collisions At Low Speeds. *21st Enhanced Safety Vehicles Conference*, pages 1–7, 2009.

[7] Turri et al. Linear model predictive control for lane keeping and obstacle avoidance on low curvature roads. *IEEE Conference on Intelligent Transportation Systems, Proceedings, ITSC*, (Itsc):378–383, 2013.

[8] Christos Katrakazas, Mohammed Quddus, Wen-Hua Chen, and Lipika Deka. Real-time motion planning methods for autonomous on-road driving: State-of-the-art and future research directions. *Transportation Research Part C: Emerging Technologies*, 60:416–442, 2015.

[9] Mohammad Saifuzzaman and Zuduo Zheng. Incorporating human-factors in car-following models: A review of recent developments and research needs. *Transportation Research Part C: Emerging Technologies*, 48:379–403, 2014.

[10] Steven M. LaValle. *Planning Algorithms*. Cambridge University Press, New York, NY, USA, 2006.

[11] James Reeds and Lawrence Shepp. Optimal paths for a car that goes both forwards and backwards. *Pacific Journal of Mathematics*, 145(2):367–393, 1990.

[12] Donald L. Margolis and Jahan Asgari. Multipurpose models of vehicle dynamics for controller design. In *SAE Technical Paper*. SAE International, 09 1991.

[13] Sascha Kolski, D. Ferguson, Mario Bellino, and Roland Siegwart. Autonomous Driving in Structured and Unstructured Environments. In *2006 IEEE Intelligent Vehicles Symposium*, pages 558–563. IEEE, 2006.

[14] Kuwata et al. Real-time motion planning with applications to autonomous urban driving. *IEEE Transactions on Control Systems Technology*, 17(5):1105–1118, 2009.

[15] Jason Hardy and Mark Campbell. Contingency Planning Over Probabilistic Obstacle Predictions for Autonomous Road Vehicles. *IEEE Transactions on Robotics*, 29(4):913–929, aug 2013.

[16] Enric Galceran, Ryan M Eustice, and Edwin Olson. Toward Integrated Motion Planning and Control using Potential Fields and Torque-based Steering Actuation for Autonomous Driving. *IEEE Intelligent Vehicles Symposium*, (Iv), 2015.

[17] Dorsa Sadigh, Shankar Sastry, Sanjit A Seshia, and Anca D Dragan. Planning for Autonomous Cars that Leverage Effects on Human Actions. *Proceedings of Robotics: Science and Systems*, 2016.

[18] Enric Galceran, Alexander G Cunningham, Ryan M Eustice, and Edwin Olson. Multipolicy Decision-Making for Autonomous Driving via Changepoint-based Behavior Prediction. *Robotics: Science and Systems*, 2015.

[19] Julius Ziegler, Moritz Werling, and Joachim Schröder. Navigating car-like robots in unstructured environments using an obstacle sensitive cost function. *IEEE Intelligent Vehicles Symposium, Proceedings*, pages 787–791, 2008.

[20] H. Fritz, A. Gern, H. Schiemenz, and C. Bonnet. CHAUFFEUR Assistant: a driver assistance system for commercial vehicles based on fusion of advanced ACC and lane keeping. *IEEE Intelligent Vehicles Symposium, 2004*, (Vc):495–500, 2004.

[21] Jur Van Den Berg, Jamie Snape, Stephen J. Guy, and Dinesh Manocha. Reciprocal collision avoidance with acceleration-velocity obstacles. *Proceedings - IEEE International Conference on Robotics and Automation*, pages 3475–3482, 2011.

[22] Daman Bareiss and Jur van den Berg. Generalized reciprocal collision avoidance. *The International Journal of Robotics Research*, 34(12):1501–1514, oct 2015.

[23] Hao Sun, Weiwen Deng, Sumin Zhang, Shanshan Wang, and Yutan Zhang. Trajectory planning for vehicle autonomous driving with uncertainties. *ICCSS 2014 - Proceedings: 2014 International Conference on Informative and Cybernetics for Computational Social Systems*, pages 34–38, 2014.

[24] Martin Treiber, Arne Kesting, and Dirk Helbing. Delays, inaccuracies and anticipation in microscopic traffic models. *Physica A: Statistical Mechanics and its Applications*, 360(1):71–88, 2006.

[25] Peter Hidas. Modelling vehicle interactions in microscopic simulation of merging and weaving. *Transportation Research Part C: Emerging Technologies*, 13(1):37–62, 2005.

[26] Bo Chen and Harry H. Cheng. A review of the applications of agent technology in traffic and transportation systems. *IEEE Transactions on Intelligent Transportation Systems*, 11(2):485–497, 2010.

[27] Geiger et al. Team AnnieWAY's Entry to the 2011 Grand Cooperative Driving Challenge. *IEEE Transactions on Intelligent Transportation Systems*, 13(3):1008–1017, 2012.

[28] Chris Urmson and et al. Autonomous Driving in Traffic: Boss and the Urban Challenge. *Al Magazine*, 30(2):17–28, 2009.

[29] Bacha et al. Odin: Team VictorTango's entry in the DARPA Urban Challenge. *Journal of Field Robotics*, 25(8):467–492, aug 2008.

[30] Jean paul Laumond, S. Sekhavat, and F. Lamiraux. Guidelines in nonholonomic motion planning for mobile robots. In *ROBOT MOTION PLANNNING AND CONTROL*, pages 1–53. Springer-Verlag, 1998.

[31] Fredrik Gustafsson. Slip-based tire-road friction estimation. *Automatica*, 33(6):1087–1099, 1997.