Traffic Accident Simulation And Animation

Hala ElAarag and Mohamed Abdel-Aty
Stetson University and University of Central Florida
Department of Mathematics and Department of Civil and Computer Science Environmental Engineering
helaarag@stetson.edu mabdel@mail.ucf.edu

Keywords: Traffic Safety, Accident Simulation, Accident Animation, Accident Visualization

ABSTRACT

The objective of this research is to present a methodology to supplement the conventional traffic safety analysis techniques. This methodology aims at using computer simulation to animate and visualize accident occurrence at high-risk locations. It aids into developing the appropriate safety countermeasures for high-risk locations (e.g., intersections, curves, ramps). A computer accident simulation and animation program has been developed. The program includes many features including accident detection, location, vehicle, and animation attributes. The program has the ability to animate accidents at specific highway location based on the configuration of this location, which is entered by the user.

INTRODUCTION

Fatalities and injuries related to motor vehicle accidents constitute a major societal problem. Statistics show that in 1999 there were an estimated 6,279,000 police reports of traffic accidents. Also in 1999, 41,611 people were killed and 3,236,000 were injured in traffic accidents. Injuries from accidents dwarf all other causes of lost human productivity and reduced quality of life. Traffic safety researchers must understand how and why motor vehicle accidents occur in order for them to find treatments and countermeasures. The role of the research presented in this paper is to improve this understanding by developing a methodology to animate and simulate traffic accidents. The main source of accident data is the standard accident report forms filled out by the police officers. Some of the information recorded on this form is then coded into computerized format. The traditional methodology of analyzing traffic safety is to apply a variety of statistical techniques using these coded accident reports. There are numerous problems with such data. There are many inconsistencies and biases in accident data. Therefore, there is a need to develop a methodology to improve the traditional accident analysis techniques.

OBJECTIVES OF THE STUDY

The introduction of the automobile in the early 1900’s brought with it little concern for the safety of the occupants of these machines. Early automobiles were slow and few in number and accidents were rarely serious. The years hence have seen a dramatic rise in both the number and performance of automobiles with a resulting increase in the frequency and severity of automobile accidents. Today tens of thousands of people die each year on United States roads and highways. One approach to traffic safety improvement is to avoid collisions in the first place, as opposed to protecting the occupants once a collision has occurred. Improvements in highway design, road markings and road signage can be aimed at this goal. Achieving such improvements inevitably requires a deep understanding of the cause of automobile collisions and a method for varying the circumstances of collisions to see how such changes may improve collision avoidance. Since crashing real cars into stationary objects, and into each other, is a dangerous,
time consuming and expensive endeavor, computer simulation seems naturally suited to the task.

The objectives of this research could be summarized as follows:

- Develop a tool that would enhance current techniques in accident analysis
- Animate and visualize accident occurrence to improve our understanding of accident causes
- Identify safety problems
- Propose solutions
- Test countermeasures

OVERVIEW OF RESEARCH OBJECTIVES AND CONTRIBUTION

A computer simulation tool is developed within the framework of this project. Several elements of this project have been developed, including: visualization and animation capabilities including different view capabilities, collision detection, database search and classification, and geometric characteristics of the site. Other elements of the program have been partially developed, such as accident reconstruction, however, the core of the program is established and full addition of accident reconstruction capabilities is possible in an extension to this work. We also propose adding a learning capability to the program based on artificial Intelligence techniques, so that the model can detect if certain pattern of accidents tend to occur at certain location, and therefore identifying the problems with other similar locations. This would enable any designer to incorporate safety countermeasures in the design process without waiting and experiencing actual accidents and then develop countermeasures. Rather the model will be able to learn from existing locations’ safety and derive what we can expect in other locations (what we refer to here as the “opposite to accident reconstruction”). In this research we have focused on intersections, because more than half of the accidents occur at intersections or at the approach to intersections. It would be simple to extend the model to include other locations (e.g., freeway ramps, etc).

In this research, we designed and implemented an automotive collision simulator that provides visualization and insight into collisions. Additionally, our simulator allows the rapid variation of accident conditions. The program has an intuitive graphical user interface that provides the user with full control over the simulation, by modifying the vehicle specification, path or the accident scenario (position, direction and speed of each object involved in the accident). It also allows for a variety of rendering options (e.g., texture mapping, wire-frame or solid polygonal models, light, camera position…etc.). The program has a modular scalable design and can run on different platforms. For collision detection, we used a collision detection tool called RAPID. We extended the functionality of RAPID to provide more accurate results. The program is a time-based simulator that uses actual vehicle models. We considered several specifications of the physical properties of the vehicles e.g. mass, rigidity, elasticity, angular moment and friction coefficient. The program is easy to use and could be exploited in a variety of applications. The most important application is to use the program to improve the safety of automobiles, highways, and driver behavior.

THE NEW APPROACH: THE ACCIDENT SIMULATOR

In our approach, we designed and implemented the accident simulator that has the following features:

Intuitive Graphical User Interface (GUI):
This allows the user to easily modify the input parameters, through a number of windows. This includes the vehicle specification and the scenario of the collision. The user friendly GUI makes the data input phase much easier. It allows accommodating different types of users, even those who do not have any computer experience. Figure 1 shows the program's main user interface. It allows the user to specify the number of vehicles involved, camera position, rendering options, and the animation time slice. The rendering options allow the user to choose wire frame or solid rendering, smooth or flat shading, and light or texture mapping.

Simulation Control
Our simulator gives the user full control over the simulation process. The user can directly watch the consequences of changing any of the input parameters. The user can do that through several windows: Intersection geometry control, car control, view control and action control.

Interaction Control
Figure 2 shows the intersection geometry control. The Intersection Geometry module allows users to change intersection configuration. It shows an example of a 4-leg intersection with major and minor roads. Figure 3 shows the view of the intersection specified in Figure 2. The module have some default values for the lane width, turning lane's length, taper length, median width, and pavement marking. The initial configuration of the intersection is stored in an ASCII file called road.def. The user is allowed to change the default configuration before or during simulation. For example, if the user wanted to study the effect of changing the geometry of the intersection on the accident, then he/she is allowed to
change the configuration even during simulation. Our simulator will then change the rendering of the intersection geometry to reflect the new configuration.

![Figure 1: Accident Control](image)

**Figure 1: Accident Control**

**Figure 2: Intersection Geometry Control**

*Car Control*

The Car Control module allows users to set the initial speed, direction, and vehicle position for each vehicle involved as shown in Figure 4. The vehicle speeds are in kilometers per hour (kph). The initial vehicle coordinates (Xpos and Ypos) are measured from the intersection point of the center lines of the roads. The direction field defines the movement direction of the vehicle. For example, a zero direction means that vehicle direction is from left to right (east direction). Information of each vehicle is stored in an ASCII file. The user is also allowed to change the initial configuration before or during simulation.

**Figure 3: Intersection Plan (Orthogonal View)**

*View Control Module*

The View Control Module shown in Figure 5 allows users to see the intersection in different views. It also allows for tracking a specific vehicle (Figure 6). Figure 3 shows an orthographic view. In addition to these two options, there are several other options as shown in Figure 5. The Drive option allows users to visualize accidents from the driver's view as shown in Figure 7. The Orbit view is like the orthographic view but it keeps rotating the scene to allow the user to view it from several angles as shown in Figure 8. The Fly view shown in Figure 9 is also an orthographic view but it tracks ("flies with") the vehicles. It does not only view the center of the intersection as in the orthographic view. In Fly view not only the vehicles are moving but the camera is also moving along.
Figure 4: Car Control Module

Figure 5: View Control Module

Figure 6: Track-Vehicle

View 7: Drive View

Figure 8: Orbit view

Figure 9: Fly View
**Action Control Module**
The Action Control Module provides the user with a simple window to control the accident simulation as shown in Figure 10. The Stop button allows user to stop the simulation run at any time during simulation. The Go button starts the simulation. The Step button runs the accident in step intervals. The Reset button is used to restart the simulation and the Quit button is used to exit the simulator.

![Action Control Module](image)

**Figure 10: Run Control Module**

**Accident Visualization**
The approach adopted in the simulation provides a realistic visualization of vehicle collision. This provides an excellent opportunity to the user to know what exactly happened in an accident. It might help the user reconstruct the precise circumstances of the accident, or investigate how the outcome of an accident may have been different under different circumstances. Figure 11 illustrates a visualization of a collision between two vehicles.

![Collision Visualization](image)

**Figure 11: Collision Visualization**

**Modularity**
The simulator has a modular design that allows for the extensibility of any of the features it currently provides. This feature results from the choice of using an object-oriented programming language, which is C++. As an object programming language, C++ has the advantages that it models real-world objects as software objects. For example, a vehicle is an object, and an intersection is another object. This will result in many benefits. One is that it allows the program to be easily extended to provide more features. Two, is it facilitates the development, especially if many programmers are involved. Three is that it makes the program manageable because it is divided into smaller modules.

**Multi-platform Support**
Our simulator can be used on many platforms. It can be used on a PC, Macintosh, SGI, etc. That is it can run on several operating systems UNIX, Linux, Irix, Windows, Windows NT or MacOS. This gives our simulator its portability feature.

**Arbitrary Number of Vehicles**
Since accidents can be very different in nature, the decision was made in an early stage that the user should have the option to simulate a collision with an arbitrary number of vehicles involved. Apart from the vehicles involved in the collision, the user should also have the option of including other objects in the scene and or in the collision. The objects can include anything from trees or
Vehicle Path Control
The vehicles in the collision simulations should be controlled in one of two ways, either given a predefined route through a scenario file passed to the program or manually through the user interface provided by the application. In both, the scenario file and the user interface, the user can choose to let the car change position, direction or speed.

Different Visualization Features
The user chooses how the collision scene should be rendered through the user interface, and the user can change the rendering method at any point during the execution of the program. The different rendering methods include texture mapping, wire-frame and solid polygonal models. When using the solid model rendering, the user should be able to include lights in the scene and control the position and intensity of the light(s) from the user interface. The user should also be able to choose different camera angles or let the camera track one of the vehicles during the simulation of the collision. The application also includes features to step through a scenario and enable playback of parts or the whole scenario for more accurate analysis of why and how the collision occurred.

The Database Query Mode
The simulator has two modes of execution. One is the animation mode and the other is a database query mode. In the animation mode, the user can view an animation of a scenario that is already known to him/her. That is, the user can use either the default parameters or change them using the graphical user interface. The user in this case can also use the accident report number. In the query mode the user does not know all specifics of a particular accident, but knows some of the circumstances of the accident. For example, the number of involved vehicles, accident type, location, date, weather condition, lighting condition, intersection specifications, etc. The user then can run the program in the query mode and enter some information about the accident he/she is interested in animating. Figure 12 shows the database query interface. The simulator will then search the accident scenario database and will match them with the user’s information. The user is allowed to choose from the matching scenarios which accident he/she wants to visualize. The simulator will animate the user’s selected accident.

Using the report number, we can access the database file. The accident database includes accident-related data such as date, time, accident type, location. The program will make some queries. e.g., one can make a query to animate (1) daytime accidents, (2) accidents in a certain weather condition, and/or (3) accidents that involve a specific number of vehicles.

Figure 12: Database Query Interface

IMPLEMENTATION
Time Based Simulation
A time based, non-real time, modeling approach has been taken in the implementation of the collision simulation. A time based approach was chosen, as opposed to an event based approach, in order to accurately represent the physics of a collision including vehicle body and chassis deformation. A non-real time simulation approach was chosen in order to support computer equipment that is commonly available. A review of the computations involved in the collision physics, and the graphics requirements for realistic three-dimensional visualization, indicated that high end computational and image generation equipment would be required for real time operation. Additionally, real time operation is not required since the user is not attempting to control vehicle movements directly in real time. This also has the advantage that our simulator does not need high-end hardware (the program works on a PC, workstation, or an SGI machine).

Collision Detection
For collision detection, we used the Robust, Accurate Polygon Interference Detection (RAPID) [1] collision detection tool developed at the University of North Carolina. For each modeled object, RAPID creates a data structure from the model coordinates of a list of triangles,
each of which is designated by a unique integer index provided to RAPID along with the coordinates. The interface to RAPID provides an operation to which a client program specifies two such object models, along with a point and a 3x3 transformation matrix which specify each object’s world coordinate position and orientation. The operation merely generates a list of pairs of triangles, one from each object that is in intersection with one another. There is no assumption that the triangles specified in the object models form a continuous surface, or even that they form a surface at all.

RAPID internally creates a hierarchy of oriented bounding boxes which define the extent of modeled objects. It uses these to perform initial rough comparisons between two objects, to determine whether it is necessary to perform the expensive additional processing which compares individual triangle pairs in the objects. Although RAPID is not unique in its use of oriented bounding boxes, other collision detection algorithms exist [2] which use bounding boxes which are aligned to the model coordinate axes. Clearly the use of oriented bounding boxes allows for a tighter fit around an arbitrarily shaped object using smaller boxes. The result is improved due to fewer unneeded comparisons. The authors of RAPID have also developed higher level tools - ICOLLIDE and VCOLLIDE - which use RAPID to perform basic collision detection. These tools perform additional processing, such as avoiding unneeded comparisons and managing a large number of objects, but also place restrictions on their use, such as the assumption that the intersecting triangles do in fact form a convex polyhedron.

It is very important to note here that to enhance the output from RAPID, considering that RAPID does not include occluded triangles – those that are completely inside the other object - in its output. The collision detection of our accident application discovers such triangles by performing a “flood fill” algorithm to identify triangles that are contained within a ring of triangles already reported to have collided. For such triangles, there is no pairing with a triangle from the other object.

Scenario Generation
An important part of our simulation was to reproduce accident scenarios. We had a scenario file for each object involved in the accident. This guarantees the scalability of our model in the sense that it is not limited to a certain number or type of objects. Objects could be vehicles, telephone poles, lampposts, trees, etc. For vehicles, there are two types of scenario generation files.

The first constitutes two parts:

1. The initial condition of the vehicle
2. The event description.

The initial condition includes the initial velocity in km/hr. The initial position of the vehicle \(x, y, z\) where \(x, y, z\) are normalized to be \(z \in [-1, 1]\), the initial value of \(z\)=0. The direction of the vehicle in degrees which is equal to the angle off the North in clockwise direction. The event description included the value and the time the event occurred. There are three possibilities for an event: a driver can accelerate, brake or turn the steering wheel. In our model these events correspond to Gas, Brake and Steering wheel. The value of the Gas means the increment of acceleration from the last speed value. It is normalized to have values between 0 and 1, where 0 means no acceleration and 1 means maximum acceleration. The value of the Brake event means that the driver applied the brakes at this time. It also can have a value between 0 and 1, where 0 means no brake was applied and 1 means full brake was applied. The Steering Wheel event is recorded if the driver turned the steering wheel right or left. It is given values between -1 and +1, where -1 means 90 degrees to the left and +1 means 90 degrees to the right.

In the second type, the scenario file describes a predefined route for every vehicle. That is it specifies at regular intervals of time each vehicle speed, position and direction. It is very important to note here that we allow the user to alter the scenario files through the user interface. Thus these files are not only readable but are also writeable. This provides the user with more flexibility and achieves better results in terms of usability that complies with our goals for that design.

CONCLUSIONS AND RECOMMENDATIONS
A computer simulation tool has been developed. Several elements of this project have been developed, including: visualization and animation capabilities including different view capabilities, collision detection, database search and classification, and geometric characteristics of the site. Other elements of the program has been partially developed, such as accident reconstruction, however, the core of the program is established and full addition of accident reconstruction capabilities is possible in an extension to this work. We also propose adding a learning capability to the program based on artificial Intelligence techniques, so that the model can detect if certain pattern of
accidents tend to occur at certain locations and therefore identify the problems with other similar locations. This would enable any designer to incorporate safety countermeasures in the design process without waiting and experiencing actual accidents and then develop countermeasures. Rather the model will be able to learn from existing locations’ safety and derive what we can expect in other locations. Therefore, the most important extension to this work would the ability of the computer model to learn from accidents that are animated at the different types of intersections, have the ability to generalize and associate specific types of accidents with specific design, location or circumstances. With this addition any designer will be able to detect safety problems at any location easily and efficiently. In this research we have focused on intersections, because more than half of the accidents occur at intersections or at the approach to intersections. It would be simple to extend the model to include other locations (e.g., freeway ramps, etc).

Acknowledgement:

The authors wish to acknowledge the Center for Advanced Transportation Simulation Systems (CATSS) at the University of Central Florida for funding this research.

References