Contingency Planning Over Probabilistic Obstacle Predictions for Autonomous Road Vehicles

Jason Hardy and Mark Campbell, *Member, IEEE*

***Abstract*—This paper presents a novel optimization-based path planner that is capable of planning multiple contingency paths to directly account for uncertainties in the future trajectories of dynamic obstacles. This planner addresses the particular problem of probabilistic collision avoidance for autonomous road vehicles that are required to safely interact, in close proximity, with other vehicles with unknown intentions. The presented path planner utilizes an efficient spline-based trajectory representation and fast but accurate collision probability bounds to simultaneously optimize multiple continuous contingency paths in real time. These collision probability bounds are efficient enough for real-time evaluation, yet accurate enough to allow for practical close-proximity driving behaviors such as passing an obstacle vehicle in an adjacent lane. An obstacle trajectory clustering algorithm is also presented to enable the path planner to scale to multiple-obstacle scenarios. Simulation results show that the contingency planner allows for a more aggressive driving style than planning a single path without compromising the overall safety of the robot.**

***Index Terms*—Artificial intelligence reasoning methods, collision avoidance, contingency planning, field robots, nonholonomic motion planning.**

I. Introduction

CURRENT autonomous vehicles are adept at identifying obstacles and planning routes and interacting in controlled environments with well-defined rules, as demonstrated in the 2007 DARPA Urban Challenge (DUC) [1]. The DUC also demonstrated that there remains a large gap in problem-solving and decision-making capabilities between autonomous vehicles and human drivers. A major reason for this gap is the inability of current algorithms to adequately identify, predict, and utilize obstacle intent [2]. Inferring obstacle intent, such as intended obstacle goals, enables an autonomous vehicle to anticipate the future motion of dynamic obstacles, which can then be used to improve the safety and robustness of the robot's motion planning.

Manuscript received July 29, 2012; revised November 29, 2012; accepted March 12, 2013. Date of publication April 12, 2013; date of current version August 2, 2013. This paper was recommended for publication by Associate Editor V. Isler and Editor G. Oriolo upon evaluation of the reviewers' comments. This work was supported in part by Army Research Office under Grant W911NF-09-1-0466, with Dr. R. Zachery as Program Manager, and in part by Government under and awarded by the Department of Defense, the Air Force Office of Scientific Research, and the National Defense Science and Engineering Graduate (NDSEG) Fellowship 32 CFR 168a.

The authors are with the Department of Mechanical and Aerospace Engineering, Cornell University, Ithaca, NY 14853 USA (e-mail: jsh256@cornell.edu; mc288@cornell.edu).

Color versions of one or more of the figures in this paper are available online at http://ieeexplore.ieee.org.

Digital Object Identifier 10.1109/TRO.2013.2254033

Previous research in this area has contributed potential solutions to two primary parts of the dynamic obstacle avoidance problem: 1) predicting future obstacle motion and 2) path planning in dynamic environments using predicted obstacle motions. Predicting future obstacle motion is a problem that has been explored in a variety of fields and contexts. Kushleyev and Likhachev [3] perform predictions of dynamic obstacles using constant velocity and constant curvature dynamics models. This approach is computationally efficient but ignores potentially useful structural information in the environment. Miura and Shirai [4] infer the potential paths that an obstacle might take using a tangent graph of the environment and a 1-D model of velocity variance along each potential path. This approach utilizes structural information in the environment, but oversimplifies obstacle motion and forms overconfident predictions. Hwang and Seah [5] present an algorithm to infer the intent of other airplanes that are based on likely flight plans, and to probabilistically model their future motion using bounded process noise and probabilistic models of transitions between maneuvers. This approach is useful for air vehicles but not readily adaptable for road vehicles which have fewer preset maneuvers and much more restrictive environmental constraints.

Nonparametric learning methods have also been applied to obstacle prediction. Tay and Laugier [6] use Gaussian processes to model multiple possible future trajectories that are based on observed training data. Joseph *et al*. [7] and Kim *et al*. [8] both use Gaussian processes to model trajectory derivatives for road vehicles. Bennewitz *et al*. [9] cluster observed trajectories of human participants through an environment into nominal trajectories and develop a hidden Markov model to predict mode transitions. These nonparametric approaches can produce detailed probabilistic trajectories, but they generally require retraining for each new environment and obstacle type.

A variety of path-planning strategies have been developed to incorporate dynamic obstacle predictions. Dynamic obstacles present a significant planning challenge because they increase the dimensionality of the planning problem. Simple dynamic planning approaches involve reducing dimensionality by projecting the predicted obstacle trajectories onto the static environment or decoupling the problem into a static motion planning problem and a 1-D dynamic velocity planning problem [10]. Approaches which attempt to solve the full dimensional dynamic planning problem typically avoid directly optimizing over the continuous search space through the use of probabilistic sampling or through discretization of the search space. Aoude *et al*. [11] use a predicted tree of reachable trajectories for each dynamic obstacle to bias a rapidly-exploring random tree (RRT)-based path planner. Kushleyev and Likhachev [3]

use a time-dependent variation of A*, which searches over a set of predefined motion primitives to avoid collisions with dynamic obstacles. These discrete and sampling-based approaches provide only an approximate optimal path due to discretization error and limited sample size. Additionally, these algorithms ignore the fact that multiple motion predictions for a single obstacle, due to uncertainties in the obstacle's intent, are mutually exclusive events.

Obstacle prediction has also been applied directly to the problem of autonomous vehicle navigation. During the DUC, the Tartan Racing team showed that it is possible to deduce a small set of short-term goal hypotheses for dynamic obstacles that are based on the surrounding road configuration [12]. The possible goals for another vehicle approaching a four-way intersection, for example, would be to stop, continue straight, turn left, or turn right. These inferred goal hypotheses are then used to make deterministic predictions about each dynamic obstacle's possible future trajectory. The simulations that are presented in [12] and the success of Tartan Racing's BOSS robot in the DUC [13] make a compelling case for the inclusion of obstacle intent in the planning process. However, the use of deterministic motion predictions in [12] can make the collision probability assumptions of the robot overconfident.

To provide better dynamic planning capabilities for autonomous road vehicles, a path-planning formulation is presented, which incorporates probabilistic obstacle predictions to enable efficient generation of a safe set of contingency paths in dynamic environments. Goal hypotheses are formed for each dynamic obstacle based on the topology of the road network, and the obstacles' state distributions are predicted forward in time using a probabilistic motion model. Path optimization is based on a nonlinear, constrained, numerical optimization formulation, extending the work in [14] and [15]. Formulating the path-planning problem as a constrained numerical optimization problem is a well-studied path optimization technique. Full nonlinear optimization is used by Milam *et al.* [16] who generate nonlinear trajectories using sequential quadratic programming, while linear and quadratic path optimization formulations have been developed and implemented by Blackmore *et al.* [17] and Bellingham *et al.* [18]. A key advantage of the numerical optimization formulation in this paper is its computational efficiency, enabling the dynamic planning problem to be solved in real time. This efficiency is achieved by 1) adopting an efficient spline-based trajectory representation, 2) solving the planning problem over a limited planning horizon, and 3) using fast but accurate collision probability bounds. While limiting the planning horizon prevents true global planning, it has been shown to be an effective navigation strategy when used in conjunction with a higher level global route planner [14].

The key novel contributions of this paper include 1) the simultaneous optimization of partially shared contingency trajectories over the robot's continuous planning space in order to accurately handle mutually exclusive obstacle predictions, 2) a computationally efficient and accurate upper bound on point-wise collision probability between two polygonal objects with uncertain relative position and orientation, 3) a spline-based trajectory representation and associated cost function and constraint definitions to express the contingency path-planning problem using only a small number of optimization variables, and 4) an obstacle trajectory clustering algorithm designed to reduce planning complexity by capturing the most important mutually exclusive obstacle decisions using a limited number of obstacle trajectory clusters. Spline-based trajectory representations have been used before, such as in [19]; here, cubic splines are employed in a unique way to enable multiple contingency trajectories to be efficiently optimized in unison. The contingency planning approach that is presented in this paper was introduced in [20], and the obstacle trajectory clustering algorithm was introduced in [21]. This paper includes new simulation results and performance analysis, a sensitivity analysis of the collision probability threshold constraint, and expanded analysis of the collision probability bound.

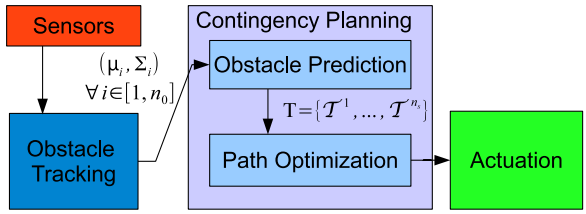

Fig. 1. Block diagram of the contingency planning architecture.

This paper is arranged as follows. Section II introduces the path optimization framework and proposes an algorithm to probabilistically predict the motion of dynamic obstacles. Section III presents the spline-based path representation, the path optimization algorithm, and the collision probability bound methods. Section IV presents a method to cluster obstacle trajectory predictions to improve scaling. Section V present simulation results, and Section VI provides conclusions.

II. Contingency Planning Architecture and Obstacle Prediction

The system architecture for the proposed contingency planner is shown in Fig. 1. In this architecture, the contingency planning procedure consists of an obstacle prediction phase followed by a path optimization phase. The contingency planning approach presented in this paper assumes that the problems to identify unique dynamic obstacles, track these obstacles between sensor measurements, and estimate the state of these obstacles have been resolved to a sufficient degree using an external tracking algorithm; paper [22] is one example of such an algorithm. As such, current Gaussian obstacle state estimates, which are defined by their mean and covariance, i.e., (μ_i, Σ_i) for the ith of n_o dynamic obstacles, are assumed to be known from the obstacle tracker at the beginning of each contingency planning cycle.

At the start of the obstacle prediction phase, a set of discrete goal modes $\mathcal{G}_i$ is inferred for each obstacle based on a known topological representation of the local road network and the planning horizon of the robot. For example, a dynamic obstacle that approaches a four-way intersection might use $\mathcal{G} = \{\text{stop, go straight, turn left, turn right}\}$. This concept of goal modes is very general and can be extended to encode

other sources of obstacle uncertainty, such as including multiple motion models to represent "good," "aggressive," or "impaired" drivers or to represent obstacle classification uncertainty such as "car" versus "bicycle."

When a new obstacle is observed, a new set of goal modes is inferred based on the topology of the surrounding road network. For this paper, all possible goal modes for a newly observed obstacle are initialized with equal probability. However, *a priori* information such as whether the obstacle is in a turn lane, along with any observed turn signals or hand signs, can also be used to more intelligently initialize these goal mode probabilities. Once a goal mode becomes infeasible, it is dropped and the probabilities for the other goal modes are renormalized. Similarly, when a tracked obstacle prediction reaches a new decision point, such as a new intersection, its goal mode is split into a set of new goal modes that are based on the road network topology.

The state distributions for each obstacle are predicted forward in time for each possible goal mode using a probabilistic motion model. The contingency planning approach proposed here is general to a wide variety of obstacle prediction models. For example, the motion model can vary in complexity from a simple path following model that tracks the center of the driving lane to individual path planners that are designed to model the motion of each obstacle type. For linear motion models, this prediction can be performed using the prediction step of the standard Kalman filter to yield the desired predicted mean and covariance at each timestep. For more general nonlinear, nondifferentiable motion models, this prediction can be performed using an algorithm such as the sigma point transform [23].

Formally, let $\mathcal{T}$ represent a set of predicted obstacle trajectories. The set of trajectories $\mathcal{T}_i$ represents all the predicted trajectories for the ith obstacle corresponding to the goal modes in $\mathcal{G}_i$. Each predicted trajectory for obstacle i, $T \in \mathcal{T}_i$, is represented by a sequence of obstacle state distributions over N timesteps into the future:

$$T = \{(\mu_i, \Sigma_i), (\hat{\mu}_i, \hat{\Sigma}_i)_1, \ldots, (\hat{\mu}_i, \hat{\Sigma}_i)_N\} \tag{1}$$

where (μ_i, Σ_i) represents the current state estimate of obstacle i, and $(\hat{\mu}_i, \hat{\Sigma}_i)_k$ represents a predicted obstacle state distribution at future timestep k.

For multiple dynamic obstacles, the set of all possible permutations of predicted obstacle trajectories is defined as

$$\mathbf{T} = \mathcal{T}_1 \times \mathcal{T}_2 \times \ldots \times \mathcal{T}_{n_o} \tag{2}$$

where $\mathbf{T}$ is the Cartesian product of the predicted obstacle trajectory sets for all n_o obstacles. For a two-obstacle environment with $\mathcal{T}_1 = \{T_\mathrm{a}, T_\mathrm{b}\}$ and $\mathcal{T}_2 = \{T_\mathrm{c}, T_\mathrm{d}\}$, $\mathbf{T} = \mathcal{T}_1 \times \mathcal{T}_2 = \{\{T_\mathrm{a}, T_\mathrm{c}\}, \{T_\mathrm{a}, T_\mathrm{d}\}, \{T_\mathrm{b}, T_\mathrm{c}\}, \{T_\mathrm{b}, T_\mathrm{d}\}\}$. The set of trajectories $\mathcal{T}^j \in \mathbf{T}$ is used to represent the jth set of trajectory permutations, such that $\mathbf{T} = \{\mathcal{T}^1, \ldots, \mathcal{T}^{n_s}\}$, where $n_s = |\mathbf{T}|$ represents the cardinality of $\mathbf{T}$, which is defined as

$$n_s = \prod_{i=1}^{n_o} |\mathcal{T}_i| \,. \tag{3}$$

At the start of each new contingency planning cycle, updated state estimates for each obstacle, $\{(\mu_1, \Sigma_1)_1, \ldots, (\mu_{n_o}, \Sigma_{n_o})_1\}$, are obtained from the obstacle tracker, as shown in Fig. 1. These new state estimates are used to update the probability of each obstacle trajectory permutation, $p(\mathcal{T}^j)$:

$$p(\mathcal{T}^j)_1 = \frac{p(\mathcal{T}^j) \prod_{i=1}^{n_o} p((\mu_i, \Sigma_i)_1 | (\hat{\mu}_i, \hat{\Sigma}_i)_1^j)}{\sum_{\mathcal{T}^m \in \mathbf{T}} p(\mathcal{T}^m) \prod_{i=1}^{n_o} p((\mu_i, \Sigma_i)_1 | (\hat{\mu}_i, \hat{\Sigma}_i)_1^m)} \tag{4}$$

where $p((\mu_i, \Sigma_i)_1 | (\hat{\mu}_i, \hat{\Sigma}_i)_1^j)$ is the innovation likelihood of the updated state estimate for obstacle i, i.e., $(\mu_i, \Sigma_i)_1$, given the predicted state distribution from the previous timestep, i.e., $(\hat{\mu}_i, \hat{\Sigma}_i)_1^j$ from the trajectory prediction for obstacle i included in $\mathcal{T}^j$.

III. Path Optimization

The path-planning problem is formulated here as a general nonlinear constrained optimization problem:

$$\mathbf{h}_\mathrm{opt} = \arg\min_{\mathbf{h}} J(\mathbf{h}) \qquad C(\mathbf{h}) < 0 \tag{5}$$

where $\mathbf{h}$ is a parameter vector that defines a trajectory or set of contingency trajectories for the robot, and $J(\mathbf{h})$ is a predefined cost function over which the path is optimized. The constraint function $C(\mathbf{h})$ places constraints on the range of possible values that $\mathbf{h}$ can take. With this formulation, the path-planning problem can be solved using well-known convex optimization techniques, including interior point methods [24] and sequential quadratic programming methods [25]. For nonconvex cost function and constraint definitions, this approach risks convergence of the optimization to a suboptimal local minimum; however, feasible local minima still represent acceptable, drivable paths. The key advantage of this approach over contemporary heuristic path-planning approaches is that first- and second-derivative information from the cost function $J(\mathbf{h})$ and constraint function $C(\mathbf{h})$ is used to guide the path optimization search.

A. Contingency Planning Approach

The goal of contingency planning is to generate a set of planned paths that account for all possible evolutions of the robot's environment. Here, uncertainty in the future state evolution of the robot's environment is encoded as a set of mutually exclusive obstacle predictions, generated by the prediction approach that is outlined in Section II. In this framework, a separate contingency path is planned for each possible permutation of predicted obstacle trajectories, $\mathcal{T}^j$. The approach that is used here further constrains all contingency paths to share the same initial segment. By using a single shared path segment at the start of the path, the contingency planner can provide the robot with an immediate deterministic action; multiple distinct contingencies for the rest of the path are then used to account for uncertainty in the evolution of the robot's dynamic environment.

This partially shared contingency planning approach is preferable to alternate approaches, such as planning a single path which attempts to avoid all obstacle predictions simultaneously as shown in Fig. 2(a), or taking a weighted combination of independent contingency paths, as shown in Fig. 2(b). Planning

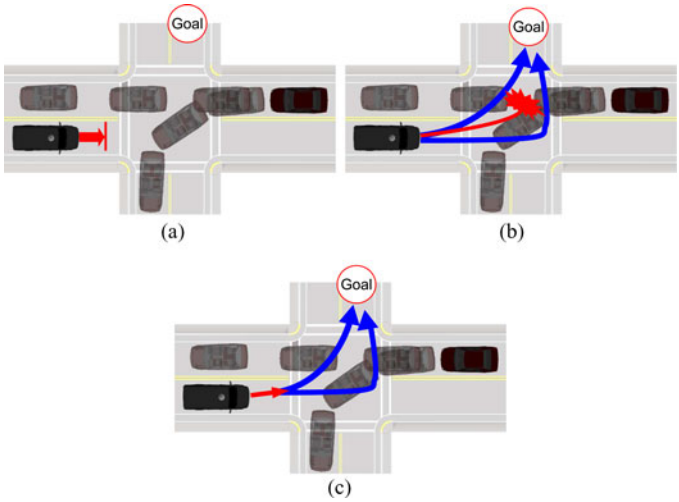

Fig. 2. Comparison of different strategies for path planning using mutually exclusive sets of obstacle predictions. (a) Single path. (b) Weighted combination. (c) Shared segment.

a single path is overcautious and ignores the fact that the obstacle goal sets are mutually exclusive, while taking a weighted combination of independent contingency paths offers no guarantees on the safety of the combined path.

Sharing the initial segment of each contingency path, as shown in Fig. 2(c), gives the robot a deterministic action to execute at the current timestep while maintaining the independence of future contingencies. In this contingency planning formulation, the path optimization cost function in (5) can be expanded as shown in

$$J(\mathbf{h}) = J_{\text{shared}}(\mathbf{h}, \mathbf{T}) + \sum_{\mathcal{T}^j \in \mathbf{T}} p(\mathcal{T}^j) J_{\text{conting}}(\mathbf{h}, \mathcal{T}^j) \quad (6)$$

where $J_{\text{shared}}(\mathbf{h}, \mathbf{T})$ is the cost for the shared path segment, and $J_{\text{conting}}(\mathbf{h}, \mathcal{T}^j)$ is the cost for the jth contingency path.

Both $J_{\text{shared}}(\mathbf{h}, \mathbf{T})$ and $J_{\text{conting}}(\mathbf{h}, \mathcal{T}^j)$ in (6) are formulated as a summation of weighted terms designed to penalize undesirable driving behaviors; terms include penalty functions on large changes in vehicle dynamics, on proximity to static and dynamic obstacles, and on the distance between the robot and its goal. Equations (7) and (8) show the component cost terms in J_{shared} and J_{conting} respectfully:

$$J_{\text{shared}} = J^{\text{dynamics}}_{\text{shared}} + J^{\text{static}}_{\text{shared}} + J^{\text{collision}}_{\text{shared}} \quad (7)$$

$$J_{\text{conting}} = J^{\text{dynamics}}_{\text{conting}} + J^{\text{static}}_{\text{conting}} + J^{\text{collision}}_{\text{conting}} + J^{\text{goal}}_{\text{conting}} \quad (8)$$

where $J^{\text{dynamics}}_{(\cdot)}$ contains terms that penalize large accelerations and large path curvatures, $J^{\text{static}}_{(\cdot)}$ penalizes proximity to static obstacles, $J^{\text{collision}}_{(\cdot)}$ penalizes high dynamic obstacle collision probabilities, and $J^{\text{goal}}_{(\cdot)}$ penalizes the distance between the end of a robot's planned path and its goal.

The rest of Section III presents the path parameterization and the path optimization cost terms and constraints in detail. Section III-B presents a novel spline-based path parameterization that efficiently encodes multiple shared contingency paths. Section III-C outlines a static obstacle penalty term f_{prx} which is used in the $J^{\text{static}}_{(\cdot)}$ cost terms. Section III-D presents a novel algorithm to compute a tight bound on the point-wise collision probability between two polygons with uncertain relative orientation and position. This collision probability bound is used as the basis for a dynamic obstacle collision probability optimization constraint. Section III-E presents a full summary of the path optimization algorithm, including a full description of the objective function terms and all of the constraint functions.

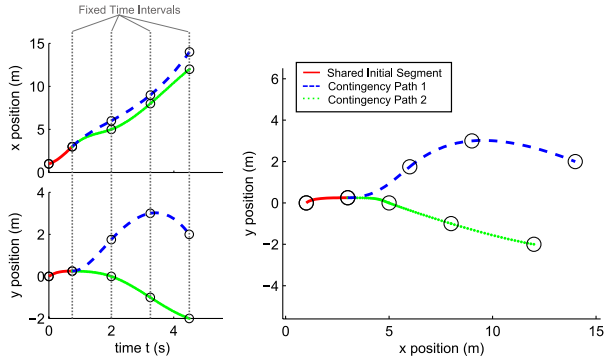

Fig. 3. Depictions of the spline-based path representations as a function of time (left) and in physical space (right).

B. Trajectory Representation

A key challenge when planning multiple contingency paths in real time is defining an efficient trajectory representation. Such a trajectory representation must be both expressive in its ability to cover the search space and compact in its dimensionality. Cubic splines are chosen for this purpose because two splines $y(t)$ and $x(t)$ define a continuous representation of the robot position (x, y) and its derivatives (v^x, v^y, a^x, a^y) as a function of time t using only a small set of control points h as variables. The optimization vector $\mathbf{h}$ is a set of parameters that define n_s contingency trajectories with a shared initial segment:

$$\mathbf{h} = \{h^x_1, h^y_1 | h^{1,x}_{2:n}, h^{1,y}_{2:n}, \ldots, h^{n_s,x}_{2:n}, h^{n_s,y}_{2:n}\} \quad (9)$$

where n is the number of cubic line segments in each contingency path. The shared initial segment constraint is enforced by defining the initial segment as an independent cubic line segment, parameterized by the robot's current state and the first control point (h^x_1, h^y_1). The first control point (h^x_1, h^y_1) represents the end of the initial shared segment, and a separate cubic spline is defined for each contingency path starting from this point, $\{h^{1,x}_{2:n}, h^{1,y}_{2:n}, \ldots, h^{n_s,x}_{2:n}, h^{n_s,y}_{2:n}\}$.

Fig. 3 depicts a cubic spline representation for two contingency paths. In order to reduce dimensionality, the control points on the spline are constrained to lie on fixed time intervals, as indicated by the dashed vertical lines. By using this cubic spline definition, the robot's position and its derivatives at a given time t depend linearly on the control point parameters $\mathbf{h}$.

The optimization vector $\mathbf{h}$ is initialized by setting all contingency paths equal to a nominal path. The nominal path is based on the robot following the lane center in an obstacle-free environment.

C. Static Obstacle Cost Map

To handle static obstacles and static driving boundaries such as road lane boundaries, a driving corridor is defined that explicitly expresses an obstacle-free space that the robot is expected to drive through. Driving corridor descriptions are common in mobile robotics, including Geraerts and Overmars [26] and Wein *et al*. [27]. Driving corridor cost terms have also been applied in the context of autonomous vehicles, most notably by Dolgov *et al*. [28], who develop a local minima free-cost map based on a Voronoi decomposition of the robot's static obstacle map. The driving corridor definition presented here is designed to provide an easily differentiable static obstacle cost term that is convex with respect to the lateral offset of the robot from the center of the driving corridor.

The obstacle-free driving corridor is represented by a sequence of connected line segments, defining the centerline of the corridor along with an associated width for each centerline vertex node. For driving in unstructured environments, an obstacle-free driving corridor is identified using a discrete search over a grid-based representation of the environment, as in [15]. For structured environments, such as driving on a road network, an obstacle-free driving corridor is identified using a combination of *a priori* information, such as road lane definitions, and sensed obstacle information. In the lane driving scenario, the centerline and node width information is obtained trivially from the lane definitions in the *a priori* map. For the unstructured environment scenario, or when the lane centerline must be shifted due to static obstacles such as parked cars, a centerline can be obtained using a straight skeleton decomposition of the corridor, as in [29]. Centerline node widths can then be computed by solving for the minimum distance to the boundary for each node.

A penalty function is defined in the path-planning optimization [as part of $J^{\text{static}}_{(\cdot)}$ in (7) and (8)] to bias the robot away from static obstacles in the environment. This penalty function $f_{\text{prx}}(x, y)$ is defined as a normalized distance between an evaluation point along the robot's path (x, y) and the centerline of the driving corridor:

$$f_{\text{prx}}(x, y) = \min_i \left(\frac{d_i}{w_i^*} \right)^2 \quad \forall i \in [1, n_c] \tag{10}$$

where d_i is the minimum distance between the robot position (x, y) and the ith centerline segment, w_i^* is the interpolated width of the driving corridor for the ith centerline segment at the robot position, and n_c is the total number of centerline segments. Fig. 4 shows this calculation for the ith centerline segment. This quadratic cost function avoids local minima by having a zero cost centerline, is convex with respect to the robot's lateral offset in its driving lane, and allows for straightforward calculation of derivatives even within obstacle regions.

D. Collision Probability

The problem of calculating collision probabilities with uncertain dynamic obstacles has been studied in a range of fields, including air traffic control [30], satellite collision avoidance [31], and mobile robotics [32]. Fig. 5 depicts a typical scenario, where

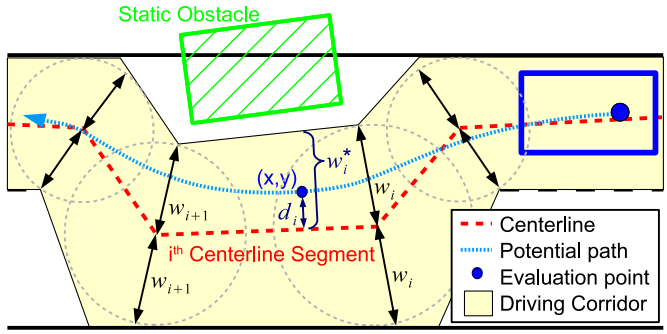

Fig. 4. Diagram of the distance calculation between an evaluation point along the robot's path and a single segment of the centerline.

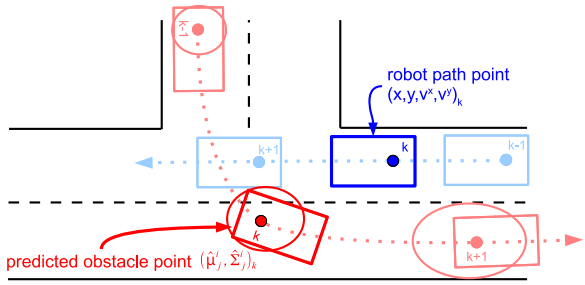

Fig. 5. Diagram of a scenario in which the dynamic collision probability calculation must be performed for multiple discrete timesteps.

point-wise collision probability calculations must be computed for multiple timesteps; timestep k is highlighted. For notational simplicity in the subsequent collision probability derivation, the timestep k case is considered, and the k notation is dropped for now. Let $z_{\text{robot}} = [x_{\text{robot}}, y_{\text{robot}}]$ represent the interpolated position of the robot, and let $\mu^z_{\text{obst}} = [\mu^x_{\text{obst}}, \mu^y_{\text{obst}}]$ and Σ^z_{obst} represent the mean position and 2×2 position error covariance of the obstacle. A normal distribution is assumed for the obstacle position distribution, as this is consistent with the assumptions of the obstacle tracker and the obstacle prediction algorithm presented in Section II. More general obstacle position distributions can be accommodated using a mixture of Gaussians representation.

For the application of path planning in autonomous road vehicles, the collision probability calculation is especially challenging because road vehicles are expected to interact closely with dynamic obstacles. This prevents the use of circular overapproximations of the obstacle shape to account for uncertainties in the obstacle's orientation. Fig. 6 shows a common scenario in which the conservativeness of a circular collision probability bound is a problem: encountering a vehicle traveling in the opposite direction on a two-lane road. In this case, using a circular collision probability bound, would prevent basic, expected driving behavior, such as driving past the obstacle vehicle in an adjacent lane.

A tighter upper bound can be achieved by using the actual rectangular robot and obstacle shapes. This complicates the collision probability calculation because the relative obstacle orientation becomes an additional uncertain variable. For computational simplicity, the obstacle orientation is

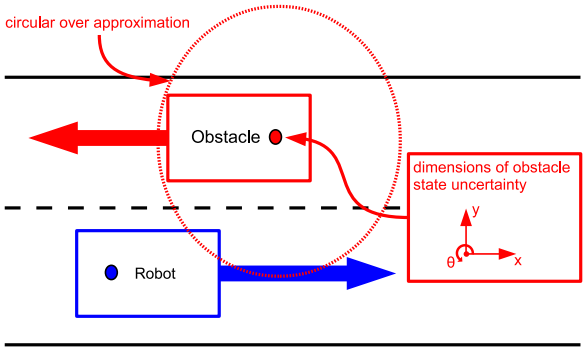

Fig. 6. Depiction of a circular overapproximation for the shape of a road vehicle in an adjacent lane.

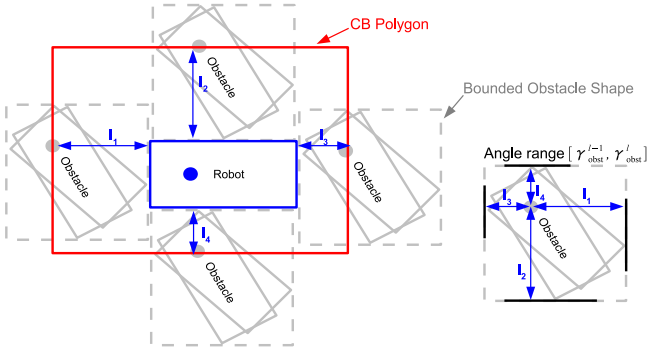

Fig. 7. Formation of a combined body for the collision probability between a rectangular robot and a rectangular obstacle with a set range of possible orientations, $\gamma_{\text{obst}} \in [\gamma_{\text{obst}}^{l-1}, \gamma_{\text{obst}}^{l}]$.

assumed to be an independent 1-D Gaussian distribution, $\gamma_{\text{obst}} \sim \mathcal{N}(\mu_{\text{obst}}^{\gamma}, (\sigma_{\text{obst}}^{\gamma})^2)$; this is equivalent to ignoring correlations between the orientation and position elements of the obstacle's state error covariance. Using this assumption, the collision probability can be found by marginalizing over the obstacle orientation uncertainty:

$$p_{\text{coll}}(z_{\text{robot}}, \mu_{\text{obst}}^{z}, \Sigma_{\text{obst}}^{z}, \mu_{\text{obst}}^{\gamma}, \sigma_{\text{obst}}^{\gamma}) = \int_{\gamma_{\text{obst}}} p(\gamma_{\text{obst}}) p_{\text{coll}}^{\text{poly}}(z_{\text{robot}}, \mu_{\text{obst}}^{z}, \Sigma_{\text{obst}}^{z}, \gamma_{\text{obst}}) \quad (11)$$

where $p_{\text{coll}}^{\text{poly}}(z_{\text{robot}}, \mu_{\text{obst}}^{z}, \Sigma_{\text{obst}}^{z}, \gamma_{\text{obst}})$ is the point-wise collision probability between the robot and the obstacle for a fixed relative orientation γ_{obst}. An upper bound on the integral in (11) can be found by taking a sum over n_γ discrete obstacle orientation ranges, $[\gamma_{\text{obst}}^{l-1}, \gamma_{\text{obst}}^{l}]$ for $l \in [1, n_\gamma]$, where $\sum_{l=1}^{n_\gamma} p([\gamma_{\text{obst}}^{l-1}, \gamma_{\text{obst}}^{l}]) = 1$:

$$p_{\text{coll}}(z_{\text{robot}}, \mu_{\text{obst}}^{z}, \Sigma_{\text{obst}}^{z}, \mu_{\text{obst}}^{\gamma}, \sigma_{\text{obst}}^{\gamma}) \leq \sum_{l=1}^{n_\gamma} p([\gamma_{\text{obst}}^{l-1}, \gamma_{\text{obst}}^{l}]) \cdot p_{\text{coll}}^{\text{poly}}(z_{\text{robot}}, \mu_{\text{obst}}^{z}, \Sigma_{\text{obst}}^{z}, [\gamma_{\text{obst}}^{l-1}, \gamma_{\text{obst}}^{l}]). \quad (12)$$

The sum in (12) is easier to compute than the integral in (11), but it is still inefficient because the obstacle orientation distribution $\gamma_{\text{obst}} \sim \mathcal{N}(\mu_{\text{obst}}^{\gamma}, (\sigma_{\text{obst}}^{\gamma})^2)$ has infinite support. For most lane driving scenarios, the obstacle's orientation uncertainty is small relative to its total range of possible orientations, and the bulk of the probability mass for γ_{obst} is grouped closely around the mean $\mu_{\text{obst}}^{\gamma}$ such that $\sigma_{\text{obst}}^{\gamma} \ll 2\pi$. Thus, the upper bound in (12) can be reformulated as a sum over discrete angle ranges that encompass a predefined confidence interval δ_γ of the γ_{obst} distribution, centered around the mean. In this case, $\sum_{l=1}^{n_\gamma} p([\gamma_{\text{obst}}^{l-1}, \gamma_{\text{obst}}^{l}]) = \delta_\gamma$. The remaining probability mass at the tails of the distribution is bounded using a circular overapproximation

$$p_{\text{coll}}(z_{\text{robot}}, \mu_{\text{obst}}^{z}, \Sigma_{\text{obst}}^{z}, \mu_{\text{obst}}^{\gamma}, \sigma_{\text{obst}}^{\gamma}) \leq \sum_{l=1}^{n_\gamma} p([\gamma_{\text{obst}}^{l-1}, \gamma_{\text{obst}}^{l}]) p_{\text{coll}}^{\text{poly}}(z_{\text{robot}}, \mu_{\text{obst}}^{z}, \Sigma_{\text{obst}}^{z}, [\gamma_{\text{obst}}^{l-1}, \gamma_{\text{obst}}^{l}]) + (1 - \delta_\gamma) p_{\text{coll}}^{\text{circ}}(z_{\text{robot}}, \mu_{\text{obst}}^{z}, \Sigma_{\text{obst}}^{z}). \quad (13)$$

The circular bound $p_{\text{coll}}^{\text{circ}}(z_{\text{robot}}, \mu_{\text{obst}}^{z}, \Sigma_{\text{obst}}^{z})$ is based on the bound presented by Hwang and Seah [5] and uses the furthest point on the body from the center of the back axle as the radius for both the robot and obstacle vehicles.

The right-hand side of (13) requires the computation of the probability of collision between the robot and the obstacle for a specified range of possible obstacle orientations, $p_{\text{coll}}^{\text{poly}}(z_{\text{robot}}, \mu_{\text{obst}}^{z}, \Sigma_{\text{obst}}^{z}, [\gamma_{\text{obst}}^{l-1}, \gamma_{\text{obst}}^{l}])$. The true instantaneous probability of collision between the robot and a predicted obstacle over a fixed range of possible orientations is calculated by integrating the obstacle position distribution $\mathcal{N}(\mu_{\text{obst}}^{z}, \Sigma_{\text{obst}}^{z})$ over the combined body (CB) of the robot shape and all possible obstacle shapes

$$p_{\text{coll}}^{\text{poly}}(z_{\text{robot}}, \mu_{\text{obst}}^{z}, \Sigma_{\text{obst}}^{z}, [\gamma_{\text{obst}}^{l-1}, \gamma_{\text{obst}}^{l}]) = \int_{\text{CB}} \mathcal{N}(\mu_{\text{obst}}^{z}, \Sigma_{\text{obst}}^{z}). \quad (14)$$

To compute an upper bound to this term, the obstacle is first rotated through its range of possible orientations, and a bounding box is used to bound the obstacle shape over the entire rotation. The problem can then be converted to a single-point collision calculation by sweeping the bounded obstacle shape around the perimeter of the robot rectangle to create a combined body polygon, as shown in Fig. 7. This is equivalent to finding the convex hull of a Minkowski sum of the robot shape and the bounded obstacle shape.

The collision probability can then be computed efficiently by exploiting the separability of Gaussian distributions to decompose the obstacle position distribution into the product of 1-D Gaussian distributions. The obstacle position covariance ellipse must be aligned with the coordinate axes in order to be decoupled. However, the CB polygon must also be aligned with the coordinate axes in order to define a tight rectangular integration region. Paielli *et al.* [30] provide a method to normalize the error covariance using a linear coordinate transformation:

$$W = (\sqrt{\Sigma_{\text{obst}}^{z}})^{-1} \quad (15)$$

where $W \Sigma_{\text{obst}}^{z} W^T = I_{2\times 2}$. This transformation allows the obstacle position distribution to be decoupled in any direction. An oriented bounding box is then applied to the transformed

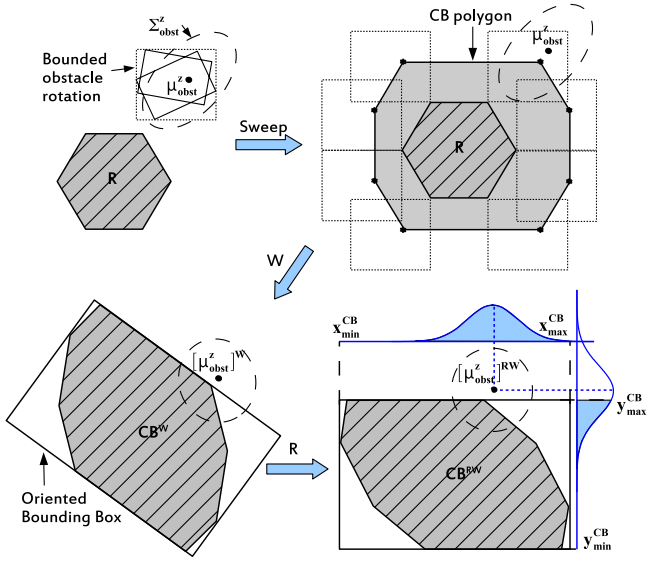

Fig. 8. Depiction of polygonal collision probability bound calculation between an arbitrary polygon region and a rectangular obstacle for a known range of obstacle orientations. (Top left) Initial scene. (Top right) Combined body of the obstacle shape and the vehicle shape. (Bottom left) Effects of the transformation W and the application of an oriented bounding box. (Bottom right) Integration over the aligned combined body CB^{RW} using the decoupled, normalized obstacle position distribution.

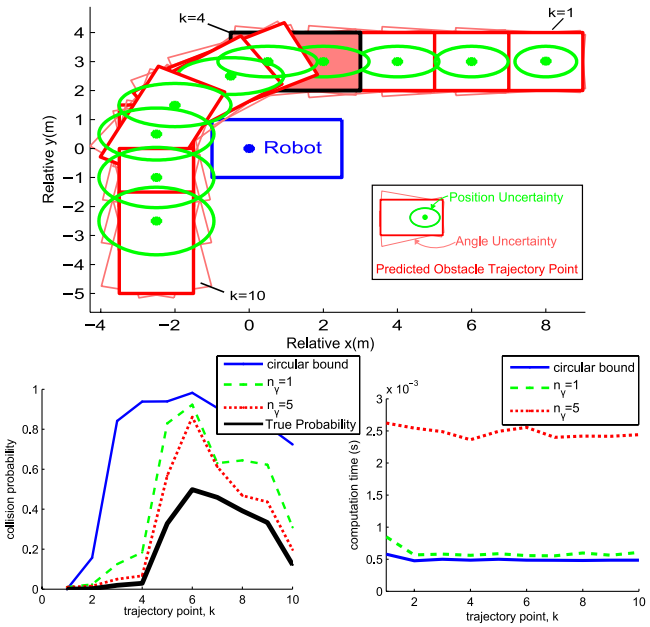

Fig. 9. Comparison of different collision probability approximations for a sample close proximity obstacle interaction scenario. (Top) Relative trajectory of the robot and obstacle vehicles. (Bottom left) Collision probability comparison for each trajectory point. (Bottom right) Computation time comparison.

combined body polygon CB^{W} to define a tight rectangular integration region, and the coordinate frame is then rotated so that the transformed combined body is aligned with the coordinate axes. This transformed, bounded and aligned CB region is defined as CB^{RW}. Decoupling the obstacle covariance along the coordinate axis allows the $p_{\mathrm{coll}}^{\mathrm{poly}}(z_{\mathrm{robot}}, \mu_{\mathrm{obst}}^{z}, \Sigma_{\mathrm{obst}}^{z}, [\gamma_{\mathrm{obst}}^{l-1}, \gamma_{\mathrm{obst}}^{l}])$ term to be bounded as

$$p_{\mathrm{coll}}^{\mathrm{poly}}(z_{\mathrm{robot}}, \mu_{\mathrm{obst}}^{z}, \Sigma_{\mathrm{obst}}^{z}, [\gamma_{\mathrm{obst}}^{l-1}, \gamma_{\mathrm{obst}}^{l}]) \leq \int_{\mathrm{CB}^{W}} \mathcal{N}([\mu_{\mathrm{obst}}^{x}]^{RW}, 1) \cdot \mathcal{N}([\mu_{\mathrm{obst}}^{y}]^{RW}, 1) \quad (16)$$

where $[\mu_{\mathrm{obst}}^{z}]^{RW} = ([\mu_{\mathrm{obst}}^{x}]^{RW}, [\mu_{\mathrm{obst}}^{y}]^{RW})$ are the coordinates of the obstacle origin after the transformation and alignment.

Once aligned with the coordinate axes, the CB^{RW} bounding box is defined by the parameters $(x_{\min}^{\mathrm{CB}}, y_{\min}^{\mathrm{CB}}, x_{\max}^{\mathrm{CB}}, y_{\max}^{\mathrm{CB}})$. An upper bound on the collision probability for the given orientation range $\gamma_{\mathrm{obst}} \in [\gamma_{\mathrm{obst}}^{l-1}, \gamma_{\mathrm{obst}}^{l}]$ can be computed as

$$\begin{aligned} &p_{\mathrm{coll}}^{\mathrm{rect}}(z_{\mathrm{robot}}, \mu_{\mathrm{obst}}^{z}, \Sigma_{\mathrm{obst}}^{z}, [\gamma_{\mathrm{obst}}^{l-1}, \gamma_{\mathrm{obst}}^{l}]) \\ &\leq (\Psi(x_{\max}^{\mathrm{CB}} - [\mu_{\mathrm{obst}}^{x}]^{RW}; 0, 1) - \Psi(x_{\min}^{\mathrm{CB}} - [\mu_{\mathrm{obst}}^{x}]^{RW}; 0, 1)) \\ &\cdot (\Psi(y_{\max}^{\mathrm{CB}} - [\mu_{\mathrm{obst}}^{y}]^{RW}; 0, 1) - \Psi(y_{\min}^{\mathrm{CB}} - [\mu_{\mathrm{obst}}^{y}]^{RW}; 0, 1)) \end{aligned} \quad (17)$$

where $\Psi(z; \mu, \sigma) = \frac{1}{2}[1 + \mathrm{erf}(\frac{z-\mu}{\sqrt{2\sigma^2}})]$ is the 1-D Gaussian cumulative distribution function.

The steps of the polygonal collision probability bound calculation for an arbitrary polygonal robot shape and a rectangular obstacle are shown in Fig. 8. The top left image shows the initial scene, with a bounded obstacle shape over the given orientation range $[\gamma_{\mathrm{obst}}^{l-1}, \gamma_{\mathrm{obst}}^{l}]$. The top right image shows the combined body of the robot shape and the bounded obstacle shape, constructed as shown in Fig. 7. The bottom left image shows the effects of the linear transformation W, and the bottom right image shows the final configuration after a rotation is applied to align the transformed CB with the coordinate axes of the reference frame.

1) Collision Probability Analysis: The polygonal collision probability bound, $p_{\mathrm{coll}}^{\mathrm{poly}}(z_{\mathrm{robot}}, \mu_{\mathrm{obst}}^{z}, \Sigma_{\mathrm{obst}}^{z}, \mu_{\mathrm{obst}}^{\gamma}, \sigma_{\mathrm{obst}}^{\gamma})$ in (13), provides a tighter bound than a circular collision probability bound, but this improvement in accuracy comes at a higher computational cost. In order to understand this trade-off, a close proximity obstacle interaction scenario is evaluated. Fig. 9 shows an interaction, in a robot-centric coordinate frame, between the robot vehicle and an obstacle trajectory prediction. A comparison is made for each point on the trajectory, $k \in [1, 10]$, for three cases: a circular bound, and two versions of the polygonal bound [(13) using $n_\gamma = 1$ and $n_\gamma = 5$]. The true probability of collision is also computed using Monte Carlo simulations. To improve computational performance, (13) is simplified by setting $p_{\mathrm{coll}}^{\mathrm{circ}}(z_{\mathrm{robot}}, \mu_{\mathrm{obst}}^{z}, \Sigma_{\mathrm{obst}}^{z}) = 1$ and $\delta_\gamma = 0.99$. This simplification eliminates the need to compute $p_{\mathrm{coll}}^{\mathrm{circ}}(z_{\mathrm{robot}}, \mu_{\mathrm{obst}}^{z}, \Sigma_{\mathrm{obst}}^{z})$ and is accurate for δ_γ values close to 1. This simplification is also used in all subsequent simulations.

Fig. 9 shows that the polygonal collision probability bound produces a much tighter upper bound than the circular collision probability bound, especially at $k = 4$, where the obstacle vehicle is driving parallel to the robot in an adjacent lane. This tighter upper bound still prevents unsafe situations, yet also allows expected close proximity driving behaviors. The results in Fig. 9 also show that $n_\gamma = 1$ provides nearly as tight of a bound

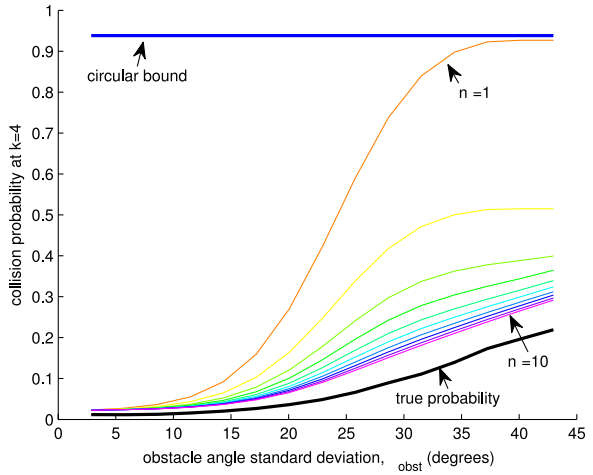

Fig. 10. Collision probability bound for $k = 4$ as a function of standard deviation of the obstacle angle uncertainty $\sigma^{\gamma}_{\text{obst}}$ and number of orientation ranges n_{γ}.

as $n_{\gamma} = 5$ but with a computational cost nearly equivalent to that of the circular bound. The polygonal bound performs the worst at $k = 6$ when the obstacle is aligned at a nearly 45° angle in the robot's reference frame. This misalignment causes the bounding box approximations to poorly fit the obstacle shape. However, cases such as $k = 6$ only occur when the obstacle blocks the robot's lane of travel, and in these situations, substantial overapproximations are acceptable.

To evaluate how the polygonal collision probability bound varies as a function of the obstacle angle uncertainty, the $k = 4$ case was reevaluated for $n_{\gamma} \in [1, 10]$ over a range of obstacle angle standard deviations, $\sigma^{\gamma}_{\text{obst}}$, as shown in Fig. 10. The results show that when $n_{\gamma} > 3$, the polygonal collision probability bound is tight compared with the true collision probability over the entire tested range of obstacle angle standard deviations, $\sigma^{\gamma}_{\text{obst}} \in [3^{\circ}, 43^{\circ}]$. The $n_{\gamma} = 1$ bound is most useful for cases with low obstacle angle uncertainty, $\sigma^{\gamma}_{\text{obst}} < 20^{\circ}$; however, the $n_{\gamma} = 1$ bound still offers a significant improvement over the circular bound for $\sigma^{\gamma}_{\text{obst}} < 35^{\circ}$. The polygonal bound does not, in general, converge to the true probability as $n_{\gamma} \to \infty$; overapproximations still occur due to successive bounding box approximations. A constant offset is also introduced by using $p^{\text{circ}}_{\text{coll}}(z_{\text{robot}}, \mu^{z}_{\text{obst}}, \Sigma^{z}_{\text{obst}}) = 1$ in (13).

E. Optimization Summary

The contingency planner uses a nonlinear, constrained numerical optimization procedure, as defined in (5), to find a locally optimal set of contingency paths, parameterized by $\mathbf{h}$. The appealing aspect of this constrained optimization approach is the formulation: The optimization cost function can be defined based on desirable path goals and behaviors, and optimization constraints can be defined to enforce safety requirements and to ensure conformance with vehicle limitations.

These costs are defined as a sum over discrete evaluation points, $k \in [1, N]$. For this paper, the time interval between evaluation points Δt_{eval} is set equal to the obstacle prediction timestep for convenience. This discrete formulation approximates a continuous integral over the path segments but allows for simpler point-wise calculations.

The cost for the shared segment $J_{\text{shared}}(\mathbf{h}, \mathbf{T})$ is defined as

$$\begin{aligned} &J_{\text{shared}}(\mathbf{h}, \mathbf{T}) \\ &= \underbrace{\alpha_a \sum_{k=1}^{n_1}((a_k^x)^2 + (a_k^y)^2) + \alpha_c \sum_{k=1}^{n_1}(v_k^x a_k^y - v_k^y a_k^x)^2}_{J^{\text{dynamics}}_{\text{shared}}} \\ &\quad + \underbrace{\alpha_d \sum_{k=1}^{n_1} f_{\text{prx}}(x_k, y_k)}_{J^{\text{static}}_{\text{shared}}} \\ &\quad + \underbrace{\alpha_p \sum_{\mathcal{T}^j \in \mathbf{T}} p(\mathcal{T}^j) \sum_{T \in \mathcal{T}^j} \sum_{k=1}^{n_1} f^{\text{circ}}_{\text{coll}}(x_k, y_k, T(k))}_{J^{\text{collision}}_{\text{shared}}} \end{aligned} \tag{18}$$

and the cost for each contingency path $J_{\text{conting}}(\mathbf{h}, \mathcal{T}^j)$ is defined as

$$\begin{aligned} &J_{\text{conting}}(\mathbf{h}, \mathcal{T}^j) \\ &= \underbrace{\alpha_a \sum_{k=n_1+1}^{N}((a_k^x)^2 + (a_k^y)^2) + \alpha_c \sum_{k=n_1+1}^{N}(v_k^x a_k^y - v_k^y a_k^x)^2}_{J^{\text{dynamics}}_{\text{conting}}} \\ &\quad + \underbrace{\alpha_d \sum_{k=n_1+1}^{N} f_{\text{prx}}(x_k, y_k)}_{J^{\text{static}}_{\text{conting}}} + \underbrace{\alpha_p \sum_{T \in \mathcal{T}^j} \sum_{k=n_1+1}^{N} f^{\text{circ}}_{\text{coll}}(x_k, y_k, T(k))}_{J^{\text{collision}}_{\text{conting}}} \\ &\quad + \underbrace{\alpha_g f_{\text{goal}}(x_N, y_N)}_{J^{\text{goal}}_{\text{conting}}} \end{aligned} \tag{19}$$

where n_1 is the number of discrete timesteps evaluated over the initial path segment, and $N - n_1$ represents the number of discrete timesteps evaluated along each contingency path. The f_{prx} term penalizes proximity to static obstacles as defined in (10), f_{goal} is the squared Euclidean distance between a given path point and the goal point, and the $f^{\text{circ}}_{\text{coll}}$ term is a circular collision probability bound that is based on the bound in [5]. This circular bound is equivalent to the collision probability bound that is presented in Section III-D, where both the robot and the obstacle shapes are approximated by circles. Here, $f^{\text{circ}}_{\text{coll}}$ is used as an underapproximation of the true collision probability, where the widths of the robot and obstacle are used as their radii. This underapproximation results in low penalties on expected close proximity obstacle interactions while allowing for an easier to optimize, orientation-independent cost term. This underapproximation is permissible because true collision probability bounds are used in the optimization constraints to ensure path safety.

The $\alpha_{(\cdot)}$ parameters in (18) and (19) are the weights that govern the relative importance of each penalty term: The α_a-weighted term penalizes high acceleration, the α_c-weighted term penalizes unnormalized path curvature, the α_d-weighted

term penalizes proximity to static obstacles, the α_p-weighted term penalizes high collision probabilities with dynamic obstacles, and the α_g-weighted term penalizes the distance of the robot from its goal at the end of the planning horizon.

The acceleration, curvature, and distance from goal terms are formulated to be convex with respect to position (x, y) and its derivatives (v^x, v^y, a^x, a^y). Unnormalized path curvature $(v_k^x a_k^y - v_k^y a_k^x)$ is used to maintain convexity and is equivalent to weighting the path curvature by the robot velocity magnitude cubed. The collision probability term $f_{\text{coll}}^{\text{circ}}$ represents a unimodal cost hill in the robot's (x, y) space and behaves as a convex function away from the cost peak. The static obstacle proximity term $f_{\text{prx}}(x_k, y_k)$ is convex with respect to the robot's lateral offset from the center of the driving corridor.

A set of constraint functions $C(\mathbf{h})$ are evaluated at each timestep k along the shared initial segment and along each contingency path, in order to ensure that the optimized solution obeys the physical constraints of the robot and stays below a required maximum probability of collision with dynamic obstacles. These constraint functions are

$$-(v_{k-1}^x v_k^x - v_{k-1}^y v_k^y) - \beta_{\max}\|\mathbf{v}_{k-1}\| \cdot \|\mathbf{v}_k\| \leq 0 \tag{20}$$

$$(v_k^x)^2 + (v_k^y)^2 - v_{\max}^2 \leq 0 \tag{21}$$

$$\left(\frac{(a_k^x)^2 + (a_k^y)^2}{[a_f]_{\max}^2}\right) + \left(\frac{(v_k^x a_k^y - v_k^y a_k^x)^2}{\epsilon^2 [a_l]_{\max}^2}\right) - 1 \leq 0 \tag{22}$$

$$f_{\text{prx}}(x_k, y_k) - 1 \leq 0 \tag{23}$$

$$-p_{\max} + \sum_{T \in \mathcal{T}^j} p(T) f_{\text{coll}}^{\text{poly}}(x_k, y_k, \theta_{\text{robot},k}, T(k)) \leq 0. \tag{24}$$

Equation (20) constrains changes in robot orientation between successive timesteps, where the parameter $\beta_{\max}$ controls the maximum allowable change in orientation (e.g., $\beta_{\max} = 0$ equates to $[\Delta\theta_{\text{robot}}]_{\max} = 90°$). Equation (21) enforces a maximum velocity constraint. Equation (22) enforces a maximum acceleration constraint using an acceleration ellipse, where $[a_f]_{\max}$ is the maximum allowable forward acceleration, $[a_l]_{\max}$ is the maximum allowable lateral acceleration, and $(v_k^x a_k^y - v_k^y a_k^x)$ is an approximation of the robot's lateral acceleration, which is equivalent to the robot's lateral acceleration multiplied by the velocity magnitude. Since $(v_k^x a_k^y - v_k^y a_k^x)$ is an approximation, the constant $\epsilon = 1$ m/s is introduced to ensure consistent units. Equation (23) constrains the robot's configuration space to prevent collisions with static obstacle regions. Equation (24) constrains the maximum probability of collision between the robot following the jth contingency plan and the dynamic obstacle trajectories in $\mathcal{T}^j$ to be less than $p_{\max}$ using the collision probability bound defined in (13), where $z_{\text{robot}} = [x_k, y_k]$ and $\theta_{\text{robot},k}$ are used to transform the calculation into a robot-centric coordinate frame, and $\mu_{\text{obst}}^z, \Sigma_{\text{obst}}^z, \mu_{\text{obst}}^\gamma, \sigma_{\text{obst}}^\gamma$ are defined by $T(k)$.

The $f_{\text{coll}}^{\text{poly}}$ constraint function in (24) includes orientation information, $\theta_{\text{robot},k} = \tan^{-1}(\frac{v_k^y}{v_k^x})$, making this term nonconvex. However, this constraint is necessary because it provides a formal bound on collision probability and ensures the safety of the final set of contingency paths. This term ensures that all feasible optimization solutions are safe up to a probability of $p_{\max}$. A circular collision probability bound could be used here to improve convexity; however, as described earlier, the circular probability bound is overly conservative and prevents normal driving behaviors such as passing in adjacent road lanes.

IV. Trajectory Clustering for Improved Computational Scaling

Equations (6) and (9) show, respectively, that both the cost function complexity and the number of optimization variables required by the proposed contingency planning algorithm scale linearly with the number of obstacle prediction permutations, n_s. In turn, (3) shows that n_s scales exponentially with the number of obstacles n_o and the number of trajectory predictions for each obstacle $|\mathcal{T}_i|$. This exponential scaling effectively prevents any reasonably complex contingency planning algorithm, with existing levels of computational power, from performing real-time navigation in environments with more than a small number of obstacles and possible goal states.

The number of required contingency plans can be reduced by clustering obstacle prediction permutations prior to executing the contingency planner based on similarities in their potential influence on the robot's future path. This allows the planner to use a fixed maximum number of contingency plans n_c regardless of the number of obstacles and possible obstacle goals in the environment. Clustering similar predicted obstacle trajectory permutations provides improved computational scaling because it allows the planner to ignore combinatorial permutations of obstacle predictions within trajectory clusters, under the assumption that these permutations all have a similar influence on the robot's future path and can be safely approximated as simultaneously occurring events. The planner is thus able to treat each cluster of trajectory permutations $\mathcal{C}^l$ as the union of all possible trajectories that are contained in its constituent permutations, $\mathcal{C}^l = \bigcup_{\mathcal{T}^j \in \mathbf{T}^l} \mathcal{T}^j$, where $\mathbf{T}^l \subset \mathbf{T}$ represents the set of obstacle trajectory permutations that are included in the lth cluster.

Trajectory clustering reduces the complexity of the planning problem from planning an exponential n_s number of contingency plans that each need to avoid n_o obstacle trajectories, to planning a fixed n_c number of contingency plans, that each must avoid a linearly scaling $|C^l| = \sum_{i=1}^{n_o} |\mathcal{T}_i^l|$ number of obstacle trajectories, where $\mathcal{T}_i^l \subset \mathcal{T}_i$ is the subset of possible trajectory predictions for obstacle i included in the lth cluster. In the limit of clustering all predicted obstacle trajectories into a single cluster, $n_c = 1$, the motion planner would recover the single path dynamic planning approach depicted in Fig. 2(a), where $|\mathcal{C}^1| = \sum_{i=1}^{n_o} |\mathcal{T}_i|$.

A. Trajectory Clustering

Clustering spatial trajectories is a popular a machine-learning technique to model the motion of people or other obstacles through a given environment. Hierarchical methods are commonly used with trajectory clustering since simple generative models are difficult to formulate. Fu *et al*. [33] present a clustering method using both spectral and hierarchical clustering to

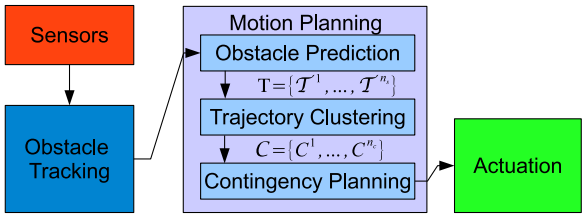

Fig. 11. Block diagram of the contingency planning architecture with trajectory clustering.

identify common trajectories in traffic videos. Lee *et al*. [34] identify similar subtrajectories in a group of trajectories by first partitioning each trajectory into a sequence of line segments and then clustering the line segments using a hierarchical joining method. Nonhierarchical approaches have also been developed, including Gaffney and Smyth [35] who assume a generative model for a set of trajectories and use the EM algorithm to learn the mixture assignments. All of these approaches effectively group similar trajectories, but they have a fundamentally different goal than a clustering algorithm that is designed to simplify path planning.

The clustering algorithm that is outlined in this section is designed to maximize dissimilarity between clustered sets of obstacle trajectory predictions in order to capture the most important mutually exclusive obstacle decisions and uncertainties with a smaller set of contingency plans, allowing the robot to maintain the performance advantages of exhaustive contingency planning while requiring less computation time. Fig. 11 shows the contingency planning architecture with trajectory clustering included as a preprocessing step between obstacle prediction and contingency planning.

Using a fixed number of clusters n_c where $n_c \leq n_s$, (6) can be rewritten as

$$J(\mathbf{h}) = J_{\text{shared}}(\mathbf{h}, \mathbf{C}) + \sum_{\mathcal{C}^l \in \mathbf{C}} p(\mathcal{C}^l) J_{\text{conting}}(\mathbf{h}, \mathcal{C}^l) \tag{25}$$

where $\mathbf{C} = \{\mathcal{C}^1, \ldots, \mathcal{C}^{n_c}\}$ is a possible clustering of the obstacle trajectory predictions.

Similarly, the dynamic obstacle cost term in J_{shared} can be redefined as

$$J_{\text{shared}}^{\text{collision}} = \alpha_p \sum_{\mathcal{C}^l \in \mathbf{C}} p(\mathcal{C}^l) \sum_{T \in \mathcal{C}^l} \sum_{k=1}^{n_1} f_{\text{coll}}^{\text{circ}}(x_k, y_k, T(k)) \tag{26}$$

and the dynamic obstacle cost term in J_{conting} can be redefined as

$$J_{\text{conting}}^{\text{collision}} = \alpha_p \sum_{T \in \mathcal{C}^l} \sum_{k=n_1+1}^{N} f_{\text{coll}}^{\text{circ}}(x_k, y_k, T(k)). \tag{27}$$

The dynamic collision probability optimization constraint in (24) can also be redefined as

$$-p_{\max} + \sum_{T \in \mathcal{C}^l} p(T) f_{\text{coll}}^{\text{poly}}(x_k, y_k, \theta_{\text{robot},k}, T(k)) \leq 0. \tag{28}$$

The goal of the trajectory clustering algorithm is to maximize dissimilarity between the obstacle predictions in different clusters. A probability weighted, minimum relevant dissimilarity metric is formulated to capture the dissimilarity between predicted obstacle trajectories, along with the relative occurrence probabilities of the obstacle predictions in each cluster, and the potential impact or relevance each obstacle prediction has on the robot's potential path. For clusters $\mathcal{C}^a, \mathcal{C}^b$, where $a, b \in [1, n_c]$, this metric is defined as

$$\text{diss}(\mathcal{C}^a, \mathcal{C}^b) = p(\mathcal{C}^a, \mathcal{C}^b) \cdot \min_{T^a \in \mathcal{C}^a} \left(w^a \cdot \min_{T^b \in \mathcal{C}^b} D(T^a, T^b) \right) \tag{29}$$

where $p(\mathcal{C}^a, \mathcal{C}^b)$ is the joint probability of the obstacle predictions in both clusters, w^a is a relevancy weighting that is based on how likely predicted obstacle trajectory T^a is to affect the robot's nominal path

$$w^a = \frac{p(T^a)}{\min_k (\mu^a(k) - z_{\text{robot}}(k))^T \Sigma^a(k)^{-1} (\mu^a(k) - z_{\text{robot}}(k))} \tag{30}$$

and $D(T^a, T^b)$ is a dissimilarity metric that reflects the dissimilarity between two predicted obstacle trajectories.

Using this dissimilarity definition, the optimal set of obstacle prediction trajectory clusters is defined as

$$\mathbf{C}_{\text{opt}} = \arg\max_{\mathbf{C}} \left(\min_{\mathcal{C}^a, \mathcal{C}^b \in \mathbf{C}} \text{diss}(\mathcal{C}^a, \mathcal{C}^b) \right). \tag{31}$$

1) Dissimilarity Metric: The dissimilarity metric $D(T^a, T^b)$ in (29) represents the likelihood of two predicted obstacle trajectories having a similar influence on the robot's free configuration space. The potential influence of a predicted obstacle trajectory on the robot's free configuration space is found in an efficient manner by representing the robot's driving corridor $\mathbf{D}_{\text{corr}}$ as the interior points of a set of convex polygons. The set of driving corridor polygons is derived by decomposing the driving corridor definition that is presented in Section III-C into a set of trapezoids $\mathcal{R}_1$ through $\mathcal{R}_{n_R}$. The polygonal collision probability bound that is presented in (13) is then applied to each polygon $\mathcal{R}_{(\cdot)}$ in $\mathbf{D}_{\text{corr}}$ to provide a fast probability bound on whether each discrete timestep along a predicted obstacle trajectory $T(k)$ lies within the driving corridor. This probability bound is then thresholded to determine whether $T(k) \in \mathbf{D}_{\text{corr}}$.

Using this driving corridor inclusion definition, the dissimilarity metric $D(T^a, T^b)$ can be defined as

$$D(T^a, T^b) = \frac{1}{N} \sum_{k=1}^{N} d(T^a(k), T^b(k)) \tag{32}$$

where N is the total number of points in each trajectory, and $d(T^a(k), T^b(k))$ is the dissimilarity between two 2-D obstacle position distributions, $T^{(\cdot)}(k) \sim \mathcal{N}(\mu^{(\cdot)}, \Sigma^{(\cdot)})$, which is defined as

$$d(T^a(k), T^b(k)) = \begin{cases} 1 - \dfrac{1}{e^{M(T^a(k), T^b(k))}}, & T^a(k) \wedge T^b(k) \in \mathbf{D}_{\text{corr}} \\ 1, & T^a(k) \oplus T^b(k) \in \mathbf{D}_{\text{corr}} \\ 0, & T^a(k) \wedge T^b(k) \notin \mathbf{D}_{\text{corr}} \end{cases} \tag{33}$$

where $M(T^a(k), T^b(k))$ is the squared Mahalanobis distance between $T^a(k)$ and $T^b(k)$:

$$M(T^a(k), T^b(k)) = (\mu^a - \mu^b)^T (\Sigma^a + \Sigma^b)^{-1} (\mu^a - \mu^b). \tag{34}$$

Trajectory points which coincide within the driving corridor have zero dissimilarity. As two trajectory points diverge within the driving corridor, their dissimilarity grows until it reaches a maximum at $d(T^a(k), T^b(k)) = 1$ for two points that have no chance of overlapping. A trajectory point within the driving corridor has a dissimilarity measure of $d(T^a(k), T^b(k)) = 1$ when compared with any point outside of the driving corridor, and two trajectory points that lie entirely outside of the robot's driving corridor have zero dissimilarity, meaning that the clustering algorithm has no incentive to separate them.

2) Hierarchical Splitting Algorithm: The clustering optimization problem that is given in (31) can be solved efficiently by using a hierarchical splitting approach that exploits the underlying structure of the problem. Since the clustering algorithm is performed over the elements of $\mathbf{T}$, which is a Cartesian product of the predicted obstacle trajectories for each obstacle, it is only possible to isolate a given obstacle trajectory from the other possible trajectories for the same obstacle. This is equivalent to splitting along a single dimension of the Cartesian product. This implies that, for any cluster of trajectory permutations, $\mathcal{C}^l$, a selected subset of obstacle trajectories for obstacle i, $\tau^* \subset \mathcal{T}_i^l$, can be isolated by splitting the total cluster into new clusters, where one cluster contains τ^*, and the other contains $\mathcal{T}_i^l \setminus \tau^*$. Both clusters also inherit all of the trajectories from the other obstacles.

Clustering is thus performed by initializing a single cluster and performing successive greedy splits in order to iteratively isolate the subset of trajectories from a single obstacle τ^* with the largest probability weighted, relevant dissimilarity when compared with the rest of the trajectories in the cluster. This approach makes the best greedy split possible, based on the probability weighted, relevant dissimilarity metric defined in (29), while still preserving all possible obstacle trajectory permutations.

V. Dangerous Intersection Simulations

Monte Carlo simulations were conducted using a dangerous intersection scenario in order to study the safety and performance of the proposed contingency planning algorithm on a robot engaging in complex obstacle avoidance interactions. This dangerous intersection scenario consists of a robot and one or more obstacle vehicles navigating a four-way intersection with no stop signs or other traffic rules. These simulations represent a worst case intersection scenario and are designed to ensure efficient use of simulation runs by consistently forcing the robot into interesting obstacle avoidance interactions. Due to the condensed number of risky encounters, more aggressive braking and higher collision frequencies are expected compared with normal driving. Collision frequency results are presented here as a relative metric of safety performance between path-planning algorithms. Sections V-A and B present simulation results for a single obstacle vehicle, while Section V-C presents two sets of simulation results for multiple obstacle vehicles; a congested multiple obstacle case is presented in Section V-C1 and a less-congested multiple obstacle case is presented in Section V-C2.

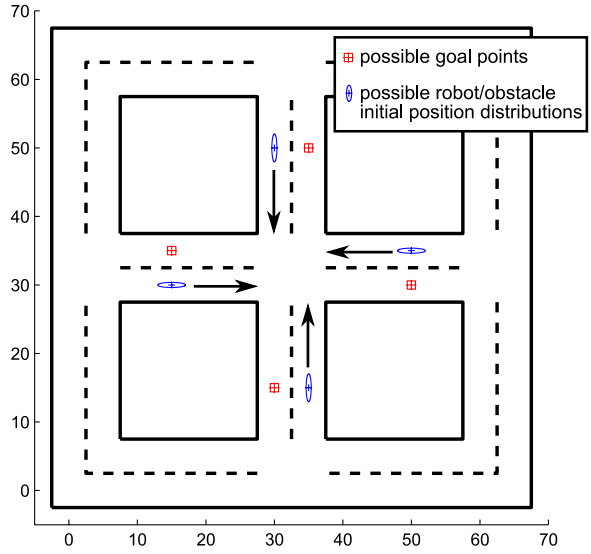

Fig. 12. Map of the simulated scenario with possible initial robot/obstacle distributions depicted by their covariance ellipses and possible goal points depicted as squares.

In all simulations, obstacle vehicles are assigned a nonintelligent driving behavior. This means that they completely ignore the robot while navigating the intersection, even if a collision is imminent. This obstacle behavior eliminates the effects of complex reciprocal robot/obstacle anticipation interactions and allows for consistent analysis of the path planner performance. Obstacle vehicles use a simplified, constraint-free version of the numerical optimization-based path planner that is presented in Section III [see (18) and (19)] but with the $J_{(\cdot)}^{\text{collision}}$ collision probability terms omitted. This same simplified path planner is used as the obstacle prediction model, and each potential obstacle goal (Turn Left, Turn Right, Go Straight) is initialized with equal likelihood.

A. Single Obstacle Simulation Results

The initial conditions for the single obstacle simulations are depicted in Fig. 12. The states of both the robot and the obstacle vehicle are randomly initialized along one of the four road segments leading into the intersection, and both are randomly assigned one of four goal points along the road segments leaving the intersection. Goal point selection is constrained to avoid U-turn maneuvers. The possible initial position distributions and goal points are depicted in Fig. 12. The obstacle vehicle is assigned a mean initial velocity of 5 m/s, and the robot is assigned a mean initial velocity of 3 m/s. This difference in the initial velocity distributions biases the simulation toward more interesting cases, where the obstacle arrives at the intersection before the robot, forcing the robot to actively react to and avoid the obstacle. Cases, where the robot arrives at the intersection

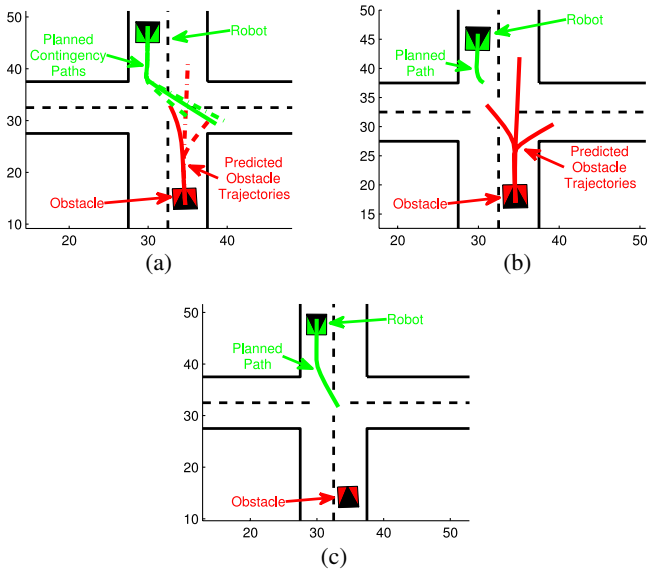

Fig. 13. Sample planning result for each of the three tested algorithms during a single planning cycle. (a) Multipath. (b) Singlepath. (c) Static.

sufficiently before the obstacle vehicle, simply result in the robot traversing the intersection unimpeded.

As a baseline comparison, three algorithms were run: 1) Multipath: the contingency approach proposed in this paper which considers all possible obstacle predictions individually (this is equivalent to using trajectory clustering with $n_c = 3$); 2) Singlepath: a single path variation of the Multipath planner, where a single contingency path is used that avoids all obstacle predictions (equivalent to $n_c = 1$); and 3) Static: a variation of the Singlepath planner which uses a stationary obstacle prediction motion model (obstacles are predicted to not move). All other optimization parameters are identical. Both the obstacle vehicle and the robot use a pure pursuit-based actuation controller [36] to follow their planned paths. Simulations were run in MATLAB on a 3GHz Core i3 processor running under Windows 7, and the path optimization solution was written in C++ using the open-source nonlinear optimization library IPOPT [37].

Fig. 13 shows a sample planning result for each of these algorithms. The Multipath planner, which is shown in Fig. 13(a), produces an aggressive plan to traverse the intersection since its multiple contingency paths can account for each possible obstacle action independently. The Singlepath planner, which is shown in Fig. 13(b), is forced to take a conservative approach to the intersection since it must avoid all possible obstacle trajectories simultaneously. The Static planner, which is shown in Fig. 13(c), plans a clearly unsafe path because it is unable to anticipate the future motion of the obstacle.

A total of $n_{\text{sim}} = 300$ simulations were run, each 8 s in length, using a maximum probability of collision threshold of $p_{\max} = 10\%$. This $p_{\max}$ threshold was found to provide a good balance between driving performance and safety. Four performance metrics were recorded: collision frequency, minimum distance to obstacle, average squared acceleration, and minimum distance to goal. Statistics for these metrics, over the $n_{\text{sim}} = 300$ runs, are listed in Table I, where $\mu_{(\cdot)}$ is the mean,

TABLE I
PERFORMANCE METRICS FOR $n_{\text{sim}} = 300$ SINGLE OBSTACLE SIMULATION RUNS

			min. distance to obstacle		avg. squared acceleration		min. distance to goal	
	$n_{\text{coll}}^{\text{tot}}$	$n_{\text{coll}}^{\text{fault}}$	μ_{obst}	SE_{obst}	μ_{acc}	SE_{acc}	μ_{goal}	SE_{goal}
Multi	2.00	0.00	3.81	0.155	1.43	0.065	12.13	0.33
Single	2.00	0.33	4.43	0.190	1.25	0.058	12.80	0.35
Static	18.00	10.0	2.63	0.151	1.57	0.064	11.36	0.36
units	%	%	(m)		$(\text{m/s}^2)^2$		(m)	

TABLE II
PAIRED t-TEST RESULTS FOR SINGLE OBSTACLE SIMULATIONS AT $\alpha = 0.05$

	min. distance to obstacle		avg. squared acceleration		min. distance to goal	
comparison	H	P	H	P	H	P
Singlepath vs. Multipath	1	< 0.001	1	< 0.001	1	0.001
Static vs. Multipath	1	< 0.001	1	0.041	1	0.003

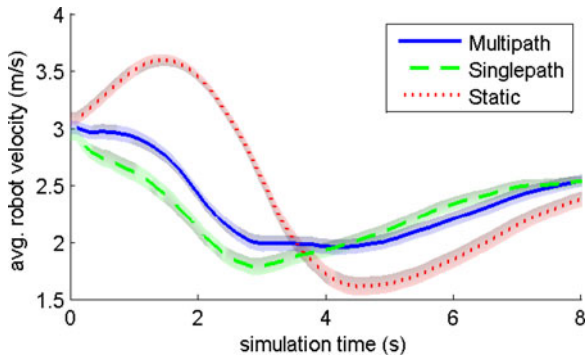

Fig. 14. Average velocity as a function of simulation time. Shaded regions reflect the one standard error of the mean (SE_{vel}) uncertainty bounds.

and $\text{SE}_{(\cdot)}$ is the standard error of the mean, i.e., $\text{SE}_{(\cdot)} = \frac{\sigma_{(\cdot)}}{\sqrt{n_{\text{sim}}}}$. Table I lists both the total percentage of simulations that resulted in a collision $n_{\text{coll}}^{\text{tot}}$ and the percentage of at-fault collisions $n_{\text{coll}}^{\text{fault}}$. At-fault collisions occur when the robot is genuinely behaving dangerously and are defined as collisions where the robot is still moving at a nonnegligible speed, $|v_{\text{robot}}| > 0.1$ m/s, at the time of the collision. Collisions that do not meet this criterion are considered not at fault because they likely would not result in a collision if the obstacle vehicle had any avoidance capabilities.

The collision frequency results in Table I show that both the Multipath and Singlepath planners experience far fewer total and at-fault collisions than the Static planner. Table I also shows that the Multipath planner is more aggressive in terms of obstacle spacing and speed through the intersection than the Singlepath planner, but it achieves this performance increase at the cost of increased acceleration effort. Table II shows the paired t-test results for each metric, indicating whether the difference in performance between the Multipath planner and the Static and Singlepath planners is statistically significant. Acceptance of the null hypothesis, i.e., H = 0, indicates that the difference between the two cases is not statistically significant. These tests indicate that all of performance differences seen in Table I are statistically significant at the $\alpha = 0.05$ confidence level.

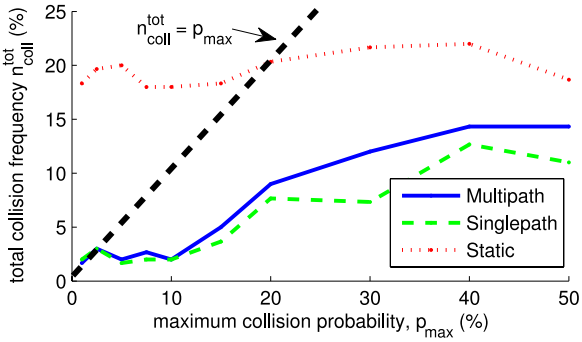

Fig. 15. Total collision frequency $n_{\mathrm{coll}}^{\mathrm{tot}}$ as a function of the collision probability bound $p_{\max}$.

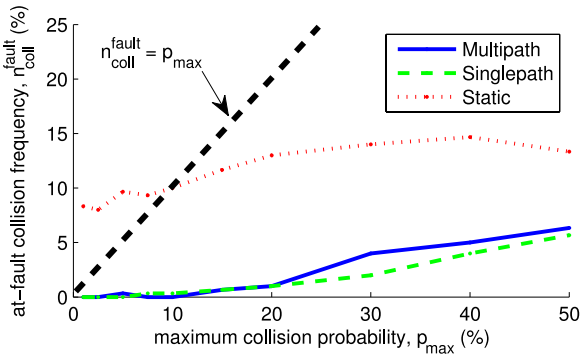

Fig. 16. At-fault collision frequency $n_{\mathrm{coll}}^{\mathrm{fault}}$ as a function of the collision probability bound $p_{\max}$.

Fig. 14 plots the velocity history of the robot averaged over all $n_{\mathrm{sim}} = 300$ simulations. This plot shows that the Multipath algorithm is more aggressive that the Singlepath planner at the start of the simulation, due to its less conservative assumptions about the obstacle's future motion. The Singlepath planner is less aggressive because it assumes that all mutually exclusive obstacle predictions are simultaneously occurring events. As the robot approaches the intersection during a typical simulation run, the predicted obstacle trajectories cause the intersection to appear completely blocked, forcing the robot to apply its brakes immediately. The Multipath planner, alternatively, is able to recognize that there may be a clear path by the time the robot reaches the intersection and allows the robot to postpone braking until it is necessary for safety reasons. This delayed braking is visible in the $t = 1$ to $t = 3$ s range of Fig. 14.

These results show that, while the performance of the Singlepath planner is similar to that of the Multipath planner in terms of collision avoidance, the Singlepath planner exhibits a more conservative driving style. The Multipath algorithm performs much closer to the Static algorithm in terms of driving aggressiveness, but is significantly safer, experiencing far fewer close calls and collisions.

B. Sensitivity to Collision Threshold

The maximum probability of collision threshold, $p_{\max}$ in (24), limits the robot's aggressiveness with respect to obstacle interactions. To investigate the effect of this parameter on the robot's performance, $n_{\mathrm{sim}} = 300$ simulations were run for a range of $p_{\max}$ values from 1% to 50%. Figs. 15–19 plot the average collision frequency and performance metrics as a function of $p_{\max}$ for the Multipath, Singlepath, and Static planners.

Figs. 15 and 16 plot the total and at-fault collision frequencies of each planning algorithm as a function of $p_{\max}$. Lines corresponding to $n_{\mathrm{coll}}^{\mathrm{tot}} = p_{\max}$ and $n_{\mathrm{coll}}^{\mathrm{fault}} = p_{\max}$ are also plotted for reference. Fig. 15 shows that at low values of $p_{\max}$ ($< 10\%$), there is little difference in total collision frequency between the Multipath and the Singlepath planners, while at high values of $p_{\max}$ ($> 40\%$), both planners begin to approach the collision frequency of the static planner. Compared with the other planners, the collision frequency of the Static planner is less sensitive to changes in $p_{\max}$. Since the Static planner is unable to anticipate obstacle motion, it encounters a high number of unavoidable

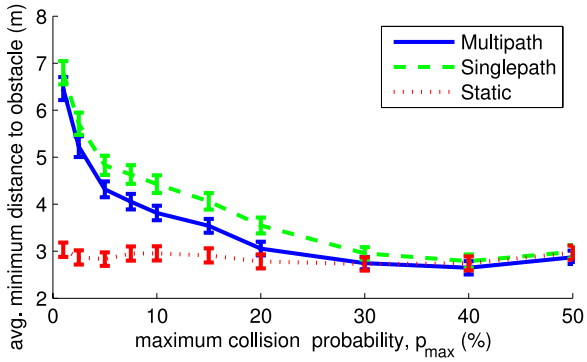

Fig. 17. Average minimum distance to obstacle as a function of the collision probability bound $p_{\max}$. Error bars indicate the standard error of the mean.

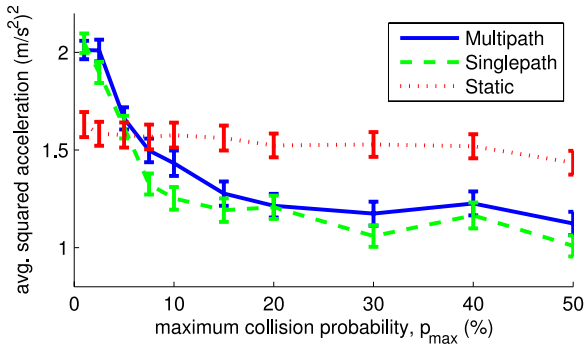

Fig. 18. Average squared acceleration as a function of the collision probability bound $p_{\max}$. Error bars indicate the standard error of the mean.

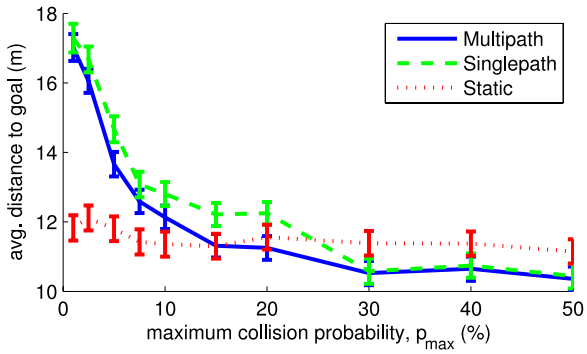

Fig. 19. Average distance to goal as a function of the collision probability bound $p_{\max}$. Error bars indicate the standard error of the mean.

collision scenarios, such as merging into an obstacle or being cut off or hit in the side/back by the obstacle; these cases are uncorrelated with $p_{\max}$. Fig. 16 shows a similar general trend for at-fault collision frequencies, with the notable difference that the Singlepath and Multipath planners are effectively collision-free ($n_{\text{coll}}^{\text{fault}} \leq 0.33\%$) for $p_{\max} \leq 10\%$. Additionally, the Static planner still experiences a large number of at-fault collisions over the entire range of $p_{\max}$ values.

Fig. 17 shows how the average minimum distance to obstacle metric varies as a function of $p_{\max}$. At very low values of $p_{\max}$ ($< 3\%$), the performance of the Multipath planner matches that of the Singlepath planner. As $p_{\max}$ increases, both the Singepath and the Multipath planners approach the performance of the Static planner; however, the Multipath planner does so at a faster rate. These results are intuitive, since at low values of $p_{\max}$, the Multipath planner attempts to avoid any possible obstacle trajectory, causing the robot to behave as conservatively as the Singlepath planner. At mid-to-high $p_{\max}$ values, the Multipath planner is able to accept some collision risk and actively utilizes its contingency plans to improve performance while maintaining low collision frequencies. Finally, at very high $p_{\max}$ values, collision risk is relatively unimportant, and the Singlepath planner approaches the aggressiveness of the other planners.

Figs. 18 and 19 plot the average squared acceleration and distance to goal metrics as a function of $p_{\max}$. Both of these plots show a similar trend: At very low values of $p_{\max}$ ($< 3\%$), the Multipath planner matches the Singlepath planner since both planners avoid any potential collision scenario. As $p_{\max}$ increases, the Singlepath and the Multipath planners diverge to a small degree, and both approach then exceed the performance of the Static planner. The acceleration results in Fig. 18 suggest that the aggressiveness of the Multipath planner causes a net loss in acceleration performance compared with the Singlepath planner over most of the tested $p_{\max}$ range. The Multipath planner allows for smaller accelerations when the robot correctly recognizes that its path is clear before entering the intersection. However, due to its increased aggressiveness, it also experiences larger decelerations in cases where the robot must stop to avoid an obstacle. For the distance to goal metric, the Multipath planner holds a noticeable advantage over the Singlepath algorithm for the range from $p_{\max} = 5\%$ to $p_{\max} = 30\%$. This advantage is due to the more aggressive driving style of the Multipath planner. For $p_{\max} > 30\%$, collision probability becomes less important, and the Singlepath Planner again approaches the performance of the Multipath planner.

C. Multiple Obstacle Simulation Results

Simulations for two scenarios were run to evaluate scaling, performance, and the effects of traffic density in multiple obstacle encounters. These scenarios were conducted using the obstacle trajectory clustering algorithm that is proposed in Section IV-A. Both scenarios consist of a robot and two obstacle vehicles navigating the intersection scenario depicted in Fig. 20. The first set of simulations represents a congested scenario, where the robot and two obstacle vehicles all reach the intersection at approximately the same time; the interior distributions in Fig. 20 depict the possible initial conditions for the robot and both obstacle vehicles for these congested simulations. The second set of simulations represents a less-congested scenario, where one obstacle reaches the intersection around the same time as the robot, but the other obstacle arrives later. The first obstacle is randomly assigned an initial condition from the interior set of distributions, as in the congested case, while the second obstacle is initialized using one of the exterior distributions, as shown in Fig. 20. Both sets of simulations included $n_{\text{sim}} = 500$ runs, each for 8 s, using maximum cluster values that range between $n_c = 1$ and $n_c = 4$. The Static planner was also tested to provide a baseline comparison. The Singlepath planner from the previous section is equivalent to the $n_c = 1$ case.

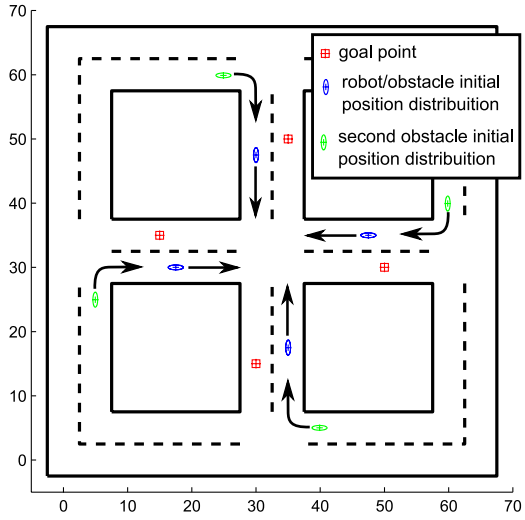

Fig. 20. Possible initial conditions for the multiple-obstacle simulations. For congested simulations, the robot and both obstacles are randomly assigned initial positions from the interior distributions. For less-congested simulations, one obstacle is initialized using one of the exterior distributions.

TABLE III
PERFORMANCE METRICS FOR $n_{\text{sim}} = 500$ CONGESTED SIMULATION RUNS

			min. distance to obstacles		avg. squared acceleration		min. distance to goal	
n_c	$n_{\text{coll}}^{\text{tot}}$	$n_{\text{coll}}^{\text{fault}}$	μ_{obst}	SE_{obst}	μ_{acc}	SE_{acc}	μ_{goal}	SE_{goal}
1	2.0	0.0	2.59	0.09	1.80	0.048	14.44	0.22
2	1.4	0.0	2.55	0.09	1.86	0.049	14.27	0.24
3	2.0	0.0	2.54	0.09	1.93	0.050	14.21	0.23
4	1.4	0.0	2.53	0.09	1.97	0.051	14.15	0.23
Static	30.2	11.4	1.69	0.08	2.26	0.46	13.45	0.27
units	%	%	(m)		$(\text{m/s}^2)^2$		(m)	

1) Congested Intersection: The congested simulations challenge the clustering algorithm since all predicted obstacle trajectories for both obstacles are potentially important and influence the robot over a similar time period. Table III presents the minimum distance to obstacle, average squared acceleration, and distance to goal performance metrics evaluated over all $n_{\text{sim}} = 500$ congested simulations. The total collision frequency $n_{\text{coll}}^{\text{tot}}$ and the total at-fault collision frequency $n_{\text{coll}}^{\text{fault}}$ are also presented.

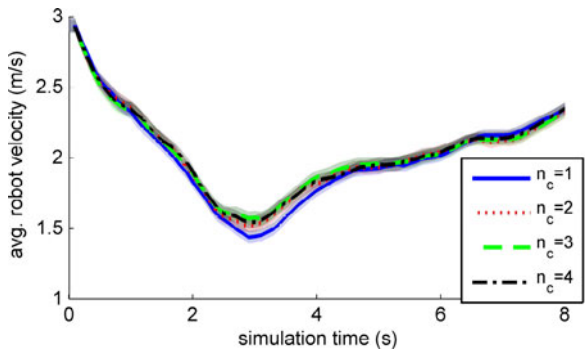

Fig. 21. Average velocity as a function of simulation time for the congested simulations. Shaded regions reflect the one standard error of the mean ($\mathrm{SE}_{\mathrm{vel}}$) uncertainty bounds.

TABLE IV
PAIRED t-TEST RESULTS FOR CONGESTED CASE AT $\alpha = 0.05$

	min. distance to obstacles		avg. squared acceleration		min. distance to goal	
comparison	H	P	H	P	H	P
$n_c = 2$ vs. $n_c = 1$	1	0.018	1	0.009	0	0.153
$n_c = 3$ vs. $n_c = 1$	1	0.004	1	< 0.001	0	0.058
$n_c = 4$ vs. $n_c = 1$	1	0.006	1	< 0.001	1	< 0.001

All metrics in Table III show clear trends as n_c varies: The minimum distance to obstacle and distance from goal metrics get smaller as n_c increases, while the average squared acceleration metric grows larger. The reason for these trends is that using multiple contingency paths allows the robot to delay making decisions, such as braking for obstacle avoidance, while there is still ambiguity in an obstacle's goal and while there is still time for the robot to react safely to any possible outcome. Since the robot has a small likelihood of being able to navigate the congested intersection without stopping, statistical differences in robot's performance as a function of n_c are small. If, as in this scenario, the robot delays braking but typically must stop and wait, the delayed braking decision causes the robot to 1) stop closer to the obstacle and 2) brake more aggressively. This ability to delay the braking decision also statistically improves the robot's performance with respect to the distance to goal metric. This delayed braking is visible in the $t = 2$ to $t = 4$ s range of Fig. 21, which shows the velocity history of the robot averaged over all $n_s = 500$ simulations. Despite this delayed decision making, the contingency planner, at all n_c values, has a significantly lower collision frequency than the Static planner, and increasing n_c appears to have no correlation with an increase in collision frequency. Table IV shows the paired *t*-test results for each metric, evaluating whether differences in the metric between the $n_c > 1$ cases and the $n_c = 1$ case are statistically significant. These tests indicate that the distance to obstacle and average squared acceleration metrics show a statistically significant change in performance over the entire range of n_c, while the distance to goal metric only becomes statistically significant for the $n_c = 4$ case.

2) Less-Congested Intersection: The less-congested simulations allow for more obvious obstacle trajectory clusterings since one obstacle's trajectories are clearly more important than those of the other. These simulations are also less challenging for contingency planner since the dynamic obstacles are more spread out both spatially and temporally. Table V presents the performance metrics for the less-congested simulations. The results show similar trends as in the congested simulations, with the minimum distance to obstacle and distance from goal metrics getting smaller as n_c increases, and the average squared acceleration metric growing larger. However, the changes in the minimum distance to obstacle and distance to goal metrics are more pronounced, while the changes in the average squared acceleration metric are nearly the same. Since the robot is less likely to require complete stops than in the congested simulations, it is better able to utilize available contingency plans to improve performance. Additionally, there is negligible difference between the $n_c = 3$ and $n_c = 4$ cases, reflecting the fact that the robot primarily must consider only one important obstacle at a time, reducing the number of important obstacle prediction clusters to 3. Fig. 22 shows the average velocity profiles for the less-congested simulations. These velocity profiles show that 1) the robot requires less braking to navigate the intersection compared with the congested simulations (see Fig. 21) and 2) that there is a significant difference between the conservativeness of the $n_c = 1$ case and the $n_c > 1$ cases.

Table VI shows the paired *t*-test results for the less-congested simulations. These tests show an increase in the statistical significance of the distance to obstacle and distance to goal metrics, while the significance of the average squared acceleration metric is similar to the congested case.

TABLE V
PERFORMANCE METRICS FOR $n_{\mathrm{sim}} = 500$ LESS-CONGESTED SIMULATION RUNS

			min. distance to obstacles		avg. squared acceleration		min. distance to goal	
n_c	$n_{\mathrm{coll}}^{\mathrm{tot}}$	$n_{\mathrm{coll}}^{\mathrm{fault}}$	μ_{obst}	SE_{obst}	μ_{acc}	SE_{acc}	μ_{goal}	SE_{goal}
1	2.2	1.2	3.10	0.094	1.82	0.055	14.53	0.300
2	1.4	0.8	2.95	0.086	1.87	0.058	13.95	0.299
3	1.8	0.8	2.86	0.086	1.98	0.062	13.71	0.297
4	1.4	0.6	2.89	0.083	1.97	0.061	13.66	0.301
Static	23.2	13.4	1.99	0.083	1.68	0.050	11.39	0.284
units	%	%	(m)		$(\mathrm{m/s^2})^2$		(m)	

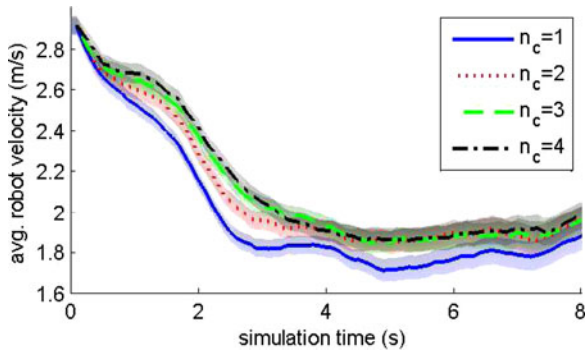

Fig. 22. Average velocity as a function of simulation time for the less-congested simulations. Shaded regions reflect the one standard error of the mean ($\mathrm{SE}_{\mathrm{vel}}$) uncertainty bounds.

TABLE VI
PAIRED t-TEST RESULTS FOR LESS-CONGESTED CASE AT $\alpha = 0.05$

	min. distance to obstacles		avg. squared acceleration		min. distance to goal	
comparison	H	P	H	P	H	P
$n_c = 2$ vs. $n_c = 1$	1	< 0.001	0	0.089	1	< 0.001
$n_c = 3$ vs. $n_c = 1$	1	< 0.001	1	< 0.001	1	< 0.001
$n_c = 4$ vs. $n_c = 1$	1	< 0.001	1	< 0.001	1	< 0.001

TABLE VII
COMPUTATION TIME STATISTICS

	single obstacle		multi-obstacle congested		multi-obstacle less-congested	
n_c	μ_{comp}	σ_{comp}	μ_{comp}	σ_{comp}	μ_{comp}	σ_{comp}
1	0.035	0.14	0.21	0.54	0.38	0.73
2	-	-	0.31	0.77	0.47	0.89
3	0.054	0.14	0.44	1.02	0.57	1.06
4	-	-	0.53	1.23	0.62	1.16
Static	0.032	0.07	0.23	0.39	0.18	0.34
units	s	s	s	s	s	s

D. Computational Performance

Table VII shows computation time statistics for each of the presented simulation scenarios. The computation time mean μ_{comp} and standard deviation σ_{comp}, are included for each value of n_c and for the static path planner. For the single obstacle case, the Multipath planner experiences a maximum of $n_s = 3$ obstacle predictions and is equivalent to setting $n_c = 3$. Similarly, the Singlepath planner is equivalent to $n_c = 1$. The single obstacle results show that the Multipath planner is capable of real-time performance for up to three simultaneous contingency paths, with nominal rates greater than 10 Hz.

The multiple obstacle computation time results in Table VII show that real-time contingency planning for $n_c \geq 4$ is difficult, and that the exponential scaling of exhaustive contingency planning is not a feasible option. In both the congested and less-congested scenarios, the mean computation time varies approximately linearly with n_c. The slight increase in average computation time for the less-congested simulations relative to the congested simulations is due to the more spaced out obstacles in the less-congested simulations, forcing the robot to perform nontrivial path planning over a longer portion of the simulation time window. The clustering algorithm was written in MATLAB and took between 0.1 and 0.45 s. The dominant computational cost in the clustering algorithm is the driving corridor inclusion evaluation, which is not dependent on the maximum number of clusters n_c and grows linearly with the number of obstacle predictions.

VI. CONCLUSION

A novel contingency path-planning algorithm has been presented, which provides a framework for the inclusion of anticipated, probabilistic obstacle motion in the planning process. This planner uses a novel path optimization algorithm to simultaneously optimize multiple continuous contingency paths with a shared initial segment, allowing the robot to account for multiple mutually exclusive obstacle predictions while maintaining a deterministic path to execute at the current timestep. The path optimization formulation uses a spline-based contingency path representation, and associated cost and constraint definitions, to enable fast optimization of the continuous contingency paths. A novel collision probability bound is also proposed which enables efficient computation of a tight bound on point-wise collision probability between oriented polygons with uncertain relative positions and orientations. This tight bound is critical to prevent overconservative behavior in common road driving scenarios. Additionally, an algorithm to cluster obstacle trajectories is presented to enable this contingency planning approach to scale to more complex, multiobstacle environments. Simulation results show that the presented Multipath algorithm is as safe in terms of collision frequency as the more conservative Singlepath approach, but is more aggressive in terms of obstacle spacing, and is faster through intersections. These results also hold for a wide range of values for the collision probability threshold $p_{\max}$. Similar performance trends occur when using trajectory clustering in multiple obstacle scenarios. As the number of allowed clusters n_c increases, the robot experiences increases in aggressiveness in terms of obstacle spacing and speed through intersections, with no associated increase in collision probability.

Future goals for this research focus on improving models for obstacle prediction, in order to allow the proposed contingency planning framework to be applied to a wider array of real-world driving scenarios. Model improvements being investigated include 1) learning obstacle behavior from previous interactions, 2) modeling additional obstacle types such as pedestrians and bicyclists, and 3) using both formal and social traffic rules to identify a broader set of possible obstacle intents.

REFERENCES

[1] M. Buehler, K. Iagnemma, and S. Singh, *The Darpa Urban Challenge: Autonomous Vehicles in City Traffic*. New York, NY, USA: Springer-Verlag, 2010.
[2] L. Fletcher, S. Teller, E. Olson, D. Moore, Y. Kuwata, J. How, J. Leonard, I. Miller, M. Campbell, D. Huttenlocher, A. Nathan, and F.-R. Kline, "The MIT-cornell collision and why it happened," *J. Field Robot.*, vol. 25, no. 10, pp. 775–807, 2008.
[3] A. Kushleyev and M. Likhachev, "Time-bounded lattice for efficient planning in dynamic environments," in *Proc. IEEE Int. Conf. Robot. Autom*, May 2009, pp. 4303–4309.
[4] J. Miura and Y. Shirai, "Probabilistic uncertainty modeling of obstacle motion for robot motion planning," *J. Robot. Mechatron.*, vol. 14, no. 4, pp. 349–356, 2002.
[5] I. Hwang and C. Seah, "Intent-based probabilistic conflict detection for the next generation air transportation system," *Proc. IEEE*, vol. 96, no. 12, pp. 2040–2059, Dec. 2008.
[6] M. Tay and C. Laugier, "Modelling smooth paths using Gaussian processes," in *Field Service Robotics* (Springer Tracts in Advanced Robotics Volume). New York, NY, USA: Springer-Verlag, 2008, pp. 381–390.
[7] J. Joseph, F. Doshi-Velez, A. Huang, and N. Roy, "A Bayesian nonparametric approach to modeling motion patterns," *Autonom. Robot.*, vol. 31, pp. 383–400, 2011.
[8] K. Kim, D. Lee, and I. Essa, "Gaussian process regression flow for analysis of motion trajectories," in *Proc. IEEE Int. Conf. Comput. Vis.*, Nov. 2011, pp. 1164–1171.
[9] M. Bennewitz, W. Burgard, G. Cielniak, and S. Thrun, "Learning motion patterns of people for compliant robot motion," *Int. J. Robot. Res.*, vol. 24, no. 1, pp. 31–48, 2005.
[10] P. Fiorini and Z. Shiller, "Motion planning in dynamic environments using velocity obstacles," *Int. J. Robot. Res.*, vol. 17, no. 7, pp. 760–772, 1998.

[11] G. Aoude, B. Luders, D. Levine, and J. How, "Threat-aware path planning in uncertain urban environments," in *Proc. IEEE/RSJ Int. Conf. Intell. Robot. Syst.*, Taipei, Taiwan, Oct. 2010, pp. 6058–6063.
[12] D. Ferguson, M. Darms, C. Urmson, and S. Kolski, "Detection, prediction, and avoidance of dynamic obstacles in urban environments," in *Proc. IEEE Intell. Veh. Symp.*, Jun. 2008, pp. 1149–1154.
[13] C. Urmson, J. Anhalt, D. Bagnell, C. Baker, R. Bittner, M. N. Clark, J. Dolan, D. Duggins, T. Galatali, C. Geyer, M. Gittleman, S. Harbaugh, M. Hebert, T. Howard, S. Kolski, A. Kelly, M. Likhachev, M. McNaughton, N. Miller, K. Peterson, B. Pilnick, R. Rajkumar, P. Rybski, B. Salesky, Y.-W. Seo, S. Singh, J. Snider, A. Stentz, W. Whittaker, Z. Wolkowicki, J. Ziglar, H. Bae, T. Brown, D. Demitrish, B. Litkouhi, J. Nickolau, V. Sadekar, W. Zhang, J. Struble, M. Taylor, M. Darms, and D. Ferguson, "Autonomous driving in urban environments: Boss and the urban challenge," *J. Robot. Syst.*, vol. 25, no. 8, pp. 425–466, 2008.
[14] I. Miller, M. Campbell, D. Huttenlocher, F.-R. Kline, A. Nathan, J. C. Sergei Lupashin, B. Schimpf, P. Moran, N. Zych, E. Garcia, M. Kurdziel, and H. Fujishima, "Team Cornell's skynet: Robust perception and planning in an urban environment," *J. Field Robot.*, vol. 25, no. 8, pp. 493–527, 2008.
[15] J. Hardy, M. Campbell, I. Miller, and B. Schimpf. (2008). Sensitivity analysis of an optimization-based trajectory planner for autonomous vehicles in urban environments. *Proc. SPIE* [Online]. vol. 7112, no. 1, p. 711211, Available:http://link.aip.org/link/?PSI/7112/711211/1
[16] M. Milam, K. Mushambi, and R. Murray, "A new computational approach to real-time trajectory generation for constrained mechanical systems," in *Proc. 39th IEEE Conf. Decis. Contr.*, 2000, vol. 1, pp. 845–851.
[17] L. Blackmore, H. Li, and B. Williams, "A probabilistic approach to optimal robust path planning with obstacles," presented at the Amer. Contr. Conf., Minneapolis, MN, USA, 2006.
[18] J. Bellingham, A. Richards, and J. How, "Receding horizon control of autonomous aerial vehicles," in *Proc. Amer. Contr. Conf.*, 2002, vol. 5, pp. 3741–3746.
[19] L. Cremean, T. Foote, J. Gillula, G. Hines, D. Kogan, K. Kriechbaum, J. Lamb, J. Leibs, L. Lindzey, C. Rasmussen, A. Stewart, J. Burdick, and R. Murray, "Alice: An information-rich autonomous vehicle for high-speed desert navigation," *J. Field Robot.*, vol. 23, no. 9, pp. 777–810, 2006.
[20] J. Hardy and M. Campbell, "Contingency planning over probabilistic hybrid obstacle predictions for autonomous road vehicles," in *Proc. IEEE/RSJ Int. Conf. Intell. Robot. Syst.*, Oct. 2010, pp. 2237–2242.
[21] J. Hardy and M. Campbell, "Clustering obstacle predictions to improve contingency planning for autonomous road vehicles in congested environments," in *Proc. IEEE/RSJ Int. Conf. Intell. Robot. Syst.*, Sep. 2011, pp. 1605–1611.
[22] I. Miller, M. Campbell, and D. Huttenlocher, "Efficient, unbiased tracking of multiple dynamic obstacles under large viewpoint changes," *IEEE Trans. Robot.*, vol. 27, no. 1, pp. 29–46, Feb. 2011.
[23] S. Julier and J. Uhlmann, "A new extension of the Kalman filter to nonlinear systems," in *Proc. Int. Symp. Aerosp./Defense Sens. Simul. Contr.*, 1997, vol. 3, p. 26.
[24] A. Forsgren, P. Gill, and M. Wright, "Interior methods for nonlinear optimization," *SIAM Rev.*, vol. 44, no. 4, pp. 525–597, 2002.
[25] P. Boggs and J. Tolle, "Sequential quadratic programming," *Acta Numer.*, vol. 4, no. 1, pp. 1–51, 1995.
[26] R. Geraerts and M. Overmars, "Creating high-quality paths for motion planning," *Int. J. Robot. Res.*, vol. 26, no. 8, pp. 845–863, 2007.
[27] R. Wein, J. Van Den Berg, and D. Halperin, "Planning high-quality paths and corridors amidst obstacles," *Int. J. Robot. Res.*, vol. 27, no. 11–12, pp. 1213–1231, 2008.
[28] D. Dolgov, S. Thrun, M. Montemerlo, and J. Diebel, "Path planning for autonomous driving in unknown environments," in *Experimental Robotics*, (Springer Tracts in Advanced Robotics). New York, NY, USA: Springer-Verlag, 2009, pp. 55–64.
[29] O. Aichholzer, F. Aurenhammer, D. Alberts, and B. Gaertner, "A novel type of skeleton for polygons," *J. Universal Comput. Sci.*, vol. 1, no. 12, pp. 752–761, 1995.
[30] R. Paielli, H. Erzberger, and A. R. Center, "Conflict probability estimation for free flight," *J. Guid. Control Dyn.*, vol. 20, no. 3, pp. 588–596, 1997.
[31] S. Alfano, "Addressing nonlinear relative motion for spacecraft collision probability," in *Proc. 15th AAS/AIAA Astrodynamics Specialist Conf.*, 2006, p. 15, AIAA Paper no. 2006-6760.
[32] A. Lambert, D. Gruyer, G. Pierre, and A. Ndjeng, "Collision probability assessment for speed control," in *Proc. 11th Int. IEEE Conf. Intell. Transp. Syst.*, Oct. 2008, pp. 1043–1048.
[33] Z. Fu, W. Hu, and T. Tan, "Similarity based vehicle trajectory clustering and anomaly detection," in *Proc. IEEE Int. Conf. Imag. Process.*, Sep. 2005, vol. 2, pp. II–602–5.
[34] J. Lee, J. Han, and K. Whang, "Trajectory clustering: A partition-and-group framework," in *Proc. ACM SIGMOD Int. Conf. Manag. Data*, 2007, pp. 593–604.
[35] S. Gaffney and P. Smyth, "Trajectory clustering with mixtures of regression models," in *Proc. 5th ACM SIGKDD Int. Conf. Knowl. Disc. Data Min.*, 1999, pp. 63–72.
[36] R. Coulter, "Implementation of the pure pursuit path tracking algorithm," The Robotics Inst., Carnegie Mellon Univ., Pittsburgh, PA, USA, Tech. Rep. CMU-RI-TR-92-01, Jan. 1992
[37] A. Wächter and L. Biegler, "On the implementation of an interior-point filter line-search algorithm for large-scale nonlinear programming," *Math. Programm.*, vol. 106, no. 1, pp. 25–57, 2006.

Jason Hardy received the B.S. degree in mechanical engineering from the University of California at Santa Barbara, Santa Barbara, CA, USA, in 2007 and the M.S. degree in aerospace engineering from Cornell University, Ithaca, NY, USA, in 2011, where he is currently working toward the Ph.D. degree in the field of dynamics, systems, and controls.

In 2009, he received a National Defense Science and Engineering Graduate Fellowship. His research interests include path planning and collision avoidance for autonomous vehicles.

Mark Campbell (M'00) received the B.S. degree in mechanical engineering from Carnegie Mellon University, Pittsburgh, PA, USA, in 1990 and the M.S. and Ph.D. degrees in control and estimation from the Massachusetts Institute of Technology, Cambridge, MA, USA, in 1993 and 1996, respectively.

He is a Professor and the S. C. Thomas Sze Director of Mechanical and Aerospace Engineering with Cornell University, Cornell University, Ithaca, NY, USA. His research interests include estimation and control of autonomous vehicles.

Dr. Campbell received a National Aeronautics and Space Administration (NASA) award for the Middeck Active Control Experiment flown on STS-67, Best Paper awards from the American Institute of Aeronautics and Astronautics (AIAA) and the Frontiers in Education, and teaching awards from Cornell University, the University of Washington, and the American Society for Engineering Education. He is an Associate Fellow of the AIAA, an Australian Research Council International Fellow, and an Associate Director of the American Automatic Control Council Board.