Monte Carlo Road Safety Reasoning.

Adrian Broadhurst, Simon Baker, and Takeo Kanade

The Robotics Institute, Carnegie Mellon University
5000 Forbes Avenue, Pittsburgh, PA 15213, USA
Tel: +1 (412) 268-5746, Fax: +1 (412) 268-5571
Email: {adrianb, simonb, tk}@cs.cmu.edu

***Abstract*— This paper presents a framework for reasoning about the future motion of multiple objects in a road scene. Unlike previous approaches, we do not look for known dangerous configurations of objects, but rather we reason about the future paths of all objects in the scene, and assess their danger. Monte Carlo path planning is used to generate a probability distribution for the possible future motion of every car in the scene. This framework can be used to either control the car, or to display warnings for the driver.**

I. Introduction

Many different types of sensors have been developed to detect cars, obstacles and pedestrians. These use a variety of techniques, such as laser scanners, radar, ultrasound, vision, inter-vehicle (IVC) communication, and road-vehicle communication (RVC). These approaches all attempt to provide the car with a map of the road and other road users, *at the current time*. This sensor data is used to provide safety warnings for the driver in known dangerous situations, such as: blind spot detection for overtaking, side collision detection [1], curb detection [2], or rear-end collision [3]. Other systems have used simultaneous localization and mapping [4] to identify obstacles. These approaches guarantee that a small number of known dangerous situations are avoided.

In this paper we argue that simply knowing there is an object at location $\mathbf{x}$ at time t does not provide sufficient information to asses its safety. A framework is needed for understanding the behavior of all the vehicles, pedestrians, obstacles and other objects on the road. The safety of the road must then be determined by considering the combined actions, and interactions, of all these objects. Can we confidently calculate that the road is safe for the next t seconds?

This is a challenging task because we must simulate both the behavior of our own car, as well as that of all other objects in the scene. We must consider the possibility of new objects entering the scene, objects leaving the scene, and the possibility of sensor failure. In addition, the simulation of objects is challenging because they are governed both by physical limits (such as maximum speed) but predominantly by human behavior. A well behaved driver will obey road conventions, a conservative driver may try to avoid accidents, and a reckless driver may take unexpected risks to avoid slowing down.

It is important to note that whether a collision is occurring *now* or whether a car is driving towards us *now*, is *not of direct use*. What *is* important, is whether or not we will be involved in a collision in the near *future*. To make this decision we must know the future, and there are many possible futures. This paper presents a framework for efficiently generating a probability distribution for the motion of multiple objects in a road scene. A technique for analyzing the probability of collision in the future, and a method for finding the safest path to avoid the danger. This paper is an extension of [5] which did not use probabilistic reasoning. In addition, this paper demonstrates reasoning about the interaction of multiple cars and pedestrians.

A. Prediction using path planning

In this paper we observe that predicting the future for another car, can be solved by path planning from that car's point of view. The following factors affect reasoning:

1) Vehicle dynamics: Every car is governed by physical mechanics. Given the initial state of a car, a series of control inputs (such as acceleration and steering), known properties of the road surface, tires, and weather; it is possible to calculate the trajectory of the car.

2) Human behavior: In practice, drivers do not use the full extent of their car's control inputs all of the time. As a result, predicting the path of a car is determined predominantly by human behavior. A detailed study of how humans choose their path in complex environments is [6].

3) Sensor uncertainty: In this paper we will assume a perfect sensor. Sensor uncertainty is a future consideration.

B. Elements of Planning

The planning process considers many possible future actions for all objects in the scene. For each possible future, the planning algorithm determines whether this future is safe, or results in a collision. For all the safe hypotheses, the algorithm generates a ranking according to how well they obey the drivers' priorities (or goal).

Previously, *start-goal* path planning problems [7] have been studied in the mobile robotics community. Many solutions exist including potential function approaches [8], [9] and provably complete sensor methods [10]. Planning algorithms have previously been applied to car-like robots [11], but not in the context of safety analysis, with future prediction, in complex multi-object environments.

1) Choosing the best (immediate) control action.: In any road scene, there is only one car which I can control, and that is *my* car. We can predict the likely path of other cars, and we

can predict how they *might* change their behavior, based on my action, but there is no guarantee.

For each possible action I can make *now*, I must consider all possible future outcomes, for all cars, and choose the paths which are least likely to cause an accident, and which best satisfy the goals of all drivers. This best *first action*, which leads to many future paths, is the output of the system that could be used to control the car.

2) Generating warnings: In many cases, we do not want to control the car, but rather want to warn the driver. In this case it is important to determine (i) which objects are dangerous, (ii) which locations on the road are dangerous. To do this, we need to consider all possible future paths of all objects, and calculate how likely each of these paths are, how severe the risk of danger is, and then combine these uncertainties to obtain a danger map.

3) Driver preference: In some situations the safest path may require sudden movements, or may be very slow. In these circumstances, the driver may choose to trade safety for comfort or speed. Thus the goal function must consider: path safety, mechanical limits of the car, ability and alertness of the driver, speed, and the degree of comfort.

C. Output for the driver, and applications

The reasoning algorithm can be applied in three ways:

1) Closed loop control:: In this approach the car is controlled directly using the best predicted action. The driver is not included in the control loop. This can be used to implement an intelligent cruise control system.

2) Human in the loop:: In this approach the car displays the recommended path to the driver. This can be displayed as a route map on a road, or as the instantaneous action to be applied. (e.g. brake now!, turn right!, or turn left!).

3) Warning system:: The car does not display the best path, but rather displays warnings on the road and on objects to convey to the driver, dangerous regions of the road.

II. Method

The aim of the reasoning algorithm is to generate a probability distribution for the future motion of all cars in the scene. This is calculated by considering all possible control inputs for all objects. All the control inputs which lead to a collision are eliminated, and a goal function is used to rank all the safe inputs in order of desirability. The output is the best control input, or a probability distribution over all control inputs, of the likelihood that any two objects collide.

A. The road scene

A road scene consists of a drivable area, obstacles, cars, pedestrians and cyclists. The drivable area consist of a 2D plane, with a Euclidean co-ordinate system. All objects are assumed to lie on this surface. The surface is assumed to be bounded. An object is a rectangle with position, orientation, and size. Each object is assigned a position $\mathbf{x}(t) = [x(t)\ y(t)]^T$ and velocity $\mathbf{v} = [\dot{x}\ \dot{y}]^T = v[\cos\theta\ \sin\theta]^T$. Together, these represent the object's state $\mathbf{s}(t)$. Objects also have two control inputs $\mathbf{u}(t)$. The motion of each object is defined by a differential equation of the form $\dot{\mathbf{s}}(t) = f(\mathbf{s}(t), \mathbf{u}(t))$. The implementation details of each object type are:

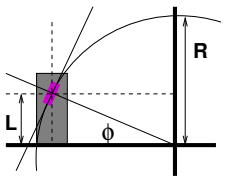

Fig. 1. Steering diagram: radius calculation

1) Obstacles: can be any size, and are assumed to be stationary. The state of an obstacle is defined as $\mathbf{s}(t) = [x\ y\ \theta]^T$. The state update equation is defined by $\dot{\mathbf{s}}(t) = 0$.

2) Pedestrians: are assumed to be 0.5m square, and move independently in the x and y directions. The state is defined as $\mathbf{s}(t) = [s_1\ s_2\ s_3\ s_4]^T = [x\ y\ \dot{x}\ \dot{y}]^T$. The state update equation is defined by $\dot{\mathbf{s}}(t) = [\dot{s}_1\ \dot{s}_2\ \dot{s}_3\ \dot{s}_4]^T = [s_3\ s_4\ u_1\ u_2]^T$ where $\mathbf{u}(t) = [u_1\ u_2]^T = [a_x\ a_y]^T$ is the control input.

3) Cars: are assumed to be rectangles[1] with origin situated at the center of the rear axle. The wheelbase is denoted L. The state of the car is defined as $\mathbf{s}(t) = [s_1\ s_2\ s_3\ s_4]^T = [x\ y\ v\ \theta]^T$ where v is the current speed of the car, and θ is the orientation. The control inputs[2] are $\mathbf{u}(t) = [u_1 u_2]^T = [a\ \phi]^T$, where a is the acceleration or braking term and ϕ is the steering angle. The radius of the car's motion is given by $L = R\sin\phi$ (see Figure 1) and the speed along the circle is given by $v = R\dot{\theta}$, thus $\dot{\theta} = \frac{v}{L}\sin\phi$. The state update equation is:

$$\dot{\mathbf{s}}(t) = \begin{bmatrix} \dot{s_1} \\ \dot{s_2} \\ \dot{s_3} \\ \dot{s_4} \end{bmatrix} = \begin{bmatrix} s_3\cos s_4 \\ s_3\sin s_4 \\ u_1 \\ \frac{s_3}{L}\sin u_2 \end{bmatrix} \quad (1)$$

B. The path of an object

The path of the object is calculated by applying the 4th and 5th order Runga-Kutta-Fehlberg algorithm to the state update equations: $\dot{\mathbf{s}} = f(\mathbf{s}, \mathbf{u})$. This is denoted $\mathbf{s}(t) = \mathbf{p}(\mathbf{s}_0, \mathbf{u}, t)$. The control inputs $\mathbf{u}(t)$ for the object, are sampled at 0.5 second intervals for 5 seconds[3].

C. Detecting collision between objects

To determine the safety of a path we must ensure, (i) that the object remains within the drivable region, and (ii) that it does not collide with other objects, and (iii) that the dynamic limits of the car are satisfied (e.g. the car will not skid or roll). In our implementation, if the car becomes uncontrollable, then

[1]We use 1.8m $\times$ 4.8m, wheelbase 2.4m (Lexus LS430).

[2]The range of control inputs is: $-0.5 < \phi < 0.5$ (radians), and $-9.1 < a < 4.3$ (ms^{-2}). These values are based on 0-60mph (6.3sec), and braking test results for the Lexus LS430.

[3]The sampling period (0.5s) is very long compared with PID controllers. At the milli-second scale, our algorithm is essentially an open-loop controller, and will not react to sudden unpredictable motions of other objects. The reason is computational cost, which is exponential in the number of sample periods. Non-linear sampling is an alternate solution.

the path is labeled as a collision. If a more complex dynamic model for the car is available, then this constraint could be relaxed.

1) Boundary constraints:: A map is defined as a binary image $m(x,y) \in \{0,1\}$, where each pixel is labeled as drivable or not-drivable. A collision occurs if the car's position $\mathbf{x}(t) = [x(t)\ y(t)]^T$ does not lie within the map bounds, or if the car enters a non-drivable region of the map. In our implementation the resolution of the map is 4 pixels/m.

2) Collision detection:: A collision between two objects occurs, if at anytime between $t = 0$ and $n\Delta t$, the objects intersect. Our implementation performs the collision test in two steps: (i) if the bounding circles intersect, then (ii) check that the corners of A do not lie inside B, and visa-versa.

3) Mechanical limits: It is unsafe for a car to attempt a U-turn at 55mph, because the tires have insufficient grip to complete the maneuver. The roll effects of the car are neglected. The maximum force that can be applied to the front tire is assumed to be $\|\mathbf{F}\| < F_{max}$ (we use $\frac{1}{m}F_{max} = 9.1\text{ms}^{-2}$). We also ensure that the grip direction does not change significantly $\frac{\partial \mathbf{F}}{\partial t} < \dot{F}_{max}$. This force can be calculated from the tangential and orthogonal (radial) components of acceleration on the tire.

$$\mathbf{F}(t) = m\begin{bmatrix} a(t)^2 \\ \frac{v(t)^2}{R} \end{bmatrix} = m\begin{bmatrix} a(t)^2 \\ \frac{v(t)^2}{L}\sin\phi(t) \end{bmatrix} \tag{2}$$

D. Goal distribution

In this paper, we are interested in determining which future control inputs are safe and which are dangerous. We are also interested in ranking the safe control inputs in order of their desirability. For this purpose, we define a scalar goal function $g(\mathbf{u}(t))$ for each car. This goal describes how well the car behaves in road space, however its input parameters, are parameterized in control space.

Given an initial state $\mathbf{s}_0$ and a particular control input $\mathbf{u}(t)$, we can calculate the path $\mathbf{p}(\mathbf{s}_0, \mathbf{u}, t)$. We can also determine whether this path results in a collision, or loss of control. For the safe paths, let the car's speed $\|\mathbf{v}\|$, and distance to the center of lane d, be normally distributed with means defined by the target speed and the center of the lane. Let the prior distribution be:

$$P(\mathbf{u}) = k_v e^{-(\frac{\|\mathbf{v}(\mathbf{p}(\mathbf{s}_0,\mathbf{u},t))\| - v_0}{\sigma_v})^2} k_d e^{-(\frac{d(\mathbf{p}(\mathbf{s}_0,\mathbf{u},t))}{\sigma_d})^2} \tag{3}$$

This can be written as a log-probability goal function, with $g : \mathbb{R}^{2n} \to \mathbb{R}$, defined as:

$$\log P(\mathbf{p}(\mathbf{s}_0,\mathbf{u},t)) = -k_g g\Big(\mathbf{u}, \mathbf{p}(\mathbf{s}_0,\mathbf{u},t)\Big) \tag{4}$$

In this paper, two goal functions are used:

1) Straight line following: This is the simplest goal function. There are three terms: (i) distance of $[x(t)\ y(t)]^T$ to line $\mathbf{l} = [l_u\ l_v\ l_w]$, (ii) difference in speed between $[\dot{x}(t)\ \dot{y}(t)]^T$ and $\mathbf{v_0}$, and (iii) sum of squares magnitudes of the control inputs. The latter is a regularization term which favors smooth (comfortable) paths[4].

$$g(\mathbf{u},\mathbf{p}) = \sum_{t=0}^{n} \lambda_1 \frac{(\mathbf{l}.[x(t)\ y(t)\ 1]^T)^2}{l_w^2(l_u^2 + l_v^2)} + \lambda_2(v(t)-v_0)^2 + \lambda_3 a(t)^2 + \lambda_4 \phi(t)^2 \tag{5}$$

2) Road following: This goal function requires a road-map with three functions defined: $d_m(x,y), \theta_m(x,y), v_m(x,y)$. The desired speed is defined by $v_m(x,y)$ (based on the local speed limit), the desired orientation by $\theta_m(x,y)$, and the lateral position by $d_m(x,y)$, which is defined as the distance from any point to the center of the lane.

$$g(\mathbf{u},\mathbf{p}) = \sum_{t=0}^{n} \lambda_5 d_m(x(t),y(t))^2 + \lambda_6(\theta(t) - \theta_m(x(t),y(t)))^2 + \lambda_2(v(t) - v_m(x(t),y(t)))^2 + \lambda_3 a(t)^2 + \lambda_4 \phi(t)^2 \tag{6}$$

E. Multiple objects

A scene consists of many objects. Each object i has an initial state $\mathbf{s}_{0i}$, a set of control inputs $\mathbf{u}_i(t)$, and a goal function $g_i(\mathbf{u},\mathbf{p})$, which specifies the path $\mathbf{p}(\mathbf{s}_{0i}, \mathbf{u}_i, t)$. The combined goal of all objects in the scene is defined as:

$$P(\mathcal{U}) = P(\mathbf{u}_1, \ldots, \mathbf{u}_m) = \prod_{j=1}^{m} P(\mathbf{u}_j)^{\alpha_j} \tag{7}$$

where α_j is a priority weighting for each object. Under normal conditions $\alpha_j = 1$, although at yield intersections these parameters can be varied to implement right-of-way.

F. Probability of danger

The probability of a collision in the future is determined by considering all possible *future* control inputs $\mathcal{U}$. The probability of a collision, for a specific set of control inputs $\mathcal{U}$, is $P(\mathcal{C} \mid \mathcal{U}) = \{0,1\}$, and the prior probability for this input is $P(\mathcal{U})$. The total probability of a collision is:

$$P(\mathcal{C}) = \int P(\mathcal{C} \mid \mathcal{U}) P(\mathcal{U})\ \partial\mathcal{U} \tag{8}$$

In this equation, the binary $P(\mathcal{C} \mid \mathcal{U})$ term, acts as a complex boundary condition in $\mathcal{U}$ space.

G. Warning generation and best paths

The total probability of collision $P(\mathcal{C})$ is the probability that at least one object will *not* safely navigate the scene. This can be used to display a warning to the driver, and can be implemented as a scalar bar which rises and falls, or as a quantized red, yellow, green display, or as a warning buzzer.

The most likely estimate of $P(\mathcal{C} \mid \mathcal{U})P(\mathcal{U})$ is used to find the best actions $\mathcal{U}$ for all cars. This output is useful for an air-traffic-control style automated driving solution:

$$\mathcal{U}_{max}(t) = \arg\max_{\mathcal{U}} P(\mathcal{C} \mid \mathcal{U}) P(\mathcal{U}) \tag{9}$$

[4]The values of λ are: $\lambda_1 = 0.0085/n$, $\lambda_2 = 0.05/n/(1 + v(t_0)^2)$, $\lambda_3 = 0.05/n/a_{max}^2$, $\lambda_4 = 0.05/n/\phi_{max}^2$, where $n = 10$ is the number of t-samples of $\mathbf{p}$. All units S.I. Values were chosen empirically.

For road vehicle use, it is more important to be able to generate warnings for the driver, than to control the car. Whilst $P(\mathcal{C})$ is useful as a general warning indicator, it is more useful to display the probability of danger for each road position $\mathbf{x}$ and time t. This display is a projection of $P(\mathcal{C} \mid \mathcal{U})P(\mathcal{U})$ onto the road space.

$$P(\mathcal{C} \mid \mathbf{x}, t) = \int P(\mathcal{C} \mid \mathcal{U})P(\mathcal{U} \mid \mathbf{x}, t)\ \partial\mathcal{U} \tag{10}$$

The new prior term $P(\mathcal{U} \mid \mathbf{x}, t)$ is calculated from the object goal prior $P(\mathcal{U})$ and a term $P(\mathbf{x}, t \mid \mathcal{U}) \in \{0, 1\}$ which specifies whether any object drives through point $\mathbf{x}$ at time t. $P(\mathbf{x}, t)$ is assumed equal to one.

$$P(\mathcal{U} \mid \mathbf{x}, t) = \frac{P(\mathcal{U})P(\mathbf{x}, t \mid \mathcal{U})}{P(\mathbf{x}, t)} \tag{11}$$

The result $P(\mathcal{C} \mid \mathbf{x}, t)$ is a danger level display for each road location, at each future time. In the results section of this paper, we show $P(\mathcal{C} \mid \mathbf{x})$, which also projects out time.

III. Implementation

This paper presents two main contributions, the first is the framework for evaluating the probability of a future collision $P(\mathcal{C})$. The second contribution is the use of Monte Carlo random sampling for the approximation the integral (8). This is significantly faster than systematic evaluation.

A. Dynamic programming

In our previous paper [5], we demonstrated the use of dynamic programming for evaluating (9). All the control inputs are discretised in time, and in control space, and represented as a decision tree. Finding the $\arg\max_{\mathbf{u}} P(\mathbf{u})$ can be efficiently implemented using a tree structure, because many branches of the tree can be quickly eliminated as candidates for the global maximum. Dynamic programming is an efficient way of calculating this maximum, but cannot be used for integration, as it relies on being able to eliminate unfavorable paths.

B. Monte Carlo sampling

An efficient algorithm for approximating integral (8) is Monte Carlos random sampling [12]. Let $\mathbf{x}_i$ be random samples drawn from the k-dimensional uniform random variable $\mathbf{X}$. Let the domain $\mathcal{X}$ enclose a volume $\mathrm{vol}(\mathcal{X})$, so that the uniform density function is given by $u(\mathbf{x}) = 1/\mathrm{vol}(\mathcal{X})$. The integral can then be written as an expectation, and approximated by random sampling.

$$\int_{\mathcal{X}} f(\mathbf{x})\partial\mathbf{x} = \mathrm{vol}(\mathcal{X})\mathbb{E}_u(f(\mathbf{X})) \approx \frac{1}{m}\sum_{i=1}^{m} f(\mathbf{x}_i) \tag{12}$$

A more efficient approach uses importance sampling. Rather than using a uniform distribution $u(\mathbf{x})$, use a density function $p(\mathbf{x})$ which simplifies (12). Specifically, try to ensure $f(X)/p(X) \propto 1$.

$$\mathbb{E}(f(\mathbf{X})) = \int_{\mathcal{X}} f(x)p(x)dx = \mathbb{E}_p\left(\frac{f(\mathbf{X})}{p(\mathbf{X})}\right) \tag{13}$$

corner	$\theta_i = \begin{cases} y_1 & i = i_1 \in [1, n] \\ 0 & otherwise \end{cases}$
change lane	$\theta_i = \begin{cases} y_1 & i = i_1 \in [1, n] \\ -y_1 & i = i_2 \in (i_1, n] \\ 0 & otherwise \end{cases}$
overtake	$\theta_i = \begin{cases} y_1 & i = i_1 \in [1, n] \\ -y_1 & i = i_2 \in (i_1, n] \\ -y_1 & i = i_3 \in (i_2, n] \\ y_1 & i = i_4 \in (i_3, n] \\ 0 & otherwise \end{cases}$
emergency stop	$accel_i = \begin{cases} 0 & i < i_1 \in [1, n] \\ -y_1 & otherwise \end{cases}$

TABLE I
DEFINITIONS OF RANDOM SAMPLE GENERATORS

The Monte Carlo estimator is then

$$\widetilde{f_m^p} = \frac{1}{m}\sum_{i=1}^{m} \frac{f(\mathbf{x}_i)}{p(\mathbf{x}_i)} \tag{14}$$

The error of the approximation $\epsilon \propto \frac{\sigma}{\sqrt{m}}$ can be decreased by choosing $p(x)$ to minimize σ (e.g. $p \approx f$), or by increasing the number of samples m.

C. Sampling distributions

Equation (8) contains a binary term $\mathrm{P}(\mathcal{C} \mid \mathcal{U})$ which acts as a complex boundary condition, and a Gaussian term $\mathrm{P}(\mathcal{U})$ which is integrated. Our first sampling distribution is, hence, a Gaussian sampling distribution:

$$p_1(\mathbf{u}) \sim G(0, \sigma) \tag{15}$$

The remaining sampling distributions are 1 or 2 dimensional linear subspaces of $\mathbf{u}$:

$$p_i(\mathbf{u}) = p(\mathbf{0} + \mathbf{A_i}\mathbf{y}) \quad \text{where } \mathbf{y} \sim G(0, \sigma) \tag{16}$$

The $\mathbf{A}_i$ are chosen heuristically to mimic human behavior. On their own, these p_i are not mathematically correct as they do not span the whole space of control inputs, rather, they span the reduced space of typical human driver actions. Consequently, the p_i's are used in combination with the full rank p_1 to ensure a valid result:

$$p(\mathbf{u}) = \sum_i k_i p_i(\mathbf{u}) \quad \text{where } \sum k_i = 1 \tag{17}$$

These actions are shown in Table I and Figure 2.

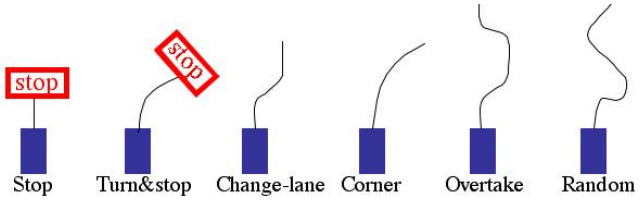

Fig. 2. Random sample generators

Fig. 3. **Q&A from JAF 'Stop the accident booklet'.** What is dangerous about the left hand road scene? Notice the car with the reverse light. The answer (right), shows the bicycle swerve to avoid the reversing car.

IV. Results

The example road scene, shown in Figure 3, shows a road scene with a parked car and a moving bicycle. The scene is dangerous because the parked car reverses into the road and the bicycle must swerve to avoid the reversing car. The driver of the test vehicle must anticipate the possibility that the bicycle will swerve into his lane. In this section we show how our reasoning framework can detect this danger. Consider three case studies:

The first example shows what would happen if the parked car reverses slowly, and the bicycle continues its current path. The reasoning algorithm is applied to *my* car only. This is implemented by assigning $\mathbf{u}(t) = 0$ for all other objects. The results are shown in Figure 4. The computation speed is 2Hz, on a 2GHz Intel PC. The shaded area (green) is the probability that my car will be present at each location on the road *at any time during the simulation period*. The blue line shows the single path which maximizes my goal function. Notice that the scene is safe for my car, but that the reversing car and bicycle collide. This shows that it is not sufficient to reason about my car alone, or to reason about objects independently.

The second case applies the reasoning algorithm to both my car and the bicycle. The result is that the bicycle will swerve and stop, to avoid the car, and that I must avoid the bicycle. This is the correct result.

The third case show what would happen if the reversing car, the bicycle and the car of interest, are all included in the reasoning process. Notice that the car chooses not to reverse because of the bicycle. This means that both the bicycle and car of interest can drive past, although there is a small level of risk. This is not what happened in practice. This result assumes too much intelligence for the reversing car. It is probably fair to assume that the car *cannot see* the bicycle. This leads to an interesting question for future work: How to handle visibility? How much reasoning can we assume for the human? Is he paying attention? What cases should be considered? and what is their relative importance?

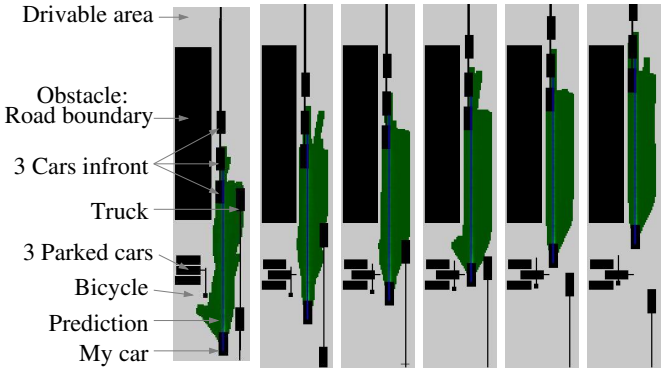

Fig. 4. **Case 1: Car reverses, bicycle continues straight**. This figure shows the state of the world at half second increments. The three black squares on the left represent parked cars. The big black square is an obstacle and represents the side of the road. There are three cars in-front of my car (bottom). The shaded (green) area shows the predicted location of my car over the next 5 seconds. All areas which are shaded have non-zero probability. Brighter intensity shows higher probability. Notice that if the bicycle and car collide, but I am still safe.

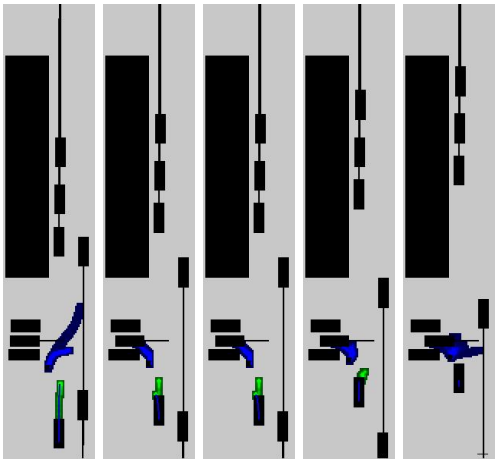

Fig. 5. **Case 2: Car reverses, intelligent bicycle.** Bicycle must swerve to avoid car. I must stop to avoid bicycle.

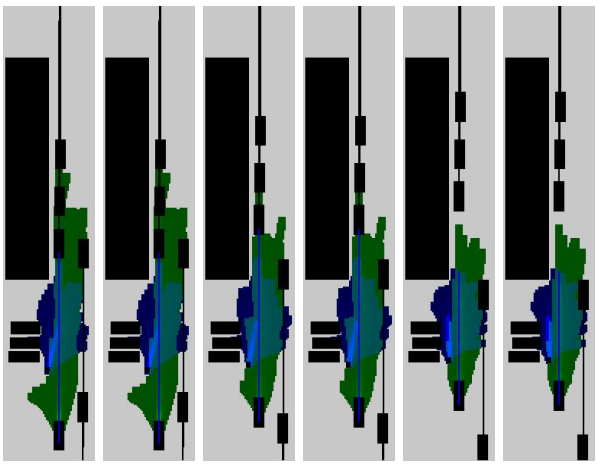

Fig. 6. **Case 3: Intelligent car, intelligent bicycle.** Car chooses not to reverse. No collision. I am safe (small risk).

V. Conclusion

This paper has presented a reasoning framework, for analyzing the safety of complex road scenes. Each object was assigned a goal function, and Monte Carlo integration was used to evaluate the probability that all the objects would safely achieved their goal. It was also shown how this probability could be used to generate warning displays for the driver. The framework has been tested using synthetic data, based on a real world example, with varying levels of complexity. It has been shown that it is important to *reason* about the *interactions* of all cars, and that *constant velocity* assumptions about other cars are often *incorrect*.

Acknowledgments

The authors would like to thank DENSO CORPORATION for funding this research, and Iain Mathews, Takahiro Ishikawa and Junya Inada for providing valuable assistance.

References

[1] C. Mertz, "A 2d collision warning framework based on a monte carlo approach," in *Proc. ITS America 14th meeting and expo*, Apr. 2004.

[2] R. Aufrere, C. Mertz, and C. Thorpe, "Multiple sensor fusion for detecting location of curbs, walls, and barriers," in *Proc. of the IEEE Intelligent Vehicles Symp.*, June 2003.

[3] L. Yang, Y. J.H., E. Feron, and V. Kulkarni, "Development of a performance-based approach for a rear-end collision warning and avoidance system for automobiles," in *Proc. of the IEEE Intelligent Vehicles Symposium (IV2003)*, June 2003, pp. 316–321.

[4] C. Wang, C. Thorpe, and S. Thrun, "Online simultaneous localization and mapping with detection and tracking of moving objects: Theory and results from a ground vehicle in crowded urban areas," in *IEEE Int. Conf. on Robotics and Automation*, May 2003.

[5] A. Broadhurst, S. Baker, and T. Kanade, "A prediction and planning framework for obstacle avoidance, path planning, and warning generation," in *ITS for livable society*, Nagoya, Japan, 2004.

[6] B. Fajen and W. Warren, "Behavioral dynamics of steering, obstacle avoidance, and route selection," *Journal of Experimental Psychology*, vol. 29, no. 2, pp. 343–262, 2003.

[7] J. Latombe, *Robot motion planning*. Boston, MA: Kluwer, 1991.

[8] O. Khatib, "Real-time obstacle avoidance for manipulators and mobile robots," *International Journal Robotics Research*, vol. 5, pp. 90–98, 1986.

[9] D. Rimon, E.and Koditschek, "Exact robot navigation using artificial potential functions," *IEEE Trans. Robotics Automation*, vol. 8, no. 5, pp. 501–518, 1992.

[10] S. Lumelsky and A. Stepanov, "Path planning strategies for point mobile automation moving amidst unknown obstacles of arbitrary shape." *Algorithmica*, pp. 403–430, 1987.

[11] A. Lambert, S. Bouaziz, and R. Reynaud, "Shortest safe path planning," in *Proceedings of the IEEE Intelligent Vehicles Symposium (IV2003)*, June 2003, pp. 282–287.

[12] J. Hammersley, "Monte carlo methods for solving multivariable problems." *Ann. New York Acad. Sci.*, vol. 86, pp. 844–874, 1960.