A 'Cognitive Driving Framework' for Collision Avoidance in Autonomous Vehicles

Alan J. Hamlet
Department of Mechanical and
Aerospace Engineering
University of Florida
Gainesville, Florida, USA

Carl D. Crane
Department of Mechanical and
Aerospace Engineering
University of Florida
Gainesville, Florida, USA

***Abstract*—The Cognitive Driving Framework is a novel method for forecasting the future states of a multi-agent system that takes into consideration both the intentions of the agents as well as their beliefs about the environment. This is particularly useful for autonomous vehicles operating in an urban environment. The algorithm maintains a posterior probability distribution over agent intents and beliefs in order to more accurately forecast their future behavior. This allows an agent navigating the environment to recognize dangerous situations earlier and more accurately than competing algorithms, therefore allowing the agent take actions in order to prevent collisions. This paper presents the Cognitive Driving Framework in detail and describes its application to intersection navigation for autonomous vehicles. The effects of different parameter choices on the performance of the algorithm are analyzed and experiments are conducted demonstrating the ability of the algorithm to predict and prevent automobile collisions caused by human error in multiple intersection navigation scenarios. The results are compared to the performance of prevailing methods; namely reactionary planning and constant velocity forecasting.**

***Keywords*—Multi-agent systems; autonomous vehicles; intent prediction; non-linear filtering; Bayesian filtering;**

I. Introduction

The potential safety and convenience benefits that autonomous vehicles can provide to our society are myriad. The World Health Organization reported that in 2010, 1.24 million people died due to road vehicle accidents.[1] In addition to the potential of reducing this massive loss of life, autonomous vehicles have shown promise in increasing vehicle efficiency and convenience for drivers [1]–[3].

The vast majority of current autonomous vehicle architectures employ a reactionary response to changes in the environment. These systems require very frequent and rapid re-planning in order to avoid dynamic obstacles. Another intuitive approach is to have the autonomous vehicle predict where the dynamic obstacles are going to be in order to plan a path. One popular approach to making this prediction is to assume the dynamic obstacle continues to move in a straight line at its current velocity, as is done in the 'velocity obstacle' literature [4], [5]. Cornell's autonomous vehicle, Skynet, uses this type of approach by using an extended Kalman filter to track dynamic obstacles and then calculates the 'time to collision' assuming a constant speed and heading [6]. This approach does not take into account the control decisions made by the dynamic obstacle that affect its trajectory, as is the case for pedestrians and other vehicles.

Some research has begun to incorporate the intentions of the dynamic obstacle in order to more intelligently predict its future position. Some methods used to predict intent are hidden Markov models [7], [8], Markov decision processes [9], and Gaussian processes or mixture models [10], [11]. These methods attempt to model trajectories and classify the dynamic obstacles' motion according to the corresponding intent. While this body of research is a step toward realizing more intelligent vehicles that truly understand their environment, it fails to consider how the obstacles' understanding of the environment will affect its future state.

The aforementioned types of planners work sufficiently well for navigating in urban environments where other vehicles are driving safely, but widespread adoption of autonomous vehicles will take time and human driven vehicles will remain on the roads for many years. With the presence of non-autonomous vehicles, the potential for accidents caused by human error will persist. According to a research study by the National Highway Traffic Safety Administration, 93% of traffic accidents were caused by human error.[2] Autonomous vehicles need to be able to operate alongside human drivers and prevent these potential collisions caused by human error. Reactive planners will often fail to recognize these dangerous situations in time to prevent a collision. Using constant velocity forecasting can result in overly cautious driving, due to frequent false predictions of dangerous situations.

In this research, both the intent and the belief of a dynamic obstacle are considered when modeling the future states of the obstacle. This is beneficial for situations in which a dynamic obstacle, e.g. a pedestrian or another vehicle, may have an incorrect belief about the environment. For example, an obstacle vehicle trying to merge into traffic may believe it has more space than it actually does or it may not see an oncoming vehicle due to occlusions or driver error. In these situations, just knowing the driver's intent does not suffice since for the same intent she may yield or begin to merge depending on her belief.

[1]http://www.who.int/gho/road_safety/mortality/traffic_deaths_number/en

[2]http://www.nhtsa.gov/people/injury/research/udashortrpt/background.html

In this paper, the dynamics of a multiple-vehicle system are modeled as a dynamic Bayesian network (DBN). Obstacle vehicles' actions are dependent on both their intent and their belief of the surrounding environment. This idea is similar to that proposed in [12], but in this paper the problem is not formulated as a Markov decision process as to avoid discretization of the state space. This is required in order to achieve the resolution necessary for the autonomous vehicle domain. Inference is performed over the network using a particle filter to jointly estimate the vehicle's intent and belief. Future vehicle states are then forecast using Monte Carlo simulation and the probability of a future collision is calculated. Simulation results show that this method of joint inference allows an autonomous vehicle to predict a collision with enough time to take evasive action.

The remainder of this paper is structured as follows. In section 2, an overview of the cognitive driving framework is given. The manner of representing the system state and dynamics is described. In section 3, the process for formulating the problem as inference over a dynamic Bayesian network is explained. The structure of the DBN is detailed and the method of performing joint inference over the network using particle filtering is discussed. At the end of section 3, forecasting the future state of the system using Monte Carlo simulation is explained. Next, in section 4, a detailed analysis is given on how different parameter choices affect the performance of the algorithm. Then simulation results demonstrating the accuracy of the proposed method are presented. Finally, concluding remarks are given and future research directions are discussed in section 5.

II. The Cognitive Driving Framework

This section provides an overview of the cognitive driving framework by describing how the state of an intersection environment with multiple vehicles is represented and by defining the form of the system dynamics.

In the cognitive driving framework, or CDF, the system consists of two vehicles, the obstacle vehicle and the ego vehicle, in a known environment. The joint state of the two vehicles is called the system pose and is represented as

$$\boldsymbol{S}_t = \begin{bmatrix} {}^1\boldsymbol{x}_t \\ {}^2\boldsymbol{x}_t \end{bmatrix}, \tag{1}$$

where a superscript 1 denotes the obstacle vehicle and a superscript 2 denotes the ego vehicle. In this paper, the term 'ego vehicle' refers to the vehicle that is trying to predict the intent of the obstacle vehicle.

In order to provide a general algorithm, the system dynamics are assumed to be nonlinear and of the form

$$\boldsymbol{S}_{t+1} = \begin{bmatrix} {}^1\boldsymbol{x}_{t+1} \\ {}^2\boldsymbol{x}_{t+1} \end{bmatrix} = \begin{bmatrix} f({}^1\boldsymbol{x}_t, {}^1\boldsymbol{u}_t, {}^1\boldsymbol{\nu}_t) \\ f({}^2\boldsymbol{x}_t, {}^2\boldsymbol{u}_t, {}^2\boldsymbol{\nu}_t) \end{bmatrix}, \tag{2}$$

where ${}^i\boldsymbol{u}_t$ is the control input and ${}^i\boldsymbol{\nu}_t$ is the process noise for vehicle *i* at time *t*. The controller for the autonomous vehicle running the CDF (the ego vehicle) is assumed to be of the form

$${}^2\boldsymbol{u}_t = h({}^2\boldsymbol{x}_t, {}^1\boldsymbol{x}_t, {}^2I), \tag{3}$$

where the arguments to the nonlinear function *h()* are the vehicle's own state, the state of the obstacle vehicle, and the intent of the ego vehicle, respectively, at time *t*. The intent variable, ${}^iI$, represents the current behavior the vehicle is trying to execute (e.g. turn left or go straight through the intersection). Here the controller, *h()*, is both highly nonlinear and discontinuous as it is a function of both continuous and discrete variables. The nonlinearities arise not only from the piecewise nature due the discrete intent variable, but also from the nonlinear kinematics of the system and the nonlinear dependence on the obstacle vehicle state.

The controller for the obstacle vehicle is modeled similarly as

$${}^1\boldsymbol{u}_t = h({}^1\boldsymbol{x}_t, \boldsymbol{B}_t, {}^1I). \tag{4}$$

The difference here is that the obstacle vehicle is not assumed to have exact knowledge of the ego vehicle's state. Instead, the obstacle vehicle's controller operates on the assumed state of the ego vehicle, the *belief*, $\boldsymbol{B}_t$. It should be noted that in this context the belief is simply a point, not a distribution or density as sometimes used in the literature. If the ego vehicle has not been observed by the obstacle vehicle, then $\boldsymbol{B}_t = \emptyset$. The obstacle updates its belief according to the equations

$$\boldsymbol{B}_{t+1} = g(\boldsymbol{B}_t, \boldsymbol{O}_{t+1}) \tag{5}$$

$$\boldsymbol{O}_t = k(\boldsymbol{S}_t, \beta, \boldsymbol{e}_t), \tag{6}$$

where $\boldsymbol{O}_t$ is the obstacle vehicle's observation at time *t*, and β is a parameter that represents the probability of the obstacle vehicle observing the ego vehicle at any given discrete time step. The observation noise, $\boldsymbol{e}_t$, is normally distributed with a mean of zero. Given $\boldsymbol{B}_t$ and $\boldsymbol{O}_{t+1}$, $\boldsymbol{B}_{t+1}$ updates deterministically. The observation model, *k()*, determines from the system pose if the ego vehicle is in the obstacle vehicle's *isovist*: the volume of space with line of sight visibility from the obstacle vehicle's pose. If the ego vehicle is occluded by other vehicles or buildings, it will not be in the obstacle vehicle's isovist, and thus $\boldsymbol{O}_t = \emptyset$. If the ego vehicle is in the obstacle vehicle's isovist, then the obstacle vehicle will make a noisy observation of the ego vehicle's pose with probability β.

The goal of the cognitive driving framework is to allow the ego vehicle to predict the future states of the obstacle vehicle using this model in order to prevent collisions. This is done by performing online inference of the obstacle vehicle's belief and intent, $\boldsymbol{B}_t$ and I_t. The following section details the procedure for performing this joint inference and prediction.

III. Filtering and Forecasting

In this section, the model outlined in the previous section is formulated as a dynamic Bayesian network. How online inference is performed using a particle filter is described and a procedure for using Monte Carlo simulation for forecasting future system states is presented.

A. Dynamic Bayesian Network

The cognitive driving framework uses a dynamic Bayesian network to capture the dependencies between the random variables in the CDF system dynamics. A Bayesian network is a directed acyclic probabilistic graphical model that is used to represent a set of random variables and their conditional dependencies. A *dynamic* Bayesian network is a Bayesian network

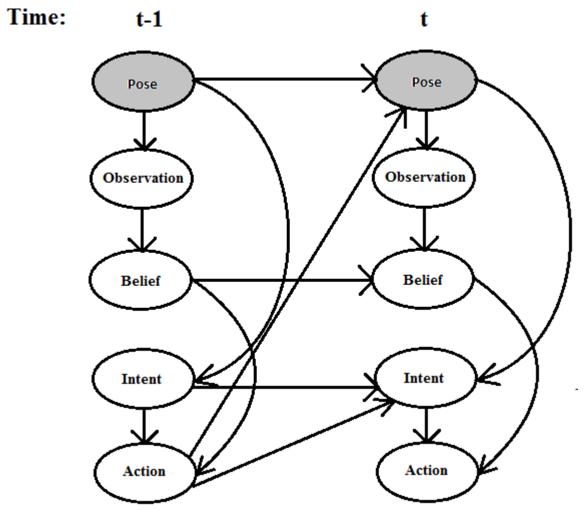

Fig. 1: The structure of the DBN used in the cognitive driving framework.

which relates the variables to each other over sequential time steps. In literature, dynamic Bayesian networks are sometimes referred to as *two-time-slice Bayesian networks* because at any point in time *t*, the value of a variable in the network can be calculated from the prior value (at time *t-1*) and the independent variables [13]. Kalman filter models and Hidden Markov Models are special cases of DBN's. In Kalman filter models, both the system dynamics and the measurements are assumed to be linear Gaussian. In hidden Markov models, the dynamics and measurements both have discrete distributions. DBN's make no assumptions about the form of the dynamics or measurements and allow the hidden state of the system to be factored into separate variables so the structure of the dependencies between the variables can be exploited.

The structure of the DBN used in the CDF is depicted in figure 1. This graphical model reflects the dependencies given by the equations in section II. The gray nodes in the graph denote the variable known by the ego vehicle, the system pose, $\boldsymbol{S}_t$, as given in equation 1. In some contexts, because the value of this variable is provided to the ego vehicle by its sensors, it is called the observation. In this work, the observation, $\boldsymbol{O}_t$, refers to the obstacle vehicle's noisy measurement of the ego vehicle's pose, ${}^2\boldsymbol{x}_t$.

Between time-slices, the variables in the DBN flow temporally from left to right and within a time-slice they flow (more-or-less) from top to bottom. The system pose affects the obstacle vehicle's observation which in turn determines the obstacle vehicle's belief. The obstacle vehicle's intent and belief of the system pose inform the obstacle vehicle's controller. The joint actions of the two vehicles result stochastically in the next system pose. Without loss of generality, the intent of the obstacle vehicle is assumed to be constant throughout an episode.

B. Filtering

Now that the two-vehicle system dynamics are represented as a DBN, a method of filtering needs to be implemented

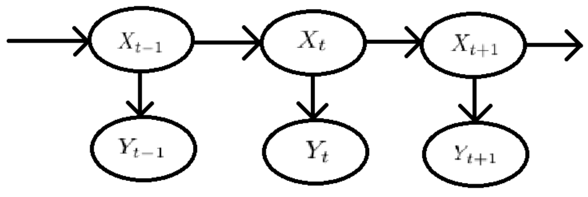

Fig. 2: Simplified model of the CDF using the joint DBN state variable X_t.

in order to perform online inference of the obstacle vehicle's belief and intent. By combining equations 2 through 6 we can represent the DBN state and dynamics, respectively, as

$$\boldsymbol{X}_t = [\boldsymbol{S}_t, \boldsymbol{B}_t, I]^T \tag{7}$$

$$\boldsymbol{X}_{t+T} \sim P(\boldsymbol{X}_{t+T} | \boldsymbol{X}_{t+T-1}) \tag{8}$$

This condenses the DBN in figure 1 to that shown in figure 2, which is the typical representation for filtering problems. The variable $\boldsymbol{Y}_t$ represents the measurement, which in this study is the system pose, $\boldsymbol{S}_t$.

The system dynamics are highly non-linear, as shown in section II . Instead of linearizing the dynamics at the expense of accuracy of the estimation, a non-linear Monte Carlo based filtering method was employed. Monte Carlo (MC) methods are ideally suited for the current application due to their ability to model highly non-linear systems with multi-modal, non-Gaussian distributions [14]. In this study, the DBN state is composed of both continuous and discrete variables, representing discrete intention hypotheses making traditional linearization methods such as the Extended or Unscented Kalman Filter unsuitable for this application.

Particle filters are sequential Monte Carlo methods that maintain an estimate of the posterior distribution of the system state as a set of particles. This non-parametric representation is capable of representing arbitrarily complex distributions as long as a large enough particle set is used. Each particle is initialized according to the a priori distribution and is propagated through the noisy system dynamics. The particles are then re-sampled according to the particles' importance weights. The importance weight of a particle is proportional to the likelihood of the particle generating the measurement, $\boldsymbol{Y}_t = \boldsymbol{S}_t$. In this study, the likelihood is represented as a Gaussian distribution centered around the measured system pose. The weight of each particle is proportional to the probability of the system pose of the particle given the measured system pose, as shown in the equation below.

$$w_t^{[m]} \propto P(\boldsymbol{X}_t^{[m]} | \boldsymbol{S}_t) \sim \mathcal{N}(\boldsymbol{S}_t, \boldsymbol{\Sigma}). \tag{9}$$

$$\boldsymbol{X}_t^{[m]} \sim P(\boldsymbol{X}_t^{[m]} | \boldsymbol{X}_{t-1}^{[m]}) \tag{10}$$

$\boldsymbol{\Sigma}$ is the variance of the Gaussian likelihood function. A superscript $[m]$ denotes that the variable corresponds to the m^{th} particle. The weights are normalized such that they sum to one.

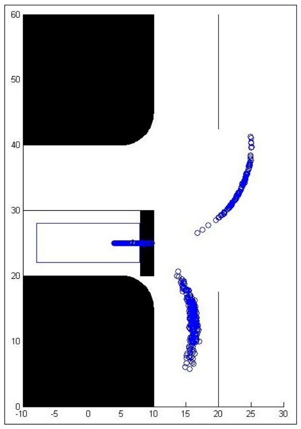

Fig. 3: The distribution of particles using Monte Carlo simulation for a look-ahead time of 2.5 seconds.

Once the importance weights are calculated, the particles are re-sampled in order to move the posterior distribution toward the region of the state space that matches the measurement. The technique of *stratified* or *low-variance* sampling is employed to reduce computational complexity [15]. The resulting set of particles is an approximation to the actual state of the system. As the number of particles, *M*, approaches infinity, the particle set converges to the true distribution of the state.

It is worthy of note that the version of the Cognitive Driving Framework presented here does not require that the obstacle vehicle controller (equation 4) be know; in fact, nor is it required that the system dynamics (equation 2) are known. All that is required for the CDF is that a simulation of the DBN system dynamics is available, i.e. that samples can be generated from the distribution $P(\boldsymbol{X}_{t+1}|\boldsymbol{X}_t)$.

C. Forecasting

Using the particle filtering method described in the previous subsection, an online estimate of the system state can be maintained. To predict and anticipate collisions in the future, though, the future state of the system must be estimated. To accomplish this, the CDF uses Monte Carlo simulation to propagate the particle set representing the current system state, $\boldsymbol{X}_t$, forward in time, creating a new particle set, $\boldsymbol{X}_{t+T}$, that approximates the state of the system at some time $t+T$ in the future. This is done by recursively sampling the DBN system dynamics given in equation 8. The probability of a collision can then be calculated from this new particle set by simply determining the percentage of particles in the set that represent a collision state.

An example future state distribution is shown in figure 3. Here, a vehicle is at a stop sign at a T-intersection and has the option to turn left or right. If it is turning left, it may also choose to yield if there is oncoming traffic. In the figure, the set of particles for a look-ahead time of 2.5 seconds is plotted. Three distinct modes can be seen in the distribution corresponding to the three potential behaviors of the vehicle: turning left, turning right, and yielding. This situation will

```
 1: **Algorithm Cognitive_Driving_Framework**($\chi_{t-1}, \boldsymbol{S}_t$)
 2: $\chi_t = \overline{\chi}_t = \emptyset$
 3: for $m = 1$ to $M$ do
 4:     $\overline{\boldsymbol{X}}_t^{[m]} \sim P(\boldsymbol{X}_t^{[m]}|\boldsymbol{X}_{t-1}^{[m]})$
 5:     $w_t^{[m]} = P(\overline{\boldsymbol{X}}_t^{[m]}|\boldsymbol{S}_t)$
 6:     $\overline{\chi}_t = \overline{\chi}_t + \langle \boldsymbol{X}_t^{[m]}, w_t^{[m]} \rangle$
 7: $\chi_t = Resample(\overline{\chi}_t, w_t)$
 8: $\chi_{t+T} = \chi_t$
 9: $collisions = 0$
10: for $m = 1$ to $M$ do
11:     for $\tau = 1$ to $T$ do
12:         $\chi_{t+\tau}^{[m]} \sim P(\chi_{t+\tau}^{[m]}|\chi_{t+\tau-1}^{[m]})$
13:     $collisions = collisions + CheckCollision(\chi_{t+T}^{[m]})$
14: $P(collision) = collisions/M$
15: if $P(collision) \geq Threshold$ then
16:     $EmergencyStop$
17: $return\ \chi_t$
```

Fig. 4: Pseudo-code overview of an update for the Cognitive Driving Framework algorithm.

be discussed and analyzed in more detail in the experimental results, section IV.

The cognitive driving framework algorithm is given in figure 4. In lines 2 through 5 the particle filter update is performed on the particle set, χ. Line 2 samples the next state from the DBN dynamics and line 3 sets the weight for the new sample based on the measured system pose. After this is done for all the particles, this weighted particle set, $\overline{\chi}_t$, is resampled according to the weights in order to move the distribution of the particles toward the measurement, as shown in line 5. In line 6, the particle set representing the future state of the system, χ_{t+T} is initialized to be equal to the set representing the current state, χ_t. Line 7 initially sets the number of particles in a collision state to zero. Lines 8 through 11 recursively propagate each particle one at a time through the system dynamics to obtain samples of the system state T time steps in the future. After each sample is propagated through the dynamics T times, it is checked to see if it is in a collision state in line 11, and the number of particles in a collision state is counted. Line 12 then calculates the probability of a collision as the number of particles in a collision state divided by the total number of particles. If the probability of a collision is higher than the set threshold, then an emergency braking maneuver is triggered in line 14. Otherwise, the algorithm just returns the particle set representing the current system state, χ_t.

There are multiple parameters used in this algorithm, namely, the look-ahead, T, the threshold, $Threshold$, and the number of particles, M. The selection of values for these parameters and their influence on the performance of the algorithm is discussed in section IV-B.

IV. Experimental Results

In order to demonstrate the ability of the CDF to perform joint inference on the intent and belief of an obstacle vehicle as well as forecast the future state of the multi-agent system, two

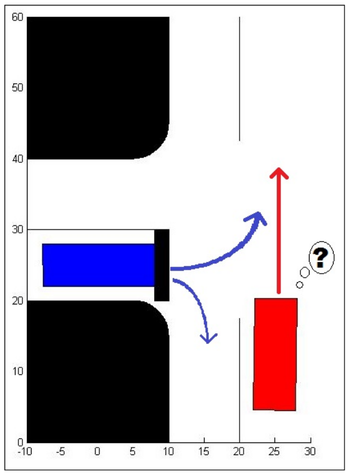

Fig. 5: A close up image of the simulated intersection environment for scenario one. The ego vehicle (red) attempts to predict the intent of the obstacle vehicle (blue).

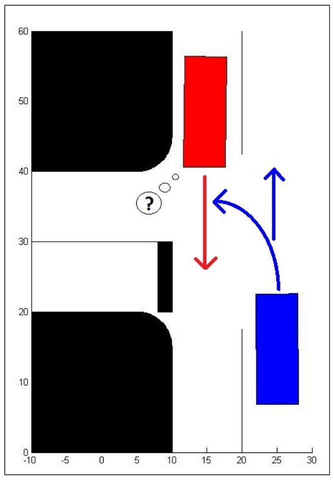

Fig. 6: A close up image of the simulated intersection environment for scenario two. The ego vehicle (red) attempts to predict the intent of the obstacle vehicle (blue).

simulated experiments were conducted. For comparison, the same experiments were also conducted using both a reactive planner and a constant velocity planner. This section first describes the simulation set up for the two tested scenarios. Next, a detailed analysis is performed to determine the optimal choice of parameters as well as examine the effects of parameter choice on the performance of the CDF algorithm. Lastly, the simulation results using all three methods are presented and discussed.

A. Simulation Setup

The CDF was tested using two simulated T-intersection scenarios. Scenario one is depicted in figure 5. The autonomous vehicle (the ego vehicle, in red) has the right of way. An obstacle vehicle (blue) is stopped at a stop sign at the intersection and can either turn left into the ego vehicle's lane or turn right. It is desirable for the ego vehicle to predict not just the intention of the obstacle vehicle to turn left, but whether the obstacle vehicle is going to turn left in front of the ego vehicle or if it is going to yield.

Scenario two takes place in the same intersection environment, but the ego vehicle is now traveling south (toward the bottom of the figure) through the intersection. The obstacle vehicle is driving north toward the intersection and has the option to proceed straight through the intersection or to turn left. Again, it is desirable to predict not just whether the obstacle vehicle is going to turn left or go straight, but whether the obstacle vehicle is going to turn left in front of the ego vehicle, yield to the ego vehicle, or go straight.

The simulation uses a bicycle kinematic model for the vehicles as described in [10]. The pose for a vehicle from equation 1 is given by the four dimensional vector

$$\boldsymbol{x}_t = \begin{bmatrix} x_t \\ y_t \\ \theta_t \\ v_t \end{bmatrix}, \qquad (11)$$

where x_t and y_t are the position of the center of the rear axle in the ground plane, θ_t is the vehicle's orientation, and v_t is the speed of the vehicle, all at time t. The superscript indicating which vehicle the pose corresponds to has been left off here for clarity. The dynamics from equation 2 are then given by

$$\boldsymbol{x}_{t+1} = f(\boldsymbol{x}_t, \boldsymbol{u}_t, \boldsymbol{\nu}_t) = \begin{bmatrix} x_t + v_t \cdot \Delta t \cdot \cos\theta_t \\ y_t + v_t \cdot \Delta t \cdot \sin\theta_t \\ \theta_t + \frac{v_t \cdot \Delta t}{l} \cdot \tan\left({}_2u_t + {}_2\nu_t\right) \\ v_t + \left({}_1u_t + {}_1\nu_t\right) \cdot \Delta t \end{bmatrix} \qquad (12)$$

where the elements of the two dimensional control input are acceleration, ${}_1u_t$, and steering angle, ${}_2u_t$. The two components of the process noise, ${}_1\nu_t$ and ${}_2\nu_t$, are both zero mean Gaussian noise affecting the realization of the controller's commanded acceleration and steering angle, respectively. The parameter l is the wheelbase of the vehicle. In these simulations, a time step, Δt, of 0.1 seconds is used.

The controller used in this simulation is a piecewise function that is composed of a different path following controller for each intent, *I*. A hand tuned finite state machine determines whether the vehicle should yield to the other vehicle or if it is clear to proceed. As shown in equations 3 and 4, the ego vehicle determines its control input based on the known poses of both vehicles, while the obstacle vehicle only has access to its own pose and a noisy estimate of the ego vehicle's pose.

B. Parameter Analysis

In this section, the affect of varying the parameter values in the CDF algorithm are analyzed. In particular, the look-ahead (how far into the future to forecast the system state) and the threshold (the collision probability at which evasive action should be triggered) are examined.

When analyzing the performance of the CDF algorithm, two key metrics were considered: the percentage of imminent collisions avoided and the number of false positive predictions of an imminent collision. Both of these metrics need to be considered when selecting values for the look-ahead and the

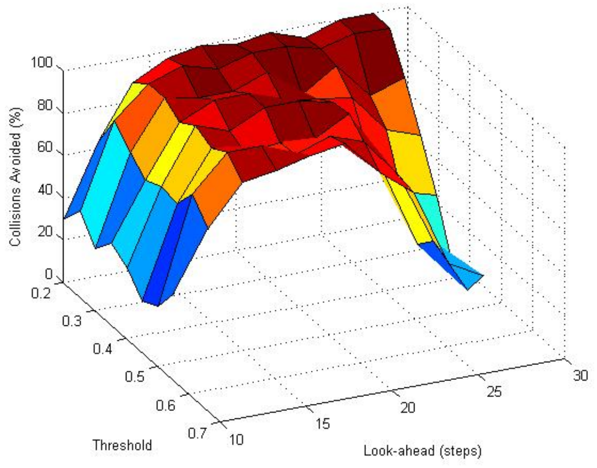

Fig. 7: How CDF performance varies with different values for the look-ahead and the threshold.

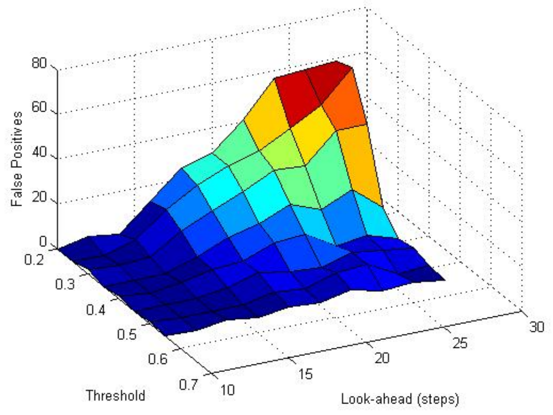

Fig. 8: Number of false positive collision predictions with different values for the look-ahead and the threshold.

threshold because the parameters that result in good performance on one metric cause poor performance according to the other. The optimal selection of parameters will simultaneously minimize both the number of collisions and the number of false positive collision predictions.

In order to characterize the effects of different parameter values on the performance of the CDF algorithm, a series of simulations were run on a range of values for both the look-ahead and the threshold. For each $(T, Threshold)$ pair, 100 simulations were run on scenario one of the T-intersection navigation problem.

Figure 7 shows how the percentage of imminent collisions avoided varied with the parameter settings. Figure 8 shows how the number of false positive imminent collision predictions varied with different parameter values. As one would expect, lower values for the threshold correspond to a greater number of collisions avoided but also correspond to a greater number of false positive predictions. The influence of the look-ahead is less obvious, as poor performance occurs at look-ahead values that are both too small or too large. For small values of the look-ahead, there is not enough time to take evasive action to prevent the collision by the time it is recognized. On the other hand, for larger values of the look-ahead, the covariance of the future state distribution grows very large making it more difficult to recognize situations where collisions are imminent.

The selection of the parameter values can be seen as a 'cautiousness/aggressiveness' setting for the autonomous vehicle. For some parameter settings (i.e. low threshold values), the vehicle will behave very cautiously, avoiding 100% of imminent collisions but also frequently braking unnecessarily when collisions are falsely predicted. Alternatively, the settings can be tuned so that the vehicle will not brake until it is nearly certain the obstacle vehicle is going to cause a collision. In this study, the collision avoidance metric was weighted more heavily and parameter values were chosen that had a reasonably low number of false positive predictions. Ultimately the look-ahead was chosen to be 1.6 seconds (16 time steps) and the threshold was set to a collision probability of 0.35.

The number of particles, M, used to approximate the current and future state distributions was chosen heuristically. As the number of particles increases, so does the accuracy of the approximation, but this improved accuracy comes at the cost of increased computational expense. Therefore, it is desirable to use as few particles as possible while still maintaining a sufficiently accurate posterior distribution. It was observed in this study that as few particles as 100 could be used to approximate the posterior distribution with acceptable convergence results.

C. Results and Analysis

Experiments were performed to compare the CDF to both a purely reactionary planner and a constant velocity planner. These two methods are commonly used in moving object tracking and collision avoidance. Reactive planners continually check to see if the planned path is still clear, if it is not, the robot will either stop or plan a new path avoiding the obstacle that is blocking the current path. Reactive planners are extremely simple but are only effective for slow moving robots using very frequent update rates. Constant velocity planners are the most common type of planner where the robot tracks moving objects' positions and velocities. Collisions are predicted by assuming the robot and the obstacle will maintain a constant velocity. The constant velocity trajectories are checked to see if a collision state will occur in the future. This technique often results in more intelligent trajectories than reactive planning, but also often results in frequent false positive collision predictions and performs poorly when the constant velocity assumption is violated.

Two experiments were performed, corresponding to the two scenarios presented in section IV-A. In both experiments, the CDF algorithm calculates the probability of a collision at a look-ahead of 1.6 seconds in the future, or 16 time steps ahead at simulation rate of 10 Hz. If the probability of a collision exceeds the threshold of 0.35, the ego vehicle then brakes at the maximum rate in an attempt to avoid the collision. The maximum rate of deceleration used in the experiments was 16 ft/s^2, which is reasonable for a passenger vehicle traveling on a road surface with a moderate coefficient of friction.

TABLE I: Simulation results using the cognitive driving framework in scenario one.

	Prediction			
Scenario	Cutoff	Yield	Right	Total
Cutoff	385	1	18	404
Yield	2	312	15	329
Right	0	0	267	267
				1000

When testing the reactive planner, this braking maneuver was initiated when any part of the obstacle vehicle entered the ego vehicle's lane. When testing the constant velocity forecaster, the braking maneuver was initiated when the planner reported a collision state at a look-ahead of 1.6 seconds, the same as used with the CDF. The constant velocity trajectories were assumed to be deterministic. Each planner was run at a rate of 10 Hz.

The first experiment performed was scenario one as described in section IV-A. In this experiment, the ego vehicle is driving north through the intersection at a speed of about 30 miles per hour (48 km/h). The obstacle vehicle is stopped at the stop sign and randomly chooses to turn right, with probability 0.25, or left, with probability 0.75. Based on the obstacle vehicle's stochastic observations, it sometimes falsely believes the intersection is clear and turns in front of the ego vehicle, causing a collision to be imminent. A collision is considered 'imminent' if the system state is such that if the ego vehicle does not take preventative measures, a collision will result. The parameter β in equation 6 was set to 0.05. This selection of β corresponds to about a 40 percent chance of the obstacle vehicle observing the ego vehicle within the first second of simulation.

Table I details the simulation results for the first experiment using the CDF. The simulation was run for a total of 1000 episodes and the obstacle vehicle cutoff the ego vehicle a total of 404 times. In this context, 'cutoff' means that if the ego vehicle were to keep its speed constant and not take preventative measures, a collision would result. The results show that the CDF was able to recognize 385 out of 404 imminent collisions and only 2 benign situations were mistaken as cutoff situations.

For comparison, the simulation was then run using the reactive planner and the constant velocity planner, again for 1000 episodes each. Table II compares the performance of all three methods. The CDF was able to avoid 95% of the imminent collisions caused by the obstacle vehicle, while the reactive planner was only able to avoid 47% of the imminent collisions. The constant velocity planner was able to prevent almost all of the imminent collisions at 98%, but at the cost of a very high false positive rate of 43%. Here, false positive means that at some point, the planner believed a collision to be imminent and initiated the emergency braking maneuver when in fact a collision was not imminent. The CDF only had a false positive rate of 9.4% and since the CDF continuously updates its online estimate of the obstacle vehicle's intent, it quickly recognizes when the obstacle vehicle is actually turning right and aborts the braking maneuver.

TABLE II: Comparison of simulation results for scenario one.

	Collisions Imminent	Collisions Occurred	False Positives	Collisions Avoided
CDF	404	22	9.4 %	94.6%
Reactive	404	213	0%	47.3%
Velocity	390	7	43.3%	98.2%

The second experiment performed was scenario two as described in section IV-A. In this experiment, the ego vehicle is driving south through the intersection, again at a speed of about 30 miles per hour, while the obstacle vehicle is heading north toward the intersection. The obstacle vehicle randomly chooses to go straight, with probability 0.25, or left, with probability 0.75. As with the first experiment, if the obstacle vehicle's intent is to turn left, it may cutoff the ego vehicle depending on it's belief. This scenario is more difficult to recognize than scenario one since the obstacle vehicle is moving at a much higher speed.

The results for the CDF algorithm on this experiment are given in table III and the comparison between all three methods is given in table IV. It can be seen that the CDF only had one false negative classification, recognizing 196 out of 197 imminent collisions before they occurred, 94% of which were able to be avoided. The reactive planner performed very poorly, only avoiding 2.5% of imminent collisions. This low collision prevention rate is due to the fact that the reactive planner is unable to detect the dangerous situation with enough time to take evasive action. The constant velocity planner also performed poorly, only preventing 8.6% of collisions. The CDF did have a significant number of false positive classifications, though, due to recognizing that the obstacle vehicle's intent is to turn left but not recognizing that the obstacle vehicle is going to yield. Similarly to scenario one, this results in initiating the braking maneuver and then aborting once the ego vehicle recognizes that the obstacle vehicle is yielding.

After testing these three different planners on two different intersection navigation scenarios, it is clear that the CDF is better at optimizing the trade off between avoiding collisions and minimizing false positive classifications (overly cautious driving) for a variety of situations. The main advantage to the reactive planner is that it has a very low false positive rate (0% for the two scenarios tested here), but it performs very poorly at preventing collisions. Both of these facts are a result of the planner not recognizing a collision as imminent until it is nearly about to occur, so the planner is very confident that the collision is indeed imminent, but there is not enough time to prevent the collision at typical driving speeds. The constant velocity planner performed well on scenario one, with a collision avoidance rate 3.6% higher than the CDF, but had nearly 5 times as many false positive classifications. Furthermore, the constant velocity planner performed very poorly on scenario two, only preventing 8.6% of imminent collisions. This is caused by the planner's failure to anticipate the driver's turning action due to the constant velocity assumption.

The cognitive driving framework, on the other hand, was able to prevent about 94% of collisions in both scenarios. At the same time, the CDF had a fairly low false positive

TABLE III: Simulation results using the cognitive driving framework in scenario two.

	Prediction			
Scenario	Cutoff	Yield	Right	Total
Cutoff	196	0	1	197
Yield	273	279	5	557
Right	1	0	245	246
				1000

TABLE IV: Comparison of simulation results for scenario two.

	Collisions Imminent	Collisions Occurred	False Positives	Collisions Avoided
CDF	197	12	34.1%	93.9%
Reactive	200	195	0%	2.5%
Velocity	234	214	8.0%	8.6%

prediction rate with only 9.4% false positive predictions in scenario one and 34% in scenario two. As discussed in section IV-B, by adjusting the parameters in the CDF algorithm, the 'cautiousness' of the autonomous vehicle can be tuned to users' preferences. Additionally, in future work, a fuzzy logic controller can be used to determine when and how much the ego vehicle should brake in order to better optimize collision avoidance and user comfort.

V. Conclusion and Future Work

This paper presented the cognitive driving framework, a method for joint inference of the intent and belief of an obstacle vehicle in an intersection navigation scenario. The goal of the CDF is to allow an autonomous vehicle to predict when a potentially hazardous situation is about to occur early enough to allow the autonomous vehicle to take evasive action to prevent a collision. The formulation of the problem as a dynamic Bayesian network was presented. A non-linear filtering method was proposed using a particle filter to estimate the posterior distribution of the state of the DBN. Monte Carlo simulation is used to estimate the future sate distribution and calculate the probability of a collision. Finally, the accuracy of the estimation method was demonstrated by simulating two intersection navigation scenarios where an obstacle vehicle cuts off the autonomous vehicle. The simulation results show that the CDF is able to predict and prevent 94% of imminent collisions in two different intersection navigation scenarios. For comparison, the same simulations were run using a purely reactionary planner and a constant velocity planner. The results show that the reactive planner prevented only 47% and 3% of imminent collisions on the two scenarios. The constant velocity planner performed well on the first scenario, preventing 98% of collisions, but only prevented 9% of collisions in the second scenario.

This work has some natural extensions that should be explored. Adding additional vehicles to the intersection environment would lead to some interesting challenges that the CDF should be evaluated on, such as how the algorithm handles occlusions and how computation time scales with the number of vehicles. The framework could be strengthened by relaxing the assumption that the vehicle poses are known. Additionally, the authors are implementing the cognitive driving framework on an autonomous vehicle platform in order to test the algorithm in an actual intersection navigation scenario.

References

[1] R. Bishop, "Intelligent vehicle applications worldwide," *Intelligent Systems and their Applications, IEEE*, vol. 15, no. 1, pp. 78–81, 2000.

[2] T. C. Folsom, "Social ramifications of autonomous urban land vehicles," in *IEEE International Symposium on Technology and Society*, 2011.

[3] Z. Juan, J. Wu, and M. McDonald, "Socio-economic impact assessment of intelligent transport systems," *Tsinghua Science & Technology*, vol. 11, no. 3, pp. 339–350, 2006.

[4] J. Alonso-Mora, A. Breitenmoser, P. Beardsley, and R. Siegwart, "Reciprocal collision avoidance for multiple car-like robots," in *Robotics and Automation (ICRA), 2012 IEEE International Conference on*. IEEE, 2012, pp. 360–366.

[5] B. Kluge and E. Prassler, "Recursive agent modeling with probabilistic velocity obstacles for mobile robot navigation among humans," in *Autonomous Navigation in Dynamic Environments*. Springer, 2007, pp. 121–134.

[6] I. Miller, M. Campbell, D. Huttenlocher, F.-R. Kline, A. Nathan, S. Lupashin, J. Catlin, B. Schimpf, P. Moran, N. Zych *et al.*, "Team cornell's skynet: Robust perception and planning in an urban environment," *Journal of Field Robotics*, vol. 25, no. 8, pp. 493–527, 2008.

[7] D. Vasquez, T. Fraichard, and C. Laugier, "Incremental learning of statistical motion patterns with growing hidden markov models," *Intelligent Transportation Systems, IEEE Transactions on*, vol. 10, no. 3, pp. 403–416, 2009.

[8] R. Kelley, A. Tavakkoli, C. King, M. Nicolescu, M. Nicolescu, and G. Bebis, "Understanding human intentions via hidden markov models in autonomous mobile robots," in *Proceedings of the 3rd ACM/IEEE international conference on Human robot interaction*. ACM, 2008, pp. 367–374.

[9] T. Bandyopadhyay, C. Z. Jie, D. Hsu, M. H. Ang Jr, D. Rus, and E. Frazzoli, "Intention-aware pedestrian avoidance," in *Experimental Robotics*. Springer, 2013, pp. 963–977.

[10] F. Havlak and M. Campbell, "Discrete and continuous, probabilistic anticipation for autonomous robots in urban environments," *Robotics, IEEE Transactions on*, vol. 30, no. 2, pp. 461–474, 2014.

[11] D. Ellis, E. Sommerlade, and I. Reid, "Modelling pedestrian trajectory patterns with gaussian processes," in *Computer Vision Workshops (ICCV Workshops), 2009 IEEE 12th International Conference on*. IEEE, 2009, pp. 1229–1234.

[12] C. L. Baker, R. R. Saxe, and J. B. Tenenbaum, "Bayesian theory of mind: Modeling joint belief-desire attribution," in *Proceedings of the thirty-second annual conference of the cognitive science society*, 2011, pp. 2469–2474.

[13] U. Lerner, R. Parr, D. Koller, G. Biswas *et al.*, "Bayesian fault detection and diagnosis in dynamic systems," in *AAAI/IAAI*, 2000, pp. 531–537.

[14] M. S. Arulampalam, S. Maskell, N. Gordon, and T. Clapp, "A tutorial on particle filters for online nonlinear/non-gaussian bayesian tracking," *Signal Processing, IEEE Transactions on*, vol. 50, no. 2, pp. 174–188, 2002.

[15] G. Kitagawa, "Monte carlo filter and smoother for non-gaussian nonlinear state space models," *Journal of computational and graphical statistics*, vol. 5, no. 1, pp. 1–25, 1996.