A prediction and planning framework for road safety analysis, obstacle avoidance and driver information

Adrian Broadhurst, Simon Baker, and Takeo Kanade

CMU-RI-TR-04-11

February 2004

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 prediction and planning framework for analysing the safety and interaction of moving objects in complex road scenes. Rather than detecting specific, known, dangerous configurations, we simulate all the possible motion and interaction of objects. This simulation is used to detect dangerous situations, and to select the best path. The best path can be chosen according to a number of different criterion, such as: smoothest motion, largest avoiding distance, or quickest path. This framework can be applied, either as a driver warning system (open loop), or as an action recommendation system (human in the loop), or as an intelligent cruise control system (closed loop). This framework is evaluated using synthetic data, using simple and complex road scenes.

Contents

1 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, between car communication, and car-road communication. 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 [9], curb detection [1], or rear-end collision [12]. Other systems have used simultaneous localisation and mapping [11] to identify obstacles. These approaches guarantee that a small number of known dangerous situations have been 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 behaviour 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 behaviour 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 behaviour. 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. These situations must all be considered.

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 possibilities. This paper presents a framework for enumerating all possible future scenarios, analysing them, and making recommendations for the driver based on their likelihood.

2 Framework

The prediction and planning framework consists of three components. First, all possible future scenarios are predicted. Second, a path planning algorithm uses the prediction results to determine the safest path through the scene. Finally the output is used, either to control the car, or to display safety information to the driver.

2.1 Elements of Prediction

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, tyres, and weather; it is possible to calculate the trajectory of the car. 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 both by the physical capabilities of the car, but predominantly by human behaviour.

Human behaviour A detailed study of how humans choose their path in complex environments is presented in Fajen & Warren [3]. There is no physical reason preventing oncoming traffic from colliding. Both cars could very easily turn in towards each other and crash. The reason they do not collide, is that both drivers obey the rules of the road, and stay in their own lane. This poses a problem for systems which attempt to asses the safety of a road scene. Almost every scene with oncoming traffic is theoretically dangerous. Road death statistics in any country show that driving is dangerous. What we must consider, is the degree of safety that we are prepared to accept, and our confidence in the ability of other drivers, to also drive safely.

Sensor uncertainty In order to predict the future, we need to understand the error in our understanding of the present. Sensors can be uncertain for a number of reasons: they can incorrectly classify objects, they can contain imprecise measurements, or detection could fail, or they could group multiple objects into a single detection result. In this paper we will assume a perfect sensor. The use of a sensor model will be included in future work. This model must provide a probability distribution for each class of object, over the surface of the road and surrounding area. It must also provide a probability distribution for the likelihood of a missed detection, at each road location, given the local context of that region of the road scene.

Objects entering the scene Another important point to consider is the possibility that new objects may enter the scene. The road in front the car is the most obvious point of entry for a new car. On a straight road, this point is on the horizon. On a corner this entry-point will be the point where the road becomes occluded. All intersections, and gateways are possible entry-points, as are blind spots caused by buildings, trucks and cars. The detection of occlusion boundaries is a known graphics problem [4]. Every occlusion boundary which borders the road, and is large enough to obscure an object, must be considered as an entry-point.

Multiple hypothesis The most simple prediction of the future, is that all objects will continue to move at their current speed. This, however, is only one of many future outcomes. Each action the driver makes, leads to a different future outcome. In addition, all other objects can also change their motion. The main issue for implementing the prediction algorithm, is the choice of method for enumerating all possible driver actions, as well as all the actions of all other objects in the scene, over the next t seconds.

2.2 Elements of Planning

The planning process considers each of the many future predictions of the road scene and determines which hypothesis will lead to a collision, and which are safe. Exactly which path is the best outcome, is dependent on the priorities of our driver. Previously, *start-goal* path planning problems [7] have been studied in the mobile robotics community. Many solutions exist including potential function approaches [5, 10] and provably complete sensor methods [8]. Our approach uses a map based approach [2]. Planning algorithms have previously been applied to car-like robots [6], but not in the context of safety analysis, with future prediction, in complex multi-object environments.

Finding the safest path through a scene. Each hypothesis of the future defines a series of actions for our car, and for all other objects in the scene. These actions define a paths for each object. A hypothesis is determined to be dangerous if any two object paths collide.

There are many different actions that we can make as driver, however, we cannot control the whole scene. Other drivers can also influence the future. For each action we can make, there are many hypothesis for the future. Some of these future outcomes are safe and some are dangerous. The safety of a particular control action, we make, is determined by considering the likelihood and safety of each of the resulting hypothesis. A conservative algorithm labels a control action as unsafe if any future outcome, resulting from that action, is unsafe. A more realistic algorithm assesses the driver's tolerance for risk, the driver's assumptions about other road users, and the likelihood of each hypothesis, and combines these probabilities in a probabilistic manner, to determine if a control action is safe, or not.

If the safety of all possible control actions is considered, then one control action can be chosen as the safest. This choice of control action still results in many future outcomes, but based on our hypothesis-safety function, we know this will lead to the most likely safe outcome. In some cases, we may choose to select the most likely path resulting from this action, and display this information to the driver.

Dynamic safety A separate issue to path safety, is dynamic safety. How likely is the car to physically realise a series of control inputs. This is dependent on the properties of the car, the road surface, and the current weather conditions. A separate but related issue, is the ability of a driver to accurately implement a series of control actions. This is dependent on the particular driver, and his current mental state, such as alertness. Whether or not these terms are included in the hypothesis-safety function, depends on the particular application being developed.

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 hypothesis-safety function must consider many factors including: path safety, dynamic safety, the ability and alertness of the driver, the speed, and the degree of comfort.

2.3 Visualisation and driver warnings

So far we have described the space of all actions which lead to a safe scenario, and the best path for a specific object to navigate the scene. From this information we also know the space of control actions (set of paths) that will lead to a collision. But how can this danger information be displayed concisely to the driver? We consider three options:

Display best path: The driver is shown a recommended path that the system considers most safe. This is calculated using the technique described in the previous section.

Road map: Each path traverses the road surface, so every point on the road will be touched by zero or more paths. We can classify each point on the road surface with a degree of safety based on the distribution of safe and unsafe paths that traverse that point. This projection of the decision tree onto the road can be used as a warning display.

Object label: Similarly, in every path, there is a minimum distance between any two objects. This distance can be used to classify each object as a possible threat to the driver, or not. One classification heuristic might be minimum distance. A warning is then displayed alongside each potentially dangerous object.

It is important to note that warning systems are not an exact science. Detailed user interface studies would need to be conducted, to determine effective heuristics for displaying dangerous road-regions and objects, to the driver. The calculation of the best path, however, is well principled, assuming the human factors are suitably approximated.

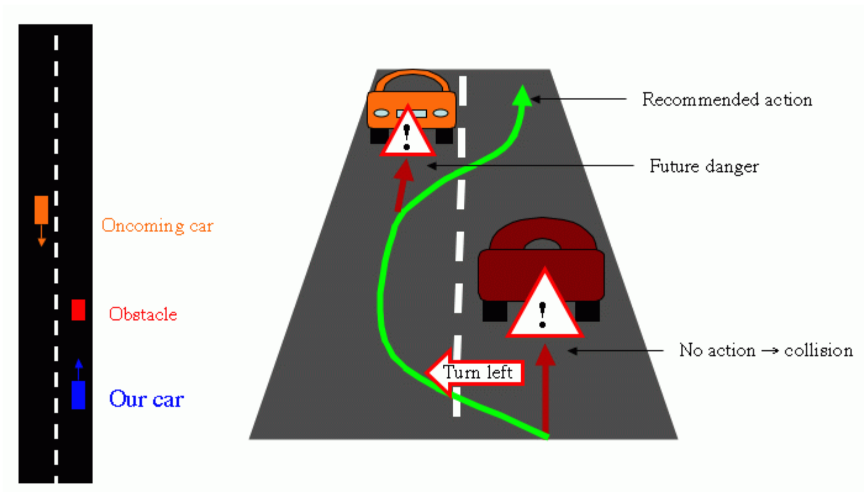


Figure 1: This figure shows an example warning display. It shows the recommended path, dangerous paths, dangerous objects and the best instantaneous control action.

2.4 Applications

The path planning and warning algorithms can be used to solve many different problems.

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, except in an emergency. This can be used to implement an intelligent cruise control system.

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!).

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 regions of action space which are dangerous.

Sensor analysis: It is important for the computer vision and physical-sensor communities to know a target accuracy for car and object detectors. By applying this algorithm to a large number of typical road scenes, it would be possible to determine whether automated control is safe or not, given the level of accuracy of the sensors in question.

3 Method

This section describes the prediction and planning framework in detail, starting with some basic definitions about road scenes. First, a single time instant is considered, then the effect of motion. The interaction of objects is modeled using game theory, with all objects moving in turns, one turn at a time. All the possible actions that an object might make are represented in a decision tree. Each leaf node describes a possible path through the scene. These paths can be classified as safe or unsafe by running a simulation, and checking that no two objects collide.

3.1 The road scene

A road scene consists of a drivable area, obstacles, cars, pedestrians, cyclists, and entry-points, at a specific instant in time. Each object is assigned a speed and velocity, and this is referred to as the objects state. An obstacle is an object with zero speed. A moving object has non-zero velocity, and an interactive object has the ability to change its velocity dependent on the road scene. An entry-point is a disc with an radius which increase at a fixed rate. This area represents the maximum distance that an unseen object can cover in a given period of time. The drivable area consist of a 2D surface with a Euclidean grid defined on its surface. All objects are assumed to move on this surface and obey the laws of Newtonian mechanics. The surface is assumed to be bounded, and the simulated car must remain strictly inside this road boundary. An object collision occurs when two objects occupy the same position on the road surface. A scene is safe if no two objects collide.

$$state_i(t) = \{position_i(t), velocity_i(t)\} = \{\mathbf{x}_i(t), \dot{\mathbf{x}}_i(t)\} \tag{1}$$

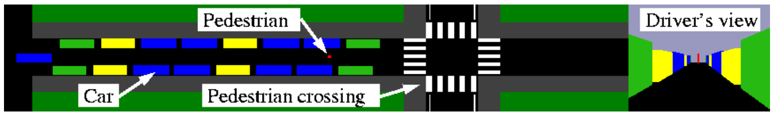


Figure 2: This scene consisting of a drivable area, non-drivable areas (sidewalk, grass), obstacles (parked cars), and a moving object (pedestrian).

3.2 Object motion and interaction

Interactive objects change their velocity (speed and direction), over time, based on the current road scene and an internal strategy function for each object. An interactive object may change its speed according to some strategy, which is dependent on the current state of the scene.

An action (2) is defined as the change in velocity of an object (3), which is determined by its strategy function. Below, $state_i(t_n)$ is the state of object i at time t_n (with $i < m$).

$$action_i(t_n) = \quad \ddot{\mathbf{x}} \quad = strategy_i(state_1(t_n), state_2(t_n), ..., state_m(t_n)) \tag{2}$$

$$state_i(t_{n+1}) = \{\mathbf{x}, \dot{\mathbf{x}}\} = \ state(t_n) \bullet action_i(t_n) \tag{3}$$

The state update ($\bullet$) is implemented, for turn duration Δt, as:

$$\mathbf{x}_i(t_{n+1}) = \mathbf{x}_i(t_n) + \dot{\mathbf{x}}_i(t_n) * \Delta t + \frac{1}{2}\ddot{\mathbf{x}}(t_n) * \Delta t^2 \tag{4}$$

$$\dot{\mathbf{x}}_i(t_{n+1}) = \dot{\mathbf{x}}_i(t_n) + \ddot{\mathbf{x}}_i(t_n) * \Delta t \tag{5}$$

The path of an object (6) is defined as an initial state, followed by a list of actions, for the following game-turns. A scenario (7) is defined as the paths of all objects in the scene.

$$path_i = \{state_i(t_0), action_i(t_0), \ldots, action_i(t_{max})\} \tag{6}$$

$$scenario = \{path_1(t_0), path_2(t_0), ...path_m(t_0)\} \tag{7}$$

3.3 Determining the safety of a scenario

A scenario consists of a list of objects together with a path for each object. To determine whether a scenario is safe, simulate all road scenes between t_0 and t_{max} and determine whether each scene is safe. A road scene is dangerous if any two objects collide. An object is controllable at t_n if it does not collide with any other object during the period $t_n \leq t \leq t_{max}$.

3.4 Game theory

In a board game, such as chess, players take their turn to make a move. In each turn, a player considers the state of the board, at that time, and then makes his best move. After he has finished, the next player considers the new state of the board, and makes his move. This process continues until the game is completed. A chess computer operates by simulating the future moves of both players, and then deciding which of these future outcomes is most beneficial. The current move is based on whether or not it leads to this beneficial outcome.

A simple chess computer can be implemented using a decision tree and a board evaluation function. The primary (root) node of the decision tree represents the current move. Each branch from this node represents a permissible move. Each branch results in a new state of the board. For each node, we add branches for the moves that the opponent could make, based on that state of the board. This process can be repeated for each player in turn, until all possible futures have been considered, or until we run out of computing resources. At this point, we apply the evaluation function to all leaf nodes, and the branches taken to get there, and select the best leaf. Since this is a tree structure, each leaf node must trace back to a primary branch. This primary branch represents the move that the computer must make now, to maximise its chance of winning the game.

One of the implementation issues is the game-turn allocation strategy, which defines when objects may take their turn to act. There are two common strategies: (i) sequential turns, or (ii) simultaneous turns. In the sequential approach, only one object may act at a particular time, and there is a predetermined sequence in which objects act. This is similar to Chess, Go and Monopoly. In the simultaneous approach, all players take their move at time t_n based on the state of the game at time t_{n-1}. All moves are then applied simultaneously. This is similar to the approach used in most reenactment war games. The simultaneous approach is usually preferable because the results are not biased by the predetermined turn sequence.

3.5 The road scene decision tree

A scenario is defined as the initial state and a path for every object being simulated. This particular set of actions represents only one, of many, possible future outcomes. The space of all possible scenarios is both large and complex. The action of each object is dependent on the actions of all the other objects in the scene. To simplify this problem, we propose that the road environment should be modeled as a board game, using Game theory. All objects are assumed to move in game-turns, one after another. In each game-turn, one object considers the state of the road, and then makes its best move based on its strategy function. After it has completed its game-turn, the next object makes its move.

By making this approximation, all object decisions can be enumerated in a decision tree. The primary (root) node in the decision tree represents the state of the road (all objects) at the current time. Each branch represents a permissible action that the first car could make. Each child-node describes the state of the road resulting from that action. For each new child-node, we build a new set of branches for all moves that the second car could make. This process is then repeated for all cars. All cars are then allowed to make second and third game-turns, until the required simulation period t_{max} is completed. Each walk from primary node to leaf defines a possible scenario, because it defines one unique series of actions for each objects in the scene. The safety of each scenario is then determined using Section 3.3.

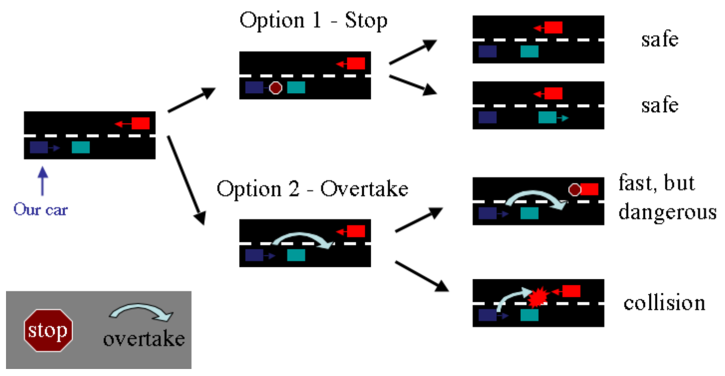


Figure 3: This figure shows an example decision tree. At the current time, the car can either stop, or overtake. Each of these decisions have a different consequence. If we stop, it is always safe, even if the obstacle starts moving. If the car overtakes, there is a chance that the oncoming car may stop, in which case the path is safe, however if the oncoming car does not stop, then it is dangerous. The best (conservative) action at the current time, is to stop. However, if our priority is the quickest path, whatever the risk, then there may be a safe outcome if the car overtakes.

3.6 Choosing the optimal path, for the current time

At this point we make an implementation decision. Our initial implementation does not consider the action of other drivers. This simplification will be removed in future work. This simplification means that the decision tree only contains actions of our own car, and is independent of other drivers. It also means a path through the tree defines a path for our car.

The previous section described how the decision tree represents all possible future scenarios. Each of these scenarios can also be classified as safe or unsafe. We now select one of the safe paths as the "best path" based on an optimality criterion. This criterion can be the path of least effort, or the quickest path, or the path which leaves the largest avoiding distance to other objects, depending on the driving strategy being implemented. After the best path has been chosen, only the action at time t_0 is implemented. We now know this action will lead to a safe outcome, over the period $t_0 < t < t_{max}$ (as long as the current object completes the maneuver, and that the other objects behave as predicted, and no new objects enter the scene).

3.7 Evaluating the strategy function

In this implementation, we have chosen to implement the smoothest path strategy, which is the path with smallest sum-of-squares actions. This path is simple to implement, and should be reasonably comfortable. It is more safe than the fastest path approach, which uses the sum-of-absolute magnitudes. Certain areas of the road surface are more desirable for driving, than other areas. For example, it is not desirable to drive on the wrong side of the road. A scalar field is described on the surface of the road which represents the desirability of a car to be present at each $\mathbf{x}$ location. This is called the $PositionPrior(\mathbf{x})$. In a similar manner, it is dangerous to drive too fast, or too slowly, so a $VelocityPrior(\|\mathbf{x}\|)$ is used. The magnitude of the prior terms is significantly lower than the cost of an action, but add up over the duration of the path. If this prior term is not used then the car will not complete an overtaking manoeuvre. Likewise, if the speed term is not used, then if the car ever stops to avoid a pedestrian, it will never start moving again. Thus these prior terms are of critical importance.

$$cost(path) = \sum_{t=t_0}^{t_{max}} cost(state(t)) + \sum_{t=t_0}^{t_{max}} cost(action(t)) \qquad (8)$$

$$cost(state) = PositionPrior(\mathbf{x})^2 + VelocityPrior(\|\dot{\mathbf{x}}\|)^2 \qquad (9)$$

$$cost(action) = \|\ddot{\mathbf{x}}\|^2 \qquad (10)$$

$$PositionPrior(\mathbf{x}) = \begin{cases} wrong\ lane & x_{lateral}.k_{lateral} \\ otherwise & 0 \end{cases} \qquad (11)$$

$$VelocityPrior(speed) = \begin{cases} speed > speed_{max} & speed.k_{fast} \\ speed < speed_{min} & speed.k_{slow} \\ otherwise & 0 \end{cases} \qquad (12)$$

3.8 Representing decisions and road state

Decision tree So far we have described the algorithm using a decision tree with quantised actions and a continuous state space (discrete time). In practice, the size of the decision tree is very large, with order $O(a^n)$ where a is the number of actions and n is the number of game turns considered. This is an action space representation. For each walk through the tree, we run a separate simulation to determine if there are any collisions. The only storage requirements, are a current path and best path. The main computing resource required is processing time.

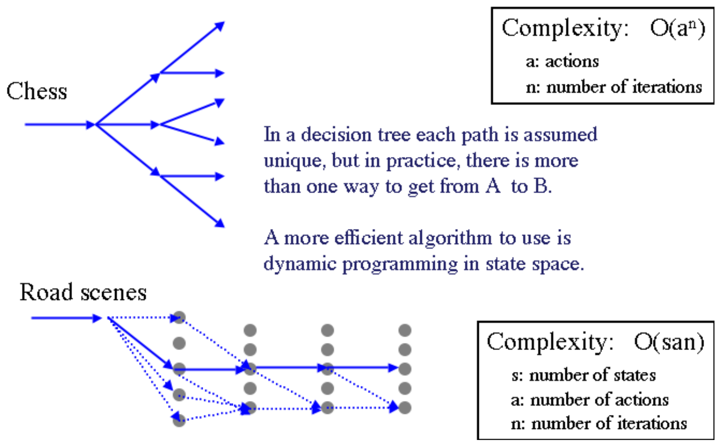
Graph representation To reduce the number of states and actions, the road can be simplified, using a graph representation. A multi-lane road can be represented as a series of nodes on the graph. The car is only allowed to change lanes, or to merge with an onramp, at specific points on the graph. We have not used this representation because it does not allow pedestrians and non-highway scenes to be represented. Instead we consider a quantised euclidean surface.

State space representation What the decision tree representation does not reflect, is that different walks through the decision tree may traverse the same partial path in state space. For example, you may go around an obstacle in the middle lane, by overtaking on the left or on the right, but both of these paths return to the middle lane and follow the same completing path. This lends itself to a dynamic programming implementation which evaluates in state space rather than decision space. In particular, it reduces the computational order to $O(sn)$ where s is the magnitude of state-space. This algorithm requires a very large number of states to be stored in memory. The key implementation decision is whether $sn < a^n$ which is determined by the number of turns n, the number of states s and the number of actions a.

Dynamic programming If we only consider the actions of our own car, then we may precompute a simulation of all other objects in the scene. This computation results in a binary obstacle map for each time-step:

$$obstacle(\mathbf{x}, t) = \bigcup_{i=1}^{m} ObjectAt(\mathbf{x}, t, state_i(t)) \qquad \in \{true, false\} \tag{13}$$

This obstacle map is sampled at a higher temporal resolution than the game time-step. This reduces the size of the decision tree. A collision can


Chess
Complexity: O(a^n)
a: actions
n: number of iterations
In a decision tree each path is assumed unique, but in practice, there is more than one way to get from A to B.
A more efficient algorithm to use is dynamic programming in state space.
Road scenes
Complexity: O(san)
s: number of states
a: number of actions
n: number of iterations

be detected by checking for an obstacle at every point $\mathbf{x}(t)$ along the path (over time).

$$Safe(path) = Safe\,(state(t_0), action(t_0), \ldots, action(t_{max})) = \bigcup_{t=t_n}^{t_{max}} obstacle(t, \mathbf{x}(t)) \tag{14}$$

Dynamic programming is an efficient algorithm for determining the best path. We define a quantised state space for the road $road(t, \mathbf{x}(t), \dot{\mathbf{x}}(t)) = \{action, cost\}$, where each cell contains the best action at that time and state, and the total cost from that time until the end of the game. The algorithm is presented in Figure 4.

3.9 Dynamic path planning

Every turn, new sensor information is available, and the decision making process is repeated using the new data. If the previous prediction was accurate, then the new sensor information should agree with our previous prediction.

```
begin
  Initialise the future-most road-state with the corresponding obstacle map.
  for all $\mathbf{x}, \dot{\mathbf{x}}$ do
      $road(t_{max}, \mathbf{x}, \dot{\mathbf{x}}) = \begin{cases} obstacle(t_{max}, \mathbf{x}) = true & \{action = 0, cost = \infty\} \\ \text{else} & \{action = 0, cost = 0\} \end{cases}$
  od
  Evaluate all entries in the road-state, working backwards through time
  for $t = (t_{max} - \Delta t)$ to $t_0$ do
      for all $\mathbf{x}, \dot{\mathbf{x}}$ do
          for all possible actions, find action with minimum $cost(t, state(t))$ do
              $future\_cost = Cost(road(t + \Delta t, state(t) \bullet action))$
              if $obstacle(t, \mathbf{x}(t))$
                then $cost = \infty$
                 else $cost = Cost(action) + Cost(state(t)) + future\_cost$
              fi
          od
          Assign $road(t, \mathbf{x}(t), \dot{\mathbf{x}}(t))$ with the best action and cost
      od
  od
  Select the best path through the road-state
  begin
    $state(t_0) =$ The car's current state (speed and position)
    for $t = t_0$ to $t_{max}$ do
        $action(t) = Action(road(t, state(t)))$
        $state(t + \Delta t) = state(t) \bullet action(t)$
    od
  end
end
```

Figure 4: Dynamic programming implementation

4 Evaluation and discussion

4.1 Experiment 1:

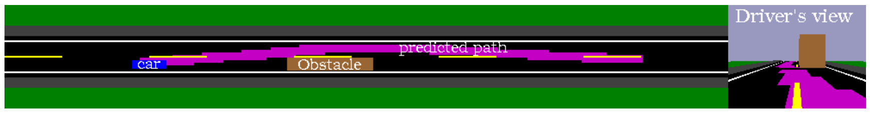
In this experiment, we demonstrate the decision making capability of our system. The road scene consists of a road with an obstacle, and the possibility of an oncoming car. The system must decide whether to stop or drive around the obstacle, depending on whether there is oncoming traffic or not. The results are shown in Figure 5. Experiment (a) shows the predicted best path around the obstacle. Experiment (b) shows the path around the obstacle, with an oncoming car at 75m. Notice how the car takes a steeper path than in (a). Experiment (c) shows how the car *cannot* overtake an obstacle with a car at 45m. The car waits for the oncoming pass, and then drives around the obstacle. This is shown in (c-i through iii).

Notice that the algorithm safely avoids the obstacle without hitting the oncoming car. The algorithm correctly makes the decision to wait for the oncoming car, when the available overtaking distance is too short. The shape of the car's path is not defined, stopping distances and time to impact are not modeled. The shape of the manoeuvre, and decision making functionality, is calculated as a result of the prediction and planning process.

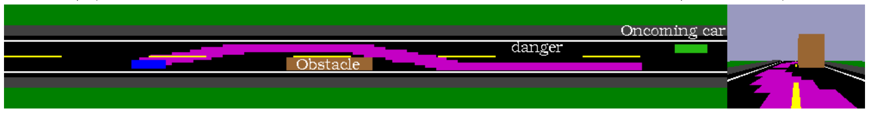
4.2 Experiment 2:

The second experiment demonstrates the use of entry-points. There is a car waiting at an intersection, and the system must consider the possibility that the car might start to move. The first experiment (a) ignores the possibility that the waiting car might move. The car initially plans to drive straight past (a-i), and is then unable to avoid the waiting car (a-ii), when it starts to move. The second experiment (b) uses an entry-point to model the possibility that the waiting car might move. Notice that the car initially plans to take a very wide path (b-i), but when it reaches the intersection, new sensor data shows that the car is not moving, so it can take a normal path through the intersection. In experiment (c) the waiting car is modeled with an entry-point. When the waiting car starts to move, the simulated car is able to drive past safely (unlike (a)).

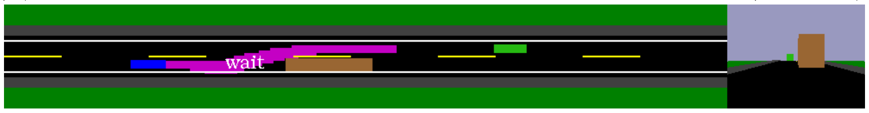
By assigning each entry-point a maximum speed we can model the unsafe region around each danger point. This guarantees a conservative solution is chosen at each time-step. As new sensor data becomes available, the danger area will always get smaller until a real object is detected. This allows the car to take a sensible but conservative path, but still be able to avoid a collision if necessary. It is hence essential to model the possibility of



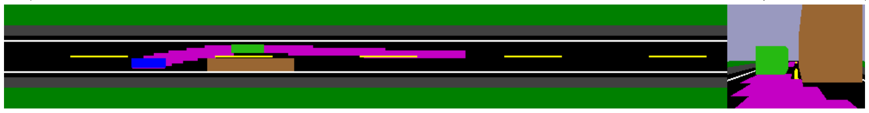
(a) Path planning result with no oncoming car (frame 0).



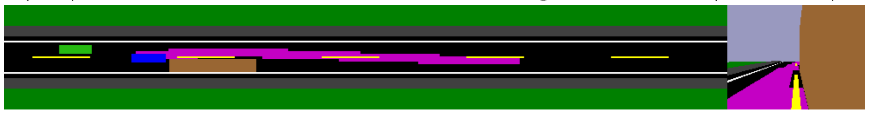
(b) It is safe to overtake with an oncoming car 75m ahead (frame 0).



(c-i) It is not safe to overtake with an oncoming car at 50m (frame 0).

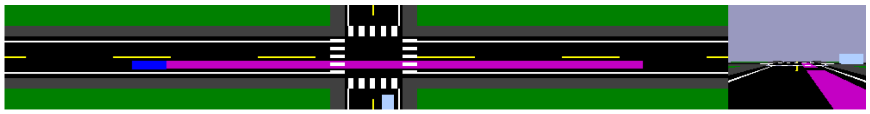


(c-ii) The car must wait for the oncoming car to pass (frame 55).

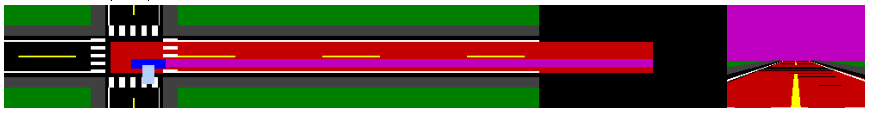


(c-iii) After the car has passed it is safe to overtake (frame 95).

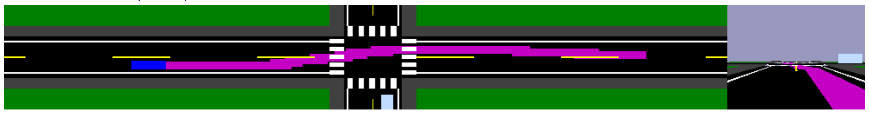
Figure 5: **Experiment 1:** This experiment analyses the safety of overtaking a stationary obstacle, with (a) an empty road, (b) an oncoming car at 75m ahead, and (c) a car at 45m. It is safe to drive around the obstacle in (a) and (b), but in experiment (c) the car must slow down, wait, and then overtake (see c-i to iii). Note: no overtaking manoeuvre is defined, the path is the result of simulation and the prior preference to drive on the right, and to keep moving.



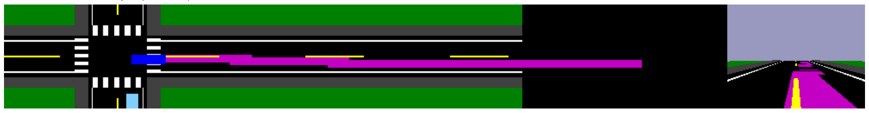
(a-i) Path planning at frame 0, without entry-point.



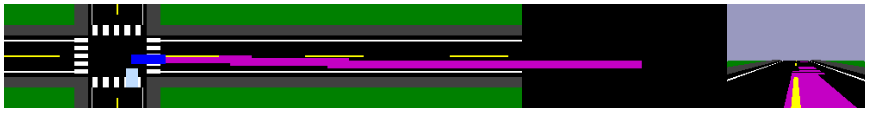
(a-ii) Collision at frame 60, without entry-point



(b/c-i) Path planning at frame 0, with entry-point



(b-ii) Path planning at frame 70, with entry-point, car is stationary.



(c-ii) Path planning at frame 70, with entry-point, car is moving.

Figure 6: **Experiment 2:** The use of entry-points. The car waiting at the intersection may start to move. In (a) the possibility of a moving is ignored. In (b) entry-points are used to model this possibility, but the car does not move. Entry-points are used in (c) and the car does move. Notice that entry-points are required to safely navigate the scene.

new objects entering the scene, and the possibility of objects changing their motion. This is not possible in traditional sensor based systems, which do not consider the future. This justifies the use of the prediction and planning framework.

4.3 Experiment 3 - Narrow street with pedestrians.

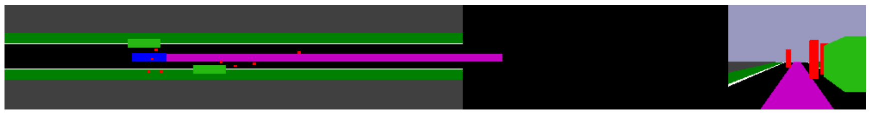
The third experiment shows how entry-points can be used to model unexpected objects entering the scene. This scene is inspired by a scene from a J.A.F. "Stop the accident" booklet. In this experiment, the car is driving down a narrow one way street with many pedestrians on the road, and two parked cars. At frame 70 an unseen pedestrian appears from behind the first parked car. This scene is modeled with and without entry-points, and with and without the unexpected pedestrian. Figure 7(a) shows the car hitting the unseen pedestrian. Figure (b) shows the car's initial plan to stop away from the pedestrian. As the car approaches, new sensor data shows that a pedestrian has not stepped out yet, and the car plans a path through the obstacles (c). At frame 70 the new pedestrian appears (d) and the car adjusts its path accordingly.

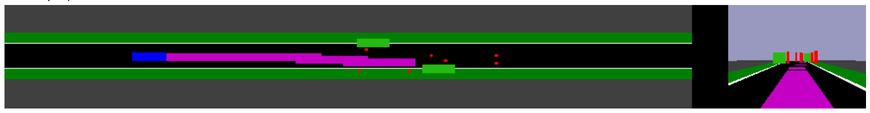
This example shows how the prediction and planning framework can navigate a complex scene with many pedestrians moving in different directions. In complex scenes there are many points where new objects could unexpectedly enter the scene. These events must considered if the system to accurately predict a safe path through the scene.

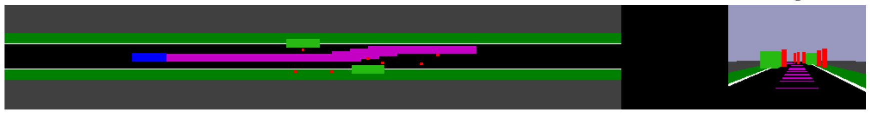
4.4 Experiment 4 - Parked cars, unexpected pedestrian

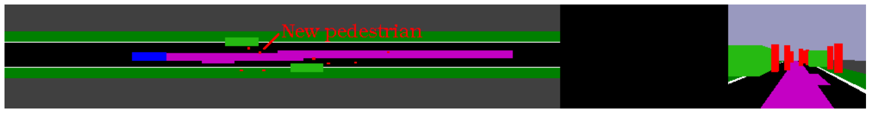
This experiment also demonstrates the importance of entry-points. It is also inspired by a scene from the J.A.F. "Stop the accident" booklet. The scene consists of a narrow one-way street with cars parked on both sides of the road. There is a pedestrian in the distance, walking towards the driver. On the right hand side of the road is a delivery van with boxes stacked on the ground. The point of the scene, is for the driver to notice the boxes, and expect the delivery driver to walk out into the street. In this experiment, we will ignore the specific cue of the boxes, and will consider the possibility of a pedestrian entering the scene from behind *any* car. To do this, we add an entry-point behind every car in the scene, and allow the prediction and planning framework to chose the optimal path and speed, which ensures the car is controllable at all times, even if a pedestrian enters the scene.

Figure 8(a) shows the predicted motion of all objects in the scene, at


(a) Simulation without entry-points. Car hits new pedestrian.


(b) With entry-points, frame 0. Car slows, and turns wide of danger point.


(c) With entry-points, frame 35. New sensor data. Car can pass danger point.



(d) Path planning, frame 70. New pedestrian!! adjust path.

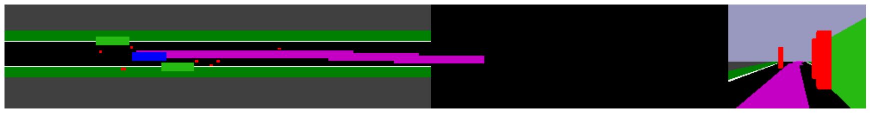
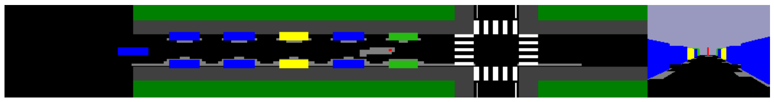

(e) Path planning, frame 130, obstacles will be avoided.

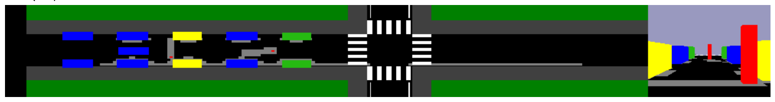
Figure 7: **Experiment 3:** This scene contains a narrow road, two parked cars, and many pedestrians. At frame 70 a new pedestrian enters the scene from behind the top car. In (a) new pedestrians are not modeled, and the car hits the pedestrian. (b-f) shows the same simulation, using entry-points. Notice how the car slows down, avoids the dangerous situation, misses the pedestrian, and completes the scene. Considering new objects in the future is essential!

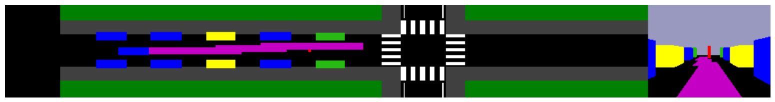
frame 0. At frame 45 a new pedestrian enters the scene and walks across the road (b). In the control experiment, we do not use entry-points. The initial path of the car is shown in (c). Between frames 44 and 45, the car must dramatically change its planned path, to avoid the new obstacle. In this experiment (without entry-points) the car is lucky that it is possible to find an avoiding path for the new pedestrian.

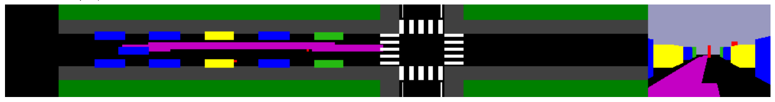
The second experiment, Figures (d-j), uses an entry-point to model the possibility of a pedestrian entering the scene. An entry-point is an elliptical region of uncertainty which has zero radius at the current time. The radius of the ellipse increases linearly over time at the theoretical maximum speed of the new object. The shape of the entry-points are shown in (d), 5 seconds in the future. Notice how the car plans to pass the danger points one parked car at a time (f) and (g). When the new pedestrian enters the scene the car slows down, waits for the pedestrian to pass (j), then continues on its way (k).

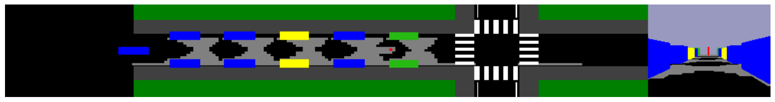
The important point to notice in this scene is that entry-points allow the car to predict new objects entering the scene. The car can then plan to navigate the scene at a safe speed, with sufficient time to stop, should a pedestrian step into the scene. Compare this to the control experiment. In the control case, the initial speed was specified, and the car planned to avoid the second pedestrian, so there was no need to drive slowly and carefully. Hence the violent swerving action in (c-d). In conclusion, this experiment shows that it is important to model the possibility of new objects entering the scene. It also demonstrates that the prediction and planning algorithm can effectively choose a safe speed to navigate a road. This would not be possible with traditional sensor based systems, which rely on speed limits to determine the safe driving speed.

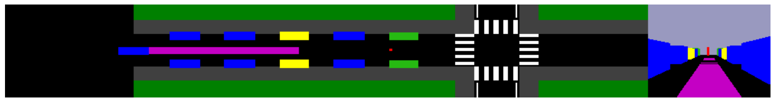

(a) Motion prediction at frame 0, for the next 5 seconds.

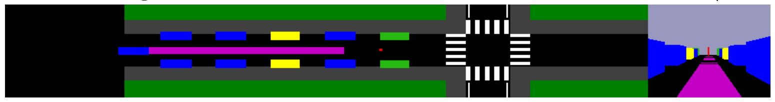

(b) Motion prediction at frame 45. A new pedestrian enters scene.

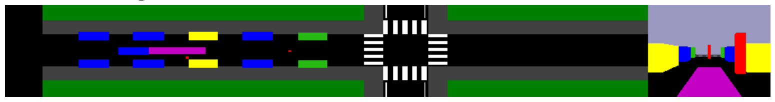

(c) Path planning, frame 44, without entry-point.


(d) Path planning, frame 45, new pedestrian, sudden path change.


(f) Motion prediction at frame 0. Observe the entry-points, between cars.


(g) Path planning at frame 0. Purple stripe shows car's path (5 seconds)


(h) Path planning, frame 5. Notice how each car is passed, one at a time.


(j) Path planning, frame 45. Car slows for pedestrian, no swerve needed.

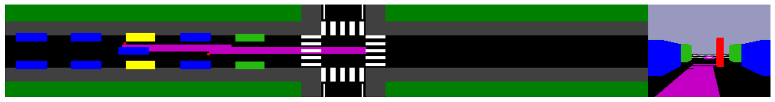

(k) Path planning, frame 85, obstacles will be avoided.

Figure 8: **Experiment 4:** This scene contains a narrow road, a pedestrian walking towards the driver (a), and an unexpected pedestrian entering the scene (b) at frame 45. Notice the sudden change in path in response to the new pedestrian (c,d) if entry-points are not used. In this example the pedestrian is lucky. The experiment is then repeated with entry-points (f) which model the possibility of a pedestrian entering from behind a parked car. Notice how the algorithm plans to pass each car one at a time, when it knowns that it is safe (f,g). This approach guarantees that the unseen pedestrian can be avoided (j).

5 Future work

This framework has been shown to effectively predict and plan safe paths through complex road environments, however, there are still several areas where future research is needed. Interaction with other cars has been described, but has not yet been demonstrated. The main issue that needs to be addressed is the modeling of driver behaviour. If the models are too well behaved then the problem is solvable but not useful, and if the human models are too general, then oncoming traffic will pose a significant problem. Warning generation has also been described. The main issue is the amount of detail available to the driver. The exact choice of algorithm will depend on human interface studies.

In this paper, all sensors and car controls are perfect. All examples were theoretically safe, which is an ultra-conservative condition. In the new implementation the obstacle map will be probabilistic. The decision about whether a situation is dangerous will depend on the drivers tolerance for risk, which may vary. In addition, the car may not accurately implement control actions (or the human driver is inaccurate). This extension is of particular interest to the authors, because real world sensors are not perfect, and most real road scenes cannot be driven without a small tolerance for risk.

The decision space of all opponent actions is very large. Instead of using actions in a decision tree, the opponents actions could be modeled as a probability distribution over position, speed and time. This reduces the decision space, but adds complexity to the cost function because it must now consider both, the probability of collision, and the probability of driver action in the same framework.

6 Conclusion

This paper has presented a prediction and planning framework, for analysing the safety of complex road scenes, consisting of moving and stationary objects. A decision tree has been used to enumerate all the possible future paths of the simulated car. A method for determining the safety of each path has been described. A strategy function is used to select the best safe path through the scene. Techniques are described for using this action to directly control the car, or to displaying warnings or recommending actions to the driver. The framework has been tested using synthetic data, on two simple and two complex road scenes.

Acknowledgments

The authors would like to thank Iain Mathews and Takahiro Ishikawa for their help in developing the ideas in this paper, and Denso Corporation for funding the research.

References

[1] 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.

[2] J.F. Canny. *The complexity of robot motion planning.* MIT Press, Cambridge, MA, 1988.

[3] B. Fajen and W. Warren. Behavioral dynamics of steering, obstacle avoidance, and route selection. *Journal of Experimental Psychology*, 29(2):343–262, 2003.

[4] J. Foley, A van Dam, S. Feiner, and J. Hughes. *Computer Graphics, Principles and Practice.* Addison-Wesley, 2nd C edition, 1996.

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

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

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

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

[9] C. Mertz. A 2d collision warning framework based on a monte carlo approach. In *Proceedings of ITS America's 14th Annual Meeting and Exposition*, April 2004.

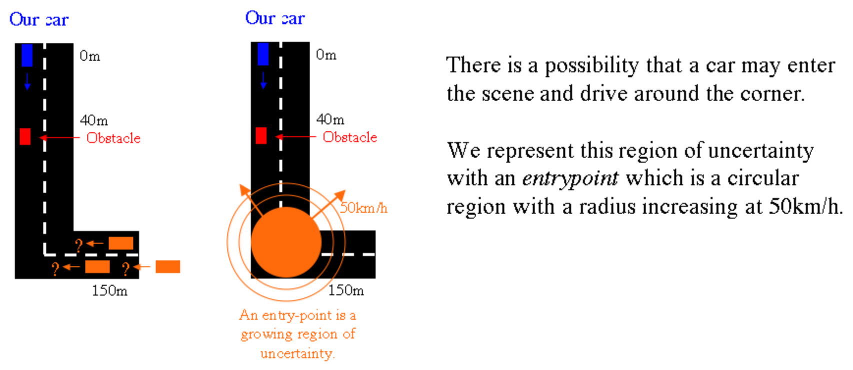
[10] D.E. Rimon, E.and Koditschek. Exact robot navigation using artificial potential functions. *IEEE Trans. Robotics Automation*, 8(5):501–518, 1992.

[11] C.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.

[12] L. Yang, Yang 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)*, pages 316–321, June 2003.

A Appendix: Road scene representation

The use of entry-points is mentioned in the main section of the paper. This appendix provides a couple more examples to aid the explanation. An entry-point is a point on the road surface, at the current time, where a new object may enter the scene. The region of the entry-point at time $t > 0$, represents all locations on the road surface where the unseen object could reach in a given time. If we assume the unseen object has a fixed maximum speed, then this region is described by a circle, with a radius increasing at the fixed maximum velocity. This is a very conservative model. If too many entry-points are used on a scene then the only conservative solution is for the car to stop. This will be addressed in the future probabilistic version of the framework. We now consider three examples, a blind corner, and intersection, and a pedestrian crossing. The explanation of each example is presented in the caption of Figures 9 through 11.



There is a possibility that a car may enter the scene and drive around the corner.

We represent this region of uncertainty with an *entrypoint* which is a circular region with a radius increasing at 50km/h.

Figure 9: An entry-point is used represent a car entering around a blind corner. The entry-point is located at the point where the road becomes occluded in the current drivers view.

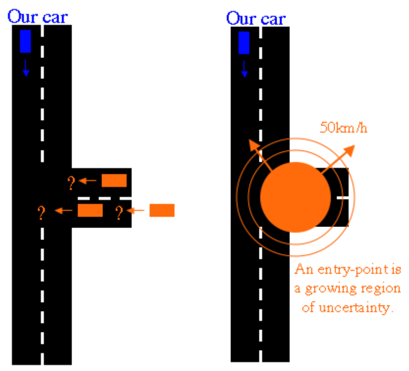


Figure 10: An entry-point is used represent a car entering at an intersection. The entry-point is located at the point on the road where the drivers view becomes occluded.

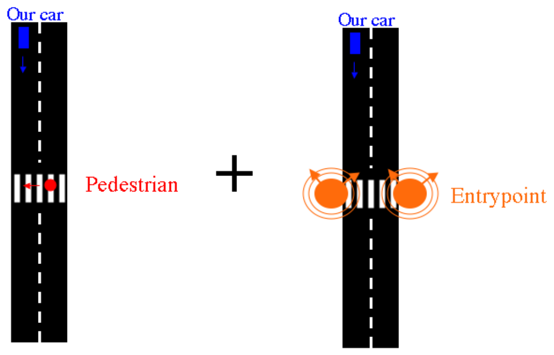


Figure 11: A pedestrian crossing is a region with a very high likelihood of detecting pedestrians. A pedestrian is modeled as a small object with slow speed and erratic behaviour. A pedestrian crossing is modeled with two entry points to represent the possibility of pedestrians stepping out into the road on either side.