Submitted to the Integrated Computer-Aided Engineering Journal.

Towards a Tool for Performance Evaluation of Autonomous Vehicle Navigation in Dynamic, On-Road Environments

Craig Schlenoff[1], Jerome Ajot, and Raj Madhavan

National Institute of Standards and Technology (NIST)

100 Bureau Drive, Stop 8230

Gaithersburg, MD 20899

Phone: 301-975-3456, Fax: 301-990-9688

Email: craig.schlenoff@nist.gov, jerome.ajot@nist.gov, raj.madhavan@ieee.org

ABSTRACT

We are developing a novel framework, PRIDE (PRediction In Dynamic Environments), to perform moving object prediction (MOP) to assist unmanned ground vehicles in performing path planning within dynamic environments. In addition to predicting the location of moving objects in the environment, we have extended PRIDE to generate simulated traffic during on-road driving. In this paper, we explore applying the PRIDE-based traffic control algorithms for the performance evaluation of autonomous vehicles. Through the use of repeatable and realistic traffic simulation, one is able to evaluate the performance of an autonomous vehicle in an on-road driving scenario without the risk involved with introducing the vehicle into a potentially dangerous roadway situation. In addition, by varying a single vehicle's parameters (such as aggressivity), we can show how the entire traffic pattern is affected. We will describe the successes that have been achieved to date in a simulated environment, as well as enhancements that are currently being researched and expected in the near future.

Keywords: PRIDE, performance metrics, moving object prediction, traffic simulation

[1] Corresponding author

1. INTRODUCTION

The field of autonomous navigation systems is continuing to gain traction both with researchers and practitioners. Funding for research is this area has continued to grow over the past few years, and recent high profile funding opportunities have started to push theoretical research efforts into practical use, as evidenced by the interest spurred by the DARPA Grand Challenge [1]. Autonomous navigation systems in this context refer to embodied intelligent systems that can navigate fairly independently without human supervision.

For an autonomous vehicle system to operate in static and dynamic environments, the planner is of paramount importance. Local path planning methods typically utilize cost-based planners and have found their widespread use in dynamic environments. Traditionally, these planners decompose the world into discrete, evenly spaced locations and then assign costs for the occupancy of these locations and for transitioning from one location to another. These costs are often calculated assuming that the objects in the environment are static, since limited information tends to be available about the future location of unknown dynamic objects. To successfully navigate and accomplish missions in dynamic environments, this assumption is overly restrictive; information about dynamic objects should be made available to the planner.

Many believe that the DEMO III eXperimental Unmanned Vehicle (XUV) effort represents the state-of-the-art in autonomous off-road driving [25]. This effort developed and demonstrated new autonomous vehicle technology emphasizing perception, navigation, intelligent system architecture, and planning. It should be noted the DEMO III XUV has only been tested in static, closed-course environments. It has not been tested in on-road driving situations, which include pedestrians and oncoming traffic. There have been experiments performed with autonomous vehicles during on-road navigation. Perhaps the most successful has been that of Dickmanns [8] as part of the European Prometheus project in which an autonomous vehicle performed a trip from Munich to Odense in Germany (over 1,600 km) at a maximum speed of 180 km/hr. Although the vehicle was able to identify and track other moving vehicles in the environment, it could only make basic predictions of where those vehicles were expected to be at points in the future, considering the external vehicles' current velocity and acceleration.

What is missing from all of these experiments is a level of situation awareness of how other vehicles in the environment are expected to behave considering the situation in which they find themselves. When humans drive, we often have expectations of how each object in the environment will move based upon the situation. For example, when a vehicle is approaching an object that is stopped in the road, we expect it to slow down and stop behind the object or try to pass it. When we see a vehicle with its blinker on, we expect it to turn or change lanes. When we see a vehicle traveling behind another vehicle at a constant speed, we expect it to continue traveling at that speed. The decisions that we make in our vehicle are largely a function of the assumptions we make about the behavior of other vehicles.

To date, the authors are not aware of any autonomous vehicle efforts that account for this information when performing path planning. To address this need, we have developed a framework, called PRIDE (**PR**ediction **I**n **D**ynamic **E**nvironments) that provides an autonomous vehicle's planning system with information that it needs to perform path planning in the presence of moving objects [22]. Even though the theoretical foundations of the framework are in place, the underlying implementational details are a subject of on-going research. In this paper, we describe how we leveraged the algorithms in the PRIDE framework to simulate traffic during on-road driving. The primary contributions of this paper are the discussion of the PRIDE framework, the detailing of the algorithms that contribute to the moving object predictions, and an exposition of how this approach could be applied to assigning performance metrics to autonomous vehicles. It should be noted that this effort is not addressing the issue of detecting moving objects. This is a very difficult problem and one that is being addressed by other efforts. In this effort, we are taking the identification of moving objects as an input, and inferring their possible future location based upon contextual information about the object and the environment.

The paper is structured as follows: In Section 2, we survey related work in moving object prediction (MOP) and traffic simulation areas. In Section 3, we describe the PRIDE framework and show how we apply it to moving object prediction. In Section 4, we explain how we use the PRIDE framework algorithms for traffic simulation. In Section 5, we show how the traffic simulation can be used to associate performance metrics to autonomous vehicles. Section 6 provides experimental results and Section 7 concludes the paper with suggestions for further research.

2. RELATED WORK

Because moving object *detection* is such a difficult problem, there has been relatively little research in moving object *prediction*. Firby [13] uses NaTs (navigation templates) as a symbolic representation of static and dynamic sensed obstacles to drive a robot's motors to respond quickly to moving objects. Gueting [14] extends database structures to allow for the representation of dynamic attributes (i.e., ones that change over time) and also extends the database's query language to allow for easier querying of the values of dynamic attributes. Singhal [26] introduces the concept of dynamic occupancy grids which allows each cell to have a state vector which contains information such as a probabilistic estimate of the entity's identity, location, and characteristics (such as velocity, acceleration) along with global probability distribution functions. Nagel [15] has performed some research on moving object prediction during on-road driving based upon the concept of generally describable situations, fuzzy logic, and situation graph trees. However, based on the literature, Nagel has not tried to project out what the next actions of the moving object will be and has not assigned probabilities to those actions. Dickmanns [9] has performed research on situation assessment and intention recognition for on-road driving using a Dynamic Objects Database. RRTs (Rapidly-exploring Random Trees) is a popular approach for path planning problems that involve obstacles. They have been applied to a number of areas including collision-free control of virtual humans [17] and Mars exploration vehicles [27]. However, this approach does not take into account situation recognition.

There have also been agent architectures that have been created to address, among other things, dealing with moving objects. Ferguson [12] developed an AI software architecture suitable for controlling and coordinating the actions of a *rational*, *autonomous*, resource-bounded agent embedded in a partially structured, dynamic, multi-agent world. Albus [4] created the RCS (Real-time Control System) architecture which supports the specification and inter-communication of agents which could be applied in dynamic environments. RCS has been used successfully in a number of domains, including controlling machine tools, post office package handling, and autonomous vehicles [2].

Statistical methods for estimating obstacle locations using statistical features have been proposed by other researchers such as the Hidden Markov Models (HMMs) to predict obstacle motion [28], Poisson distribution to describe the probability of collision with obstacles [24], autoregressive models for one-step ahead prediction of moving obstacles [10] or probability of occupancy of cells in grid maps [20]. The principal disadvantages of these methods are that they are computationally intensive thus precluding real-time implementations. They have also only been implemented for 2D polygonal indoor environments.

In addition to moving object prediction, there are few distinctive techniques for planning around moving objects. In [6], an implementation of genetic algorithms called Ariadne's Clew is used to perform path planning. Inspired by Probabilistic-Roadmap Methods (PRM), the planner described in [19] samples the state and time space of a robot by picking control inputs at random in order to compute a roadmap that captures the connectivity of the space. Balakirsky [5] uses a combined logic-based and cost-based planning approach in order to allow for the creation of logic-constrained, cost-optimal plans with respect to dynamic environments, user objectives, and constraints. Ratering and Gini [21] use hybrid potential field, to navigate a robot in situations in which the environment is known except for unknown and possibly moving obstacles. Kindel [19] developed a randomized motion planner for robots that must achieve a specified goal under kinematic and/or dynamic motion constraints while avoiding collision with moving obstacles with known trajectories.

As for traffic simulation, most of the work in the literature dealing with drivers' actions and predicted behavior has been performed by psychologists in an attempt to explain drivers' behaviors and to identify the reason for certain disfunctions. There have been a few efforts that have tried to simulate traffic patterns.

One of the more prominent ones in the literature is ARCHISM [7,11], but even this effort is based upon driving psychology studies. These traffic simulations use laws that can be applied for a specific environment or a specific situation. Some of those postulates can be expanded to generic situations. Additional work preformed at the Sharif University of Technology of Tehran [16] is based on two assumptions: the maximum speed on a road segment and the potential danger that could be incurred by traversing this road segment. These assumptions and concepts allow the simulation program to create a rank of actions the vehicle can execute based on the danger that could be incurred.

The work described in this paper is different in that it introduces a novel way to perform moving object prediction based upon a multi-resolutional, hierarchical approach which incorporates multiple prediction algorithms into a single, unifying framework. The lower levels of the framework utilize estimation theoretic short-term predictions based upon an extended Kalman filter while the upper levels utilize a probabilistic prediction approach based upon situation recognition with an underlying cost model.

3. THE PRIDE FRAMEWORK

3.1. Overview

We are using the 4D/RCS (Real-Time Control System) reference model architecture [3] as the basis in which to apply the representational approaches that are being developed in this effort. 4D/RCS was chosen due to its explicit and well-defined world modeling capabilities and interfaces, as well as its multi-resolution, hierarchical planning approach. Specifically, 4D/RCS allows for planning at multiple levels of abstraction, using different planning approaches as well as supporting inherently different world model representations. By applying this architecture, we can ensure that the representations being developed for moving objects can accommodate different types of planners that have different representational requirements.

The RCS architecture supports multiple behavior generation (BG) systems working cooperatively to compute a final plan for the autonomous system. At each successively lower level in the hierarchy, the spatial and temporal resolution of the individual BG systems increases while the amount of time allowed for each BG system to compute a solution decreases. In addition to multiple BG systems, multiple world models are supported with each world model's content being tailored to the systems that it supports (in this case the different BG systems). At each successively lower level in the hierarchy, the resolution of the world model increases to support the needs of the BG. To align with the hierarchy present within 4D/RCS, it is necessary for moving objects to be represented differently at the different levels of the architecture.

To support this requirement, NIST has developed the PRIDE (PRediction In Dynamic Environments) framework. This framework supports the prediction of the future location of moving objects at various levels of resolution, thus providing prediction information at the frequency and level of abstraction necessary for planners at different levels within the hierarchy. To date, two prediction approaches have been applied to this framework.

3.2 Prediction Algorithms

3.2.1. Lower-level Prediction Algorithms

At the lowers levels, we utilize estimation theoretic short-term predictions via an extended Kalman filter-based algorithm using sensor data to predict the future location of moving objects with an associated confidence measure. Estimation-theoretic schemes using Kalman Filters (KFs) are well established recursive state estimation techniques where state estimates of a system are computed using the process and observation models [18]. The recursive nature of the algorithm utilizes the system's CPU more uniformly to provide estimates without the latency resulting from batch processing techniques. The (linear) KF is simply a recursive estimation algorithm that provides minimum mean squared estimates of the states of a linear system utilizing knowledge about the process and measurement dynamics, process and measurement noise statistics subject to Gaussian assumptions and initial condition information. When these assumptions are satisfied, the estimates provided by the Kalman filter are *optimal*. The extension of the linear Kalman filtering ideas to a nonlinear system is termed extended Kalman filtering.

The strength of using an EKF is that it provides a covariance matrix that is indicative of the uncertainty in the prediction. An EKF employs a process model to estimate the future location of the object of interest. Since the object classification module provides the type of moving object whose position and orientation needs to be predicted, we have envisaged a bank of EKFs for each type of classified object. In turn, this has the added advantage of cross-corroborating the object classification itself as the uncertainty in the EKF prediction will be an indicator of the quality of the prediction. The higher the uncertainty, the lower the confidence in the selection of the correct set of object models and thus consequently decreasing the confidence of the object classification. Thus, our approach combines low-level (image segmentation/classification) and mid-level (recursive trajectory estimation) information to obtain the short-term prediction and combines it with the cross-corroboration to work symbiotically to effectively reduce the total uncertainty in predicting the positions and orientations of moving objects.

For short-term predictions, it was found that a separate EKF is necessary for different types of moving objects as opposed to a separate EKF for each individual moving object. In essence, a separate prediction equation is needed when the dynamics of the moving object significantly change. For example, multiple variations of tracked tanks could all use the same prediction equations since the kinematics of these tanks do not differ significantly. However, these equations could not be used for wheeled vehicles. Additionally, a generalized prediction equation was sufficient for near-term planning until the moving object could be classified.

3.2.2. High-level Prediction Algorithms

At the higher levels of the framework, moving object prediction needs to occur at a much lower frequency, while allowing for a greater level of inaccuracy. At these levels, moving objects are identified as far as the sensors can detect, and a determination is made as to which objects should be classified as "objects of interest”. In this context, an object of interest is an object that has a possibility of affecting our path in the time horizon in which we are planning. At this level, we use a moving object prediction approach based on situation recognition and probabilistic prediction algorithms to predict where we expect that object to be at various time steps into the future. Situation recognition is performed using spatio-temporal reasoning and pattern matching with an *a priori* database of situations that are expected to be seen in the environment. In these algorithms, we are typically looking at planning horizons on the order of tens of seconds into the future with one second plan steps. At this level, we are not looking to predict the exact location of the moving object. Instead, we are attempting to characterize the types of actions we expect the moving object to take and the approximate location the moving object would be in if it took that action. More information about this approach is included in the follow sections.

3.3. How the Predictions Are Used by the Planner

The original purpose for the development of the PRIDE framework was to inform a planner about the probable location of moving objects in the environment so that the planner could make appropriate plans in dynamic environments. Although this is not how the Moving Object Prediction algorithms are being used in this paper, the authors feel it is important to show the original intent so as to better inform the reader of how it is being adapted to traffic simulation. As such, in this section, we will describe how a planner uses the outputs of the MOP algorithms. Before we do, we need to describe the algorithms’ expected output.

Each time the algorithms are executed, information, such as what is shown in each line of Table 1, is provided for each possible future location of every pertinent moving object in the environment. The MOP output is composed of a list of time steps in the future, external vehicle information (ID and type of the vehicle), all the possible future locations (XPosition, YPosition), and probability information. Every predicted location has an associated probability to represent the probability that the vehicle will be at that location at a certain time in the future. Some of these predicted positions are not relevant due to a low probability, so a threshold can be applied to ignore those locations under the threshold value.

It is expected that a planner will use the probability information from the MOP to determine the damage potential of occupying a location in space at a given time. Specifically, this damage potential will be based on the object it will encounter and the probability that the object will be there. For example, if the MOP algorithms determine that a HMMWV (High-Mobility Multipurpose Wheeled Vehicle), which we assume has a maximum damage potential of 200, has a 40 % chance of occupying a point in space, then the planner may associate a damage potential (due to the presence of moving objects) of 80 (40 % of 200) when determining the most optimal path. With this information, a planner can produce appropriate plans in the presence of a dynamic environment.

4. APPLYING THE PRIDE FRAMEWORK TO TRAFFIC SIMULATION

Although the PRIDE framework was originally developed to inform a planner about the future position of moving objects for the purpose of path planning and collision avoidance, we have found that the same set of algorithms could be applied to simulating traffic patterns during on-road driving. More specifically, we applied the situation recognition and probabilistic algorithms to determine the likely actions that a vehicle in the environment would take when confronted with a specific situation, and then command that vehicle to perform that action. By doing this with multiple vehicles, we are able to simulate fairly sophisticated traffic situations in which vehicles behave in a way that is very similar to how a human would behave. Vehicles will slow down and/or pass when approaching a stopped or slow object in their lane, they will typically only change lanes when the next lane is clear and they are going slower than desired, they will keep a safe following distance, etc. By providing realistic simulations of traffic situations, we are able to test the autonomous vehicle during realistic on-road driving situations, without having to place the vehicle on a potentially dangerous city street or highway.

Throughout the remainder of the paper, we will refer to two types of vehicles. The first is traffic vehicles that we are trying to predict the future location of. We refer to these as TV (traffic vehicles). The second is the autonomous vehicle that perceives these vehicles. We refer to this vehicle as AV (autonomous vehicle).

Whenever the term ‘vehicle’ is used throughout the remainder of the paper, either TV or AV will follow it to make clear which vehicle to which we are referring.

We are initially using these algorithms in simulated environments (such as the OneSaf and AutoSim[2] simulation environments) to test the planning algorithms in the presence of moving objects. Then, when the NIST-developed Autonomous Road Driving Arenas [23] are completed, these algorithms will be used to control “environmental” vehicles (TV) in the arena to simulate on-road traffic.

4.1. Scenario

The algorithms described in this section are used to generate realistic traffic patterns in the environment. These traffic patterns are generated at planning horizons on the order of tens of seconds with one-second plan steps. During the explanation of the algorithm, the following scenario will be used (Figure 1). This scenario is composed of three vehicles (TV), two of which (A and B) are in lane L1 and moving to the right, and the third (C) is in lane L2 and moving to the left. In this scenario, D is a static object and is located in L1.

4.2. Implementation details

In this section, we will describe, in detail, the moving object prediction (MOP) algorithms. Figure 2 graphically shows the overall process flow.

The steps within the algorithm are:

1. For each vehicle (TV) on the road (α), the algorithm gets the current position and velocity of the vehicle (TV) by querying external programs/sensors (β).
2. For each set of possible future actions (δ) (explained in Section 4.3.), the algorithm creates a set of next possible positions and assigns an overall cost to each action based upon the cost incurred by

[2] Certain commercial software and tools are identified in this paper in order to explain our research. Such identification does not imply recommendation or endorsement by the National Institute of Standards and Technology, nor does it imply that the software tools identified are necessarily the best available for the purpose.

performing the action and the cost incurred based upon the vehicle's (TV) proximity to static objects (explained in Section 4.4.1).

3. Based upon the costs determined in Step 2, the algorithm computes the probability for each action the vehicle (TV) may perform (ε) (explained in Section 4.4.2). At this step in the scenario (Figure 1) the possible actions/probabilities for the three vehicles (TV) are shown in Figures 3A-C. The location of each dot represents the possible location that vehicle can reach in the predefined timeframe and the size of each dot represents the relative probability of the vehicle occupying that location with respect to the other locations.
4. Predicted Vehicle Trajectories (PVT) (ξ) are built for each vehicle (TV) which will be used to evaluate the possibility of collision with other vehicles (TV) in the environment. PVTs are vectors that indicate the possible paths that a vehicle (TV) will take within a predetermined number of time steps into the future. The Predicted Vehicle Trajectory notion is explained in more detail in Section 4.5.
5. For each pair of PVTs (η), the algorithm checks if a possible collision will occur (where PVTs intersect) and assigns a cost if collision is expected. In the scenario, for the vehicles A and C, Figure 4 shows two PVTs that cross, indicating that a collision is possible. This is explained further in Section 4.5.
6. In this step, the probabilities of the individual actions (θ) are recalculated, incorporating the risk of collision with other moving objects that was calculated in Step 5. The resultant probabilities are shown in Figures 5A-C. Similar to Figures 3A-C, the location of each dot represents the possible location that the vehicle can reach in the predefined timeframe and the size of each dot represents the relative probability of the vehicle occupying that location with respect to the other locations.

The output of this loop is a list of locations with associated probabilities showing where a vehicle (TV) is expected to be at specific times in the future (refer to Table 1). Using these probabilities, we can create traffic patterns in one of two ways:

1. Control the vehicle (TV) to move to the location with the highest probability. For example, if the vehicle has a 40 % chance of being at location A, a 30 % chance of being at location B, a

20 % chance of being at location C, and a 10% chance of being at location D, the vehicle (TV) will always be commanded to move to location A.

2. Control the vehicle (TV) to move to a location whose likelihood is proportional to the probability that it is expected to be there. In the example above, one approach would be to use a random number generator. In this way, a vehicle's (TV) movement would be closely tied to the probabilities coming out of the moving object predictor, as opposed to always moving to the location with the highest probability.

Independent of the approach used to control the vehicles (TV), the output of these algorithms result in realistic traffic patterns involving one to many vehicles (TV) that can be used a basis to evaluate the performance of autonomous vehicle (AV) within simulated on-road driving scenarios.

4.3. Actions

For the purpose of this work, actions are a discretized set of basic behaviors that a driver may perform during on-road driving. To represent the process of predicting several time steps into the future, a series of continuous actions (action sequences) are created *a priori* that are consistent with a set of preset rules. These rules will be explained in Section 4.3.2.

4.3.1. Elementary actions

On the straight road, a vehicle (TV) can execute two types of actions. The first type of action pertains to its acceleration profile. The possible values of this type are: Quick Acceleration (QA), Slow Acceleration (SA), Keep the same Speed (KS), Slow Deceleration (SD), and Quick Deceleration (QD). The second type of action pertains to the changing lane process. The vehicle (TV) can stay in the same lane or change lanes to the right or the left. So there are: Change to the Left lane (CL), stay in the Same Lane (SL), and Change to the Right lane (CR). Note that at this time we had only dealt with continuous road segments. At a later date we will be addressing navigation through intersections.

Thus a vehicle (TV) can have up to fifteen possible actions (Figure 6). Every possible elementary action is assumed to be able to be accomplished in one time step. Some actions may not be possible due to the vehicle's (TV) current velocity (for example, a vehicle (TV) moving very slowly cannot change lanes in one second during a deceleration) or location (if a vehicle (TV) is in the rightmost lane, it can not change lanes to the right). In this case, those actions are not considered. Referring back to Figure 1, all vehicles can execute any acceleration profile, but all vehicles (A, B, and C) cannot change to the right lane (since they are all on the right-most lane).

4.3.2. Action Sequences

One action is assumed to be performed in one time step. Thus when predicting *n* time steps in the future, *n* actions will be completed. This set of *n* actions is referred to as an action sequence. An example of an action sequence may be a passing maneuver, which may include the steps of accelerating in your current lane, changing lanes to the left while continuing to accelerate, changing lanes to the right, and then decelerating. These action sequences are made up of, among other things, acceleration profiles (QA, SA, KS, SD, and QD) described in Section 4.3.1. Assuming that every time step is set to one second (note that this is not a fixed value, it can be set by the user), the action sequence has to be realistic. As such, there are some action sequences that are improbable, and thus eliminated from consideration. This elimination procedure is performed using rules, as described below.

For now, a single rule is applied to all of the possible action sequences to generate the most realistic ones. To evaluate these rules, we associate a value to each "acceleration profile": 2 for QA, 1 for SA, 0 for KS, -1 for SD, and -2 for QD. The rule states that a vehicle (TV) can only switch from an action to another action if their values differ by one. It should be noted that the inclusion of this rule prohibits "emergency stops," in which the acceleration change of the vehicle could change by more than one unit within one time step. Emergency stops and the addition of other rules will be addressed as the research progresses. Example of action sequences and their associated validity is shown in Table 2.

Again referring to Figure 1, an appropriate action sequence for Vehicle A may be a passing maneuver (if its desired speed is greater than the speed of Vehicle B), Vehicle B would also need to perform a passing maneuver to get around the obstacle, while Vehicle C would most likely perform an action sequence that would involve constant forward velocity since there are currently no objects to impede its path.

4.4. Cost Model and Probability

4.4.1. Cost Model

All of the probabilities in the Moving Object Prediction (MOP) algorithms are based upon an underlying cost model. The Cost Model (CM) simulates the danger that a driver would incur by performing an action or occupying a state. These costs are being used by multiple efforts within the program that this effort is a part of. Thus, there is value of building the probabilities directly from these costs to allow for synergy with other efforts. These costs can be separated into two different categories.

1. **Cost representing the vehicle's (TV) actions:** This cost represents the penalties for performing an action as a function of the amount of attention needed. For example, the changing lane action needs more concentration than going straight in the same lane, thus the cost for changing lanes is greater. In the same vein, a slow deceleration needs less attention than a fast deceleration, thus the slow deceleration has a lower cost.
2. **Cost representing possible collisions on the road:** This includes collisions with static and moving objects. Examples of static objects on the road are roadblock and barrier. Examples of dynamic objects on the road are other vehicles. The costs associated with static or moving objects is proportional to the danger and imminence of collision. For example, a road block at one kilometer ahead is less dangerous than another vehicle passing at three meters ahead. In Figure 1, Vehicle B represents a possible collision obstacle to Vehicle A and would therefore have a cost associated with being to close to it. Similarly, Obstacle D would have a cost associated with it for Vehicle B.

Examples of costs are shown in Table 3.

4.4.2. Converting Costs to Probabilities

Based on the costs associated with a given action, we can determine the probability that the vehicle (TV) will perform that action in the following way.

The first step is to create a ratio of the cost for performing a given action to the sum of all of the costs for performing all possible actions. This is shown in Equation 1. Note that the equation is inverted since the probability of an action is inversely proportional to the cost of performing that action.

$$ratio = \frac{\sum_j \cos t_j}{\cos t_i} \qquad (1)$$

We then normalize the ratio by dividing it by the sum of all of the ratios, as shown in Equation 2.

$$\Pr ob_i = \frac{ratio_i}{\sum_j ratio_j} \qquad (2)$$

Equation 2 (*Prob*$_i$) computes the normalized probability of a given action occurring as compared to all other actions that are possible at that time.

4.4.3. Vehicle Aggressivity

Unlike other approaches that use an underlying static cost model for activities such as path planning, this approach introduces the concept of a dynamic cost model, where the costs are vehicle (TV) specific and are a function of what is perceived in the environment. As explained above, we associate underlying costs with various actions and states. We then sum the costs that are associated with a specific driving maneuver and use that overall cost to determine the probability that a vehicle (TV) will perform that maneuver; the higher the cost to perform the maneuver, the lower the probability that it will occur. But different types of drivers have different underlying costs model that affect their behavior. A very aggressive driver will have a lower cost for changing lanes, changing his/her velocity quickly, following another vehicle too closely, not driving in the right-most lane, etc. Conversely, a very conservative driver will have much higher costs for these actions.

In our approach, we have set up two aggressivity profiles: aggressive and passive. Since we are controlling the traffic, we pre-specify the aggressivity of each driver on the road before the system starts to ensure that the vehicle (TV) behaves in a way that is in line with its personality. The aggressivity of the vehicle (TV) can be modified as the algorithms are run to cause changing behaviors.

When these algorithms are used for moving object prediction (i.e., when we are observing vehicles (TV) that we cannot control), the aggressivity of the driver can be "learned" based on observations (e.g., how many times the vehicle (TV) changes lanes, its average speed compared to the speed limit, following distance from other vehicles, etc.). Based on this information, an aggressivity profile can be assigned to (or changed for) the vehicle (TV) as the algorithms run. The initial aggressivity of other vehicles (TV) on the road can be set to either aggressive or passive. Determining aggressivity of other vehicles (TV) is a research area that is only conceptual at this point and will be further explored as the research progresses.

Although we currently have only two aggressivity profiles, (aggressive and passive), we foresee more of a continuous scale in the future, where the costs of the actions in the profiles would be closely tied to the specific actions that were perceived of other vehicles (TV). For example, if we perceive that a vehicle (TV) has a tendency to follow the vehicle in front of it very closely but does not change lanes very often, we will increase the cost of changing lanes (thus decreasing the probability that it will occur) but decrease the cost associated with following distance for that vehicle (TV).

4.5. Predicted Vehicle Trajectory

A Predicted Vehicle Trajectory (PVT) represents the possible movements of a vehicle (TV) throughout the time period being analyzed. The PVT is represented by a trajectory. Although current efforts do not account for the dimensions of the vehicle (TV) (i.e., we treat the vehicle (TV) as a point in space as opposed to a solid object), the PVT concept could easily be extended to account for the width and length of the vehicle (TV). This has not been a point of emphasis thus far in the work since the "buffer zone" surrounding the vehicle (TV) (the area around the vehicle in which a cost is incurred if the controlled vehicle (TV) enters it) subsumes the dimensions of the vehicle (TV). This will be the topic of future research.

The PVT is built from the origin position (x_{IP}, y_{IP}, $t_{IP}=0$) at time=0 to the predicted position (x_{PP}, y_{PP}, $t_{PP}=t_{Pred}$) where t_{Pred} is the predetermined time in the future for the prediction process. Also contained within the PVT is the action-cost and action-probability information.

The PVT is used to determine if potential collision will occur, as shown in Figure 4. Because a PVT represents a trajectory of one predicted position (initial to predicted), to obtain the collision information between two vehicles (TV), the possible intersection between two PVT has to be checked.

Thus, no collision occurs when two PVTs are not crossing. But when two PVTs do cross, there is a probable collision (Figure 7). When two PVTs cross, it is important to know when (where) they cross. This information can be obtained by using a parametrization of each PVT.

The parametrization is:

$$\begin{aligned} x_1(u_1) &= x_{PP1}\; u_1 + x_{IP1}\;(1-u_1) \\ y_1(u_1) &= y_{PP1}\; u_1 + y_{IP1}\;(1-u_1) \end{aligned} \qquad \text{where } u_1 \in [0, 1] \tag{3}$$

$$x_2(u_2) = x_{PP2}\; u_2 + x_{IP2}\;(1-u_2) \qquad \text{where } u_2 \in [0, 1] \tag{4}$$

$$y_2(u_2) = y_{PP2}\ u_2 + y_{IP2}\ (1-u_2)$$

where u_1 and u_2 are the parameters of each PVT.

The two equations (3 and (4) create a linear system which, after using the Cramer's Theorem, can be used to determine u_1 and u_2:

So the two vehicles (TV) will cross each other at two different times ($u_1\ t_{Pred}$) for the first vehicle (TV) and ($u_2\ t_{Pred}$) for the second vehicle (TV). For a small difference, the collision is probable or certain. Conversely, for a large difference, the collision is improbable. Thus if the PVTs cross and the difference of time is less than a predetermined time (T), we use Equation 6 to determine the collision cost:

$$\text{CollisionCost} = CO\left(T - \left(t_{Pred}\ |u_1-u_2|\right)\right) \qquad (6)$$

where CO is the predetermined maximum cost that can occur when colliding with a specific object (Table 2) and T is the predetermined time difference in which a cost for collision will be incurred.

5. APPLYING TRAFFIC SIMULATION TO PERFORMANCE METRICS

Now that we've described how we can simulate traffic patterns, we will discuss how this could be used to associate performance metrics to an autonomous vehicle (AV). In evaluating how an autonomous vehicle (AV) is performing during on-road driving, one needs the ability to test that vehicle (AV) in various driving situations. Those situations are a function of the environment (e.g., winding roads, steep slopes, traffic signals, intersections), weather conditions (e.g., rain, fog, ice on the roadway), and static and dynamic objects in the environment (e.g., traffic barrels, pedestrians, other vehicles (TV)). The traffic simulator allows us the ability to dynamically change information about static and dynamic objects in the environment in order to introduce a variety of situations against which we can evaluate the autonomous vehicle (AV).

The traffic simulator allows one to have repeatable, realistic traffic patterns. As such, it is able to place two different autonomous vehicles (AV) in an identical traffic environment to evaluate how each performs. If the two autonomous vehicles (AV) behaved in identical fashion, the entire flow of traffic would be identical. Conversely, if the two autonomous vehicles' (AV) behaviors differed in any way, the flow of traffic would most likely differ (since other vehicles (TV) in the traffic pattern may be reacting to the actions of the autonomous vehicle (AV)). In using the traffic simulator to assign performance metrics, levels of difficulty are associated with different traffic patterns, based on the number of vehicles (TV) in the traffic, location of the vehicles (TV) on the road, types of vehicles (TV) on the road (e.g., buses, taxis, police cars), and the aggressivity of other vehicles (TV). The traffic simulator allows one to create situations where an accident among a pair of traffic vehicles (TV) is imminent. The autonomous vehicle (AV) is then evaluated based upon its ability to predict this accident and take precautions in time.

Metrics are assigned to the autonomous vehicle's (AV) performance based on a number of criteria, including proximity to other vehicles (TV) (within buffer distance), staying within the speed limit, number of major velocity and acceleration changes, number of lane changes, obeying traffic signs and signals, etc.

6. EXPERIMENTAL RESULTS

The situation-based probabilistic prediction approach has been implemented in the AutoSim simulation package developed by Advanced Technology Research Corporation. AutoSim is a high-fidelity simulation tool which models details about road networks, including individual lanes, lane markings, intersections, legal intersection traversibility, etc. Using this package, we have simulated typical traffic situations (e.g., multiple cars negotiating around obstacles in the roadway, bi-directional opposing traffic, etc. and have predicted the future location of individual vehicles (TV) on the roadway based upon the prediction of where other vehicles (TV) are expected to be (Figure 8).

At the point this paper was written, we have simulated numerous driving situations and have used eleven costs to determine the probabilities of one action over another. In all driving situations, there were anywhere from one to three vehicles and obstacles placed at different random locations on the roadway. In all cases, there were no “road blocks”, meaning that there was always at least one lane on the roadway that would allow a vehicle to pass. The initial velocity of the vehicles varied from being at rest to 30 m/s. Current costs are incurred based on: 1) proximity to other objects in the environment as a function of the necessary stopping distance, 2) two costs associated with exceeding or going below the speed limit by a given threshold, 3) changing lanes, 4) not being in the rightmost lane, 5) five costs associated with acceleration profiles (constant velocity, slowly accelerating and decelerating, rapidly accelerating or decelerating), and 6) changing lanes where double yellow lines in the road exist.. It should be emphasized that costs are not static numbers. Using these costs, we were able to predict up to ten seconds into the future at a rate of five predictions per second. Determining the costs that are appropriate for a given aggressivity of a driver is an art more than a science. The ability of a system to adjust the costs as a function of perceptions of vehicle’s behavior is expected to have dramatic effects of determining and refining the costs, but this has not been implemented yet.

In the experiments, it was interesting to see how the vehicles (TV) performed when the “action sequences” (as described in Section 4.3.2) were varied with respect to time. As a reminder, the action sequence is the amount of time that it would take the vehicle to perform a driving maneuver (e.g., changing lanes). For most of these experiments, a passing maneuver, as described in Section 4.3.2, was used as the action sequence to be evaluated. When we set this time to a longer duration (e.g., 7 s), the vehicle was very slow to respond to unexpected events (e.g., a vehicle pulling into its lane quickly), which cause, in some cases, near collisions. Conversely, when the time for the action sequence was set to a shorter time (e.g., 2 s), the vehicle becomes jerky and could not make up its mind. For example, it would start a lane change and then quickly return to the original lane. Four seconds appears to be a good middle-ground to create realistic traffic patterns. The existence of these realistic traffic patterns is important to be able to truly assess the performance of autonomous vehicles in on-road driving scenarios.

Future work will create different action sequence timeframes for different types of driving maneuvers. The introduction of action sequences into the prediction algorithms have resulted in dramatic increases in performance with respect to time. Before the concept of action sequences was introduced, we were able to predict up to 5 seconds into the future for two vehicles at a rate of two predictions per second. Once actions sequences were introduced (along with the rules which state which action sequences are valid and invalid, as described in Section 4.3.2.), we are now able to predict 10 s into the future with two vehicles at a rate of 10 predictions per second.

All experiments described in the section were performed on a Pentium 4 CPU machine with a 1.8 GHz CPU and 512 MBytes of memory.

7. CONCLUSIONS AND FURTHER WORK

In this paper, we have described the PRIDE framework, which was developed to perform moving object prediction for autonomous ground vehicles (AV). PRIDE is based upon a multi-resolutional, hierarchical approach that incorporates multiple prediction algorithms into a single, unifying framework. The lower levels of the framework (not discussed in detail in the paper) utilize estimation-theoretic short-term predictions while the upper levels utilize a probabilistic prediction approach based on situation recognition with an underlying cost model. We showed the results achieved from applying PRIDE in a simulated environment.

We have then shown how PRIDE can be extended to simulate traffic patterns during on-road navigation. The PRIDE algorithms are used to provide the underlying logic to control vehicles (TV) in the environment, thus generating a realistic flow of traffic. We use the prediction algorithms within PRIDE to determine the probability that a vehicle (TV) will exhibit a certain behavior given a set of environmental conditions, and then command the vehicle (TV) to perform the action which has the greatest probability. By doing this, we are able to create realistic, repeatable, and non-scripted traffic patterns that closely mimic the types of traffic flow expected to be encountered during on-road driving.

We then explored how the PRIDE-based traffic control algorithms can be applied to performance evaluation of autonomous vehicles (AV). Through the use of repeatable and realistic traffic flow simulation, one is able to evaluate the performance of an autonomous vehicle (AV) in an on-road driving scenario without the risk involved with introducing the vehicle (AV) into a potentially dangerous roadway situation. In addition, by varying a single vehicle's (TV) parameters (e.g. aggressivity, speed, location) with the traffic flow, we can show how the entire traffic pattern is affected.

The goal of this paper was to describe how the work to date in moving object prediction could contribute to performance metrics for autonomous systems. Though the algorithms described in this paper have been developed and implemented, there is still much work to be accomplished. For one, the metrics that should be applied when evaluating an autonomous system in the presence of traffic situations being generated by these algorithms need to be further explored. This will be the topic of future work and will be realized in the NIST Autonomous Road Driving Arenas [23] as a way to assess the performance of autonomous vehicles being tested within these arenas. Also, this set of algorithms has been shown to work successfully on straight roads, but has not been fully tested on intersections. This will also be a topic of future research.

Another area of future research will involve the incorporation of intent into the predictions. For example, if one could recognize that the "intent" of a vehicle was to get off at an upcoming exit and it would have to move over three lanes to get to the exit ramp, then the system could dynamically adjust the costs to align the possible future actions with the intent of the vehicle. For example, in this scenario, one could adjust the cost of changing lanes to have a very low cost, thus increasing the probability that this will happen.

There is also the need to ensure that the outcome of these prediction algorithms truly represent the expected behavior of drivers in those situations. We plan on enacting the driving situations that we are simulating and compare the reaction of the physical drivers with the predictions from our algorithms to determine the accuracy of the predictions. This will be performed once the algorithms are fully tested in a simulated environment.

ACKNOWLEGEMENT

This work was supported by the Defense Advanced Research Projects Agency (DARPA) Mobile Autonomous Robot Software (MARS) program (PM. D. Gage) and the Army Research Lab (ARL) (PM. C. Shoemaker).

REFERENCES

1. "The DARPA Grand Challenge," *http://www.darpa.mil/grandchallenge/index.html*, 2005.

2. Albus, J., "The NIST Real-time Control System (RCS): An Application Survey," *Proceedings of the 1995 AAAI Spring Symposium Series*, 1995.

3. Albus, J. and et al, "4D/RCS Version 2.0: A Reference Model Architecture for Unmanned Vehicle Systems," NISTIR 6910, National Institute of Standards and Technology, Gaithersburg, MD, 2002.

4. Albus, J. S., "RCS: A Reference Model Architecture for Intelligent Control," *Computer*, Vol. 25, No. 5, 1992, pp. 56-59.

5. Balakirsky, S., *A Framework for Planning with Incrementally Created Graphs in Attributed Problem Spaces*, IOS Press, Berlin, 2003.

6. Bessière, P., Ahuactzin, J.-M., Talbi, E.-G., and Mazer, E., "The 'Ariadne's Claw' Algorithm: Global Planning with Local Methods," *Proceedings of the 1993 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, Vol. 2, 1993, pp. 1373-1380.

7. Champion, A., Espie, S., and Auberlet, J., "Behavioral Road Traffic Simulation with ARCHISM," *Proceedings of the Summer Computer Simulation Conference, USA*, 2001.

8. Dickmanns, E. D., "An Expectation-Based Multi-Focal Saccadic (EMS) Vision System for Vehicle Guidance," *Proceedings of the 9th International Symposium on Robotics Research (ISRR'99)*, Salt Lake City, 1999.

9. Dickmanns, E. D., "The development of machine vision for road vehicles in the last decade," *Proceedings of the Int. Symp. on Intelligent Vehicles '02*, Versailles, 2002.

10. Elnager, A. and Gupta, K., "Motion Prediction of Moving Objects Based on AutoRegressive Model," *IEEE Transactions on Systems, Man and Cybernetics - Part A: Systems and Humans*, Vol. 28, No. 6, 1998, pp. 803-810.

11. Espie, S., Saad, F., and Schnetler, B., "Microscopic traffic simulation and driver behaviro modeling: the ARCHISM project," *Proceedings of the Strategic Highway Research Program and Traffic Safety on Two Continents*, Lille, France, 1994.

12. Ferguson, I. A., "TouringMachines: An Architecture for Dynamic, Rational Mobile Agents," *Unpublished PhD Thesis*, University of Cambridge, UK, 1992.

13. Firby, J., "Architecture, Representation, and Integration: An Example from Robot Navigation," *Proceedings of the 1994 AAAI Fall Symposium Series Workshop on the Control of the Physical World by Intelligent Agents*, New Orleans, LA, 1994.

14. Gueting, R. H., "A Foundation for Representing and Querying Moving Objects," *ACM Transactions on Database Systems (TODS)*, Vol. 25, No. 1, 2000, pp. 1-42.

15. Haag, M. and Nagel, H.-H., "Incremental recognition og traffic situations from video image sequences," *Image and Vision Computing*, Vol. 18, 2000, pp. 137-153.

16. Hoseini, S., Vaziri, M., and Shafahi, Y., "Combination of Car Following and Lane Changing Models as a Drivers' Optimization Process," *Applications of Advanced Technologies in Transportation Engineering*, Vol. 0-7844-0730-4, 2004, pp. 601-605.

17. Kallman, M. and Mataric, M., "Motion Planning Using Dynamic Roadmaps," *Proceedings of the IEEE International Conference on Robotics and Automation (ICRA)*, New Orleans, LA, 2004.

18. Kalman, R., "A New Approach to Linear Filtering and Prediction Problems," *Transactions of the ASME Journal of Basic Engineering*, Vol. 82, No. Series D, 1960, pp. 35-45.

19. Kindel, R., Hsu, D., Latombe, J.-C., and Rock, S., "Kinodynamic Motion Planning Amidst Moving Obstacles," *Proceedings of the 2000 IEEE International Conference on Robotics & Automation*, Vol. 1, 2000, pp. 537-543.

20. Moravec, H., "Sensor Fusion in Certainty Grids for Mobile Robots," *AI Magazine*, Vol. 9, No. 2, 1988, pp. 61-74.

21. Ratering, S. and Gini, M., "Navigation Amidst Unknown Moving Obstacles," *Autonomous Robots*, Vol. 1, No. 2, 1995, pp. 149-165.

22. Schlenoff, C., Madhavan, R., and Barbera, T., "A Hierarchical, Multi-Resolutional Moving Object Prediction Approach for Autonomous On-Road Driving," *Proceedings of the 2004 ICRA Conference*, 2004, pp. 1956-1961.

23. Scrapper, C., Balakirsky, S., and Weiss, B., "Autonomous Road Driving Arenas for Performance Evaluation," *Proceedings of the Performance Metrics for Intelligent Systems (PerMIS) 2004 workshop*, 2004.

24. Sharma, R., "Locally Efficient Path Planning in an Uncertain, Dynamic Environment using a Probability Model," *IEEE Transactions on Robotics and Automation*, Vol. 8, No. 1, 2003, pp. 105-110.

25. Shoemaker, C. and Bornstein, J. A., "Overview of the Demo III UGV Program," *Proceedings of the SPIE Robotic and Semi-Robotic Ground Vehicle Technology Conference*, Vol. 3366, 1998, pp. 202-211.

26. Singhal, A., *Issues in Autonomous Mobile Robot Navigation*, Computer Science Dept, U. of Rochester 1997.

27. Williams, B. and Kim, P., "Model-based Reactive Programming of Cooperative Vehicles for Mars Exploration," *Proceedings of the Int. Symp. on Artificial Intelligence, Robotics and Automation in Space*, St. Hubert, Canada, 2001.

28. Zhu, Q., "Hidden Markov Model for Dynamic Obstacle Avoidance of Mobile Robot Navigation," *IEEE Transactions on Robotics and Automation*, Vol. 7, No. 3, 1991, pp. 390-396.

TABLES

Time Step In The Future	**Vehicle ID**	**Vehicle Type ID**	**Xposition [m]**	**Yposition [m]**	**Probability [%]**
1	10	2	10.5	11.5	40
1	10	2	11.5	11.5	20
1	10	2	10.5	10.5	30
1	10	2	11.5	10.5	10
1	11	2	10.5	12.0	30
...	...	...	...	...	...

Table 1. MOP Output

Actions					Validity	Description
SD	SD	SD	SD	QD	Valid	
QD	QD	QA	QA	QA	Invalid	QD to QA illegal
QA	QA	SA	SA	KS	Valid	

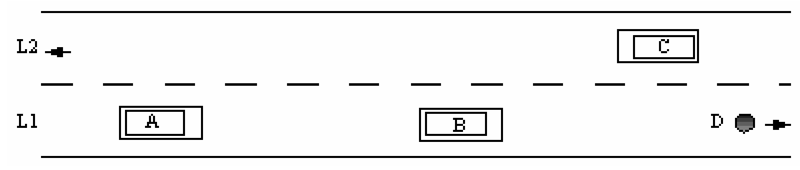
Table 2. Example of valid and invalid Set of action

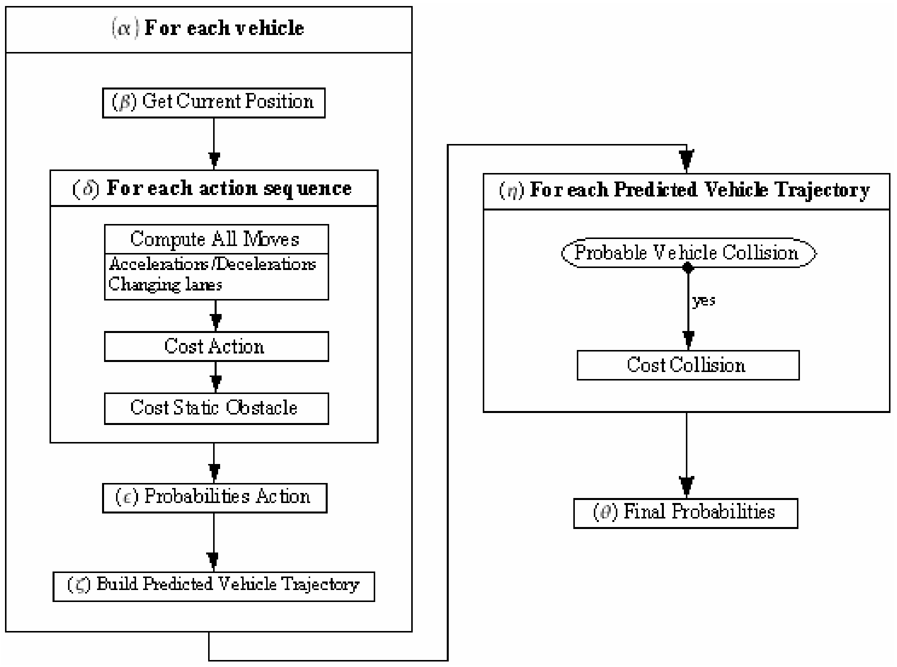
Action	**Cost**
Quick Acceleration (QA)	3
Slow Acceleration (SA)	2
Keep the same Speed (KS)	1
Slow Deceleration (SD)	2
Quick Deceleration (QD)	4
Changing Lane (CL, CR)	30
Opposite direction	500
Collision (CO)	1000

Table 3. Example Cost Model Values

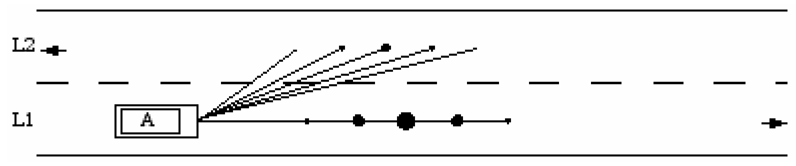
FIGURE CAPTIONS

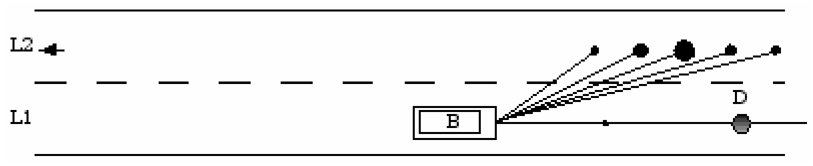
FIGURES

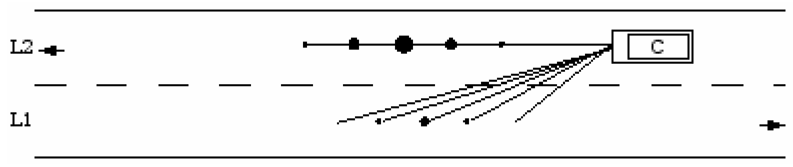


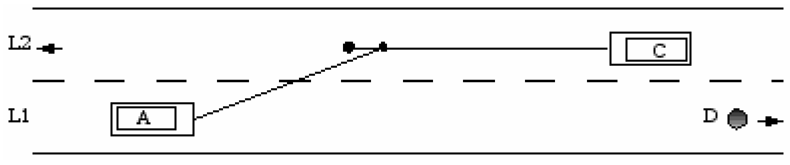


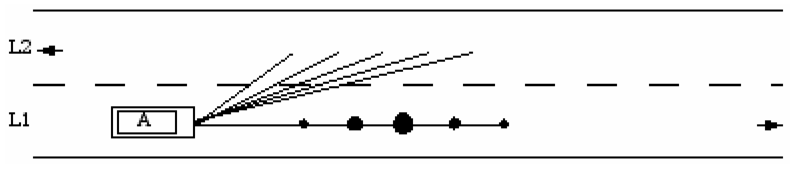
(α) For each vehicle
(β) Get Current Position
(δ) For each action sequence
Compute All Moves
Accelerations/Decelerations
Changing lanes
Cost Action
Cost Static Obstacle
(ϵ) Probabilities Action
(ζ) Build Predicted Vehicle Trajectory
(η) For each Predicted Vehicle Trajectory
Probable Vehicle Collision
yes
Cost Collision
(θ) Final Probabilities

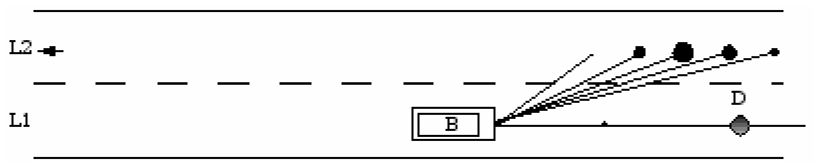

L2
L1
A

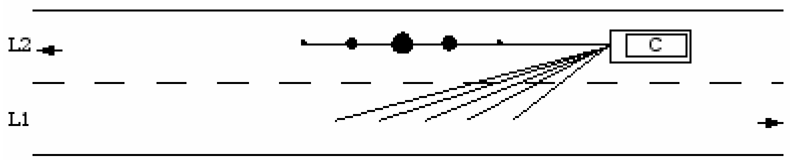

L2
L1
B
D

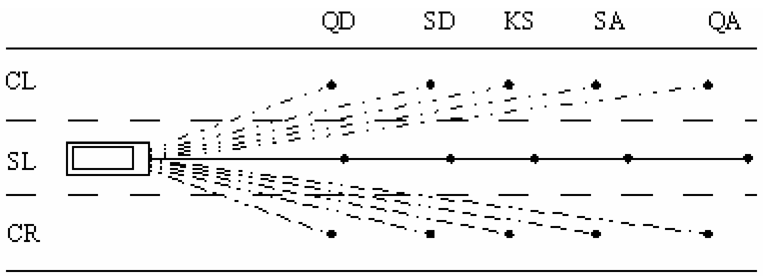

L2
L1
C

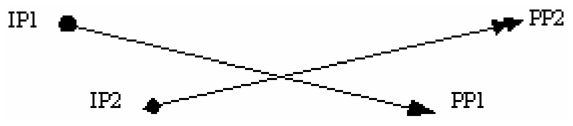

L2
L1
A
C
D


L2
L1
A


L2
L1
B
D


L2
L1
C


QD
SD
KS
SA
QA
CL
SL
CR


IP1
PP2
IP2
PP1