NETWORK-BASED VEHICLE COLLISION DETECTION AND SIMULATION

Liangshou Wu  
Research Assistant  
Virtual Reality Application Center  
Iowa State University  
1620 Hower Hall  
Ames, IA, 50010  
lswu@iastate.edu

James Oliver  
Professor  
Virtual Reality Application Center  
Iowa State University  
1620 Hower Hall  
Ames, Iowa, 50010  
oliver@iastate.edu

Adrian Sannier  
Professor  
Virtual Reality Application Center  
Iowa State University  
1620 Hower Hall  
Ames, Iowa, 50010  
sannier@iastate.edu

ABSTRACT
Vehicle driving simulation, collision detection, and collision simulation of rigid bodies are not new concepts in physical-based simulations, but the integration of all these techniques is a challenging and interesting topic. This paper implements a network-based vehicle collision detection and response simulation system that has all the components that are required by a vehicle driving simulation. It supports vehicle-to-scene and vehicle-to-vehicle collision detection and response simulation in real-time required by a human-in-the-loop driving simulation. Additionally, it supports collaborative driving simulations for multiple vehicles in the same virtual environment operated from different physical locations. A network-based collision server is developed to provide consistent and realistic collision response for vehicles that collide.

Implementation testing shows that the network-based collision simulation can provide a real-time, realistic and robust system in a network with relatively small time lags, such as a LAN or a city-wide network. The implementation has demonstrated it can support simultaneous collision simulation with up to 32 vehicles operating at 150 miles/h in a high speed local area network.

Keywords: Collision detection, Collision response simulation, Virtual reality, Vehicle driving simulation

1. INTRODUCTION
In companies, universities, and research institutes around the world, there are at least one hundred vehicle simulators in use with different levels of capability [1]. Usually, they are used for vehicle design evaluation, driver behavior studies, or traffic safety studies. Real-time human-in-the-loop simulators can give the driver a very “real” feeling by introducing stereo visual, audio, force feedback, and physical motion effects. Vehicle simulators generally do not incorporate collision simulation.

Like most engineering simulations, methods for representing the collision of moving bodies range from highly accurate and hence computationally demanding techniques, such as non-linear finite element impact analyses, to cases with simplistic underlying physics and therefore less computational demand, such as a ball bouncing off a wall in a computer game. Physically accurate vehicle collision or crash evaluation is done offline due to its high computational requirements. Detailed non-real-time vehicle collision simulations have been used for more than forty years by the automobile industry for assessing a vehicle’s safety rating and by legal departments to recreate traffic accidents in order to understand what went wrong.

Meanwhile, with continuing rapid development of computer technology, virtual reality (VR) is now very popular. For many VR applications, simple collision detection is used solely for navigation. Complex collision simulation is used in a few VR applications, generally those designed for specific purposes, such as virtual prototyping or assembly simulation. Many 3D games in the market involve some sort of vehicular interaction (racing, etc.). These generally have some specialized vehicle collision response simulation, but most of them are simplified to maintain real-time performance. Sometimes, the collision response is not reasonable and not consistent for multi-user games, especially for those designed to operate in a distributed networked environment.

In this paper, a vehicle-to-vehicle collision detection and simulation algorithm is developed and implemented in addition to a vehicle-to-scene collision simulation similar to Knight’s [2]. The collision algorithm is a compromise between a non-real-time detailed collision simulation algorithm and very
simple, but often physically unreasonable, vehicle collision simulation algorithm. The result is a real-time, physically reasonable, and realistic collision algorithm. Additionally, it supports the consistent collision responses for multi-vehicle simulation by introducing a network-based collision server, called the collider server. This server eliminates or minimizes the inconsistency of collision results caused in multi-vehicle simulations in which the collision algorithm resides on clients running individual vehicle simulations as well. A new network architecture is developed to implement the collider. The integration and implementation of different simulation components (a vehicle dynamics model, an image generator, a network server, and a collision algorithm) is described as well. The resulting system has all the basic components needed to support a realistic multi-user virtual driving simulator. Figure 1-1 is a screen shot that shows one of the vehicle-to-vehicle collision scenarios.

![Figure 1-1 Front to Rear Side Collision Simulation](image)

2. LITERATURE REVIEW

2.1. Driving Simulation

Since the first human-in-the-loop driving simulator was created by General Motors and Virginia Polytechnic Institute and State University [1], more than one hundred driving simulators have been developed around the world. Different kinds of vehicle driving simulators have the following four basic components in common [1, 3, 4]:

- Real-time simulation of a complex physical system.
- Simulation of environment.
- Control devices.
- Visual and audio systems.

A driving simulator gives users the impression that they are actually driving by simulating changes in the visualization, sound, haptic force feedback and motion in the environment [5,6,7]. But none of the simulators presented in the literature incorporated collision.

With the continual increase in network bandwidth, more recent research is focused on joint driving simulations distributed over a network. Balling, et al. [8] has an example of a collaborative driving simulator that supports a dual-location driving simulation. Based on this collaborative driving simulation Knight moved one step further, simulating vehicle-to-scene collision [2]. The limitation of Knight's approach is that it is hard to incorporate the vehicle-to-vehicle collision simulation. New network architecture is necessary to accommodate both the original and the new functionality.

2.2. Collision Detection and Response Simulation

Collision detection has been extensively pursued in the fields of robotics and computational geometry. Lin and her colleagues presented various collision detection algorithms in [9,10,11]. Most of them can be used for complicated, large-scaled collision detection scenarios. But they are generally unsuitable for real-time simulation, especially for situations in which objects move quickly. The typical simplified collision detection method used mostly in computer graphics and animation is ray segment to convex polygon collision detection [12]. It can be applied in user navigation, ray picking, and simple object collision detection (such as box, sphere, and cylinder).

In computer graphics and animation, there are numerous algorithms for dynamic simulation of rigid bodies [13,14,15,16]. Kawachi, Suzuki and Kimura used impulse friction force to simulate rigid body motion and collision response. They use static and dynamic friction impulse to simulate the friction force, but the static impulse is hard to implement in practice.

Kamal presents a method for computer analysis and simulation of vehicle to barrier impact [17]. Greene describes a method for computer simulation of car-to-car collisions [18]. Both methods are relatively complex, and hence more accurate. They are generally used in the offline simulation of some specific collision scenario, such as vehicle head-to-head collision. These two methods are not suitable for general real-time driving simulation.

2.3. Research Emphasis

The research presented in this paper addresses the needs of multi-user, network distributed driving simulation. The primary challenge is to provide collision detection and realistic post-collision dynamics of vehicles at rates that enable real-time human-in-the-loop interaction.

For vehicle-to-scene collision detection this work takes advantage of the ray-surface intersection functionality of an open source scene graph rendering package called Open Scene Graph [19]. A bounding circle and the Cohen-Sutherland clipping algorithm [20] are implemented for vehicle-to-vehicle collision detection. In this research, a new vehicle-to-vehicle collision algorithm is developed based on the coefficient of restitution model by Macmillan [21]. The new algorithm is improved to provide more realistic and generalized behavior by incorporating a friction force model.

Walter [22] presents a basic networking architecture for multi-vehicle simulation. This research improves on Walter's approach for dealing with vehicle-to-vehicle collision detection and simulation. A new collision server is added to realize consistent response for vehicles detected to have been involved in the same collision.
In addition to a commercial vehicle dynamics simulator called VDANL [23], a simplified vehicle dynamics model (VDM) is implemented according to Gillespie's vehicle dynamics theory [24]. The simplified VDM is extremely useful for testing when multiple versions of the commercial VDM are not available or the frame rate of the commercial VDM is not sufficient.

In reality, the vehicle collision scenarios can vary greatly. This work only focuses on the general cases where the collisions occur between a vehicle body and the scene, and between multiple vehicle bodies. The collision responses are applied in the vehicle's yaw plane. So the roll-over collision cases are not covered in this paper, neither are collisions between tires and the road or curb. All the tests are carried out on flat ground, but there is no limitation that prevents the simulated terrain from being 3D as long as the VDM can handle the texture.

3. NETWORK ARCHITECTURE

A well designed architecture is critically important, not only for ease of maintenance and further development, but also for the effectiveness, robustness and correctness of the resulting application. In this section, the network architecture designed for vehicle-to-vehicle collision simulation is presented.

3.1. Network Architecture

As shown in Figure 3-1, the network architecture is a client-server-based style that consists of the surrounding clients and the central collider server. In addition to functioning as the network server, the collider also behaves as a collision center that handles collision detection and response calculations. The detected collision can be vehicle-to-scene collision or vehicle-to-vehicle collision.

![Figure 3-1 Network Architecture](image)

For each client, there are two subcomponents; the image generator (IG) and the vehicle dynamics model (VDM). Usually, the VDM and IG run on different threads, perhaps even on different machines, and the VDM sits between the IG and the server. The separation of these components is motivated by their different computational requirements. The frame rate of the IG is motivated by a desire for smooth graphics and is typically acceptable down to 10-20 Hz. Accurate dynamic simulation, on the other hand, requires much higher frame rates, ideally on the order of 100 Hz. The separation also enables the VDM to be specialized for complex dynamics calculation. The details about each component's functionality and their network communication are described in [25].

3.2. Architecture Pros and Cons

In most previously proposed systems (e.g., Knight [2]), the collision algorithm typically resides on the client side. It is intuitive to think that the collider should be on the client side because the client generally represents a vehicle. This is reasonable if only one vehicle is involved. However, if the simulation involves multiple vehicles at different places, then problems arise with this straightforward approach.

For example, there are two vehicles running in the same scene, and they are colliding with one another. Both vehicles do the collision calculation separately after the collision is successfully detected. Because of network delay there is no guarantee that both vehicles are able to detect the collision at the same time. The information from the client on the other end of the network is always behind real time. That means both vehicles get different collision responses calculated based on different collision conditions. The level of difference depends on the network delay between these two clients. It is quite possible that Client 1 gets one result as Vehicle 2 hits Vehicle 1, while Client 2 gets the opposite result.

Figure 3-2 depicts this case schematically. On the Client 1 side, Vehicle 2 hits the middle of Vehicle 1 at time t2. But on the Client 2 side, at the same time t2, Vehicle 1 hits the rear part of Vehicle 2. Instead, if we calculate both vehicles' positions based on their velocities from time t1, as shown in the middle area in Figure 3-2, then Vehicle 1 would actually hit the front part of Vehicle 2. This is correct behavior because both packets used for doing the collision calculation are from the same time point, t1.

![Figure 3-2 Problematic Case](image)

The advantage of moving the collider to a shared network resource is obvious. The information used for collision calculation from different clients is more likely to come from the same time point, so it will be more accurate. Moreover, there is only one collider for all the clients. With a single
collider, all collided vehicles get consistent results from the same collision calculation, which allows those vehicles to have consistent responses. Those responses are very important for multi-user simulation. So the collision results calculated by one collider for all clients are more reliable, reasonable and consistent than the ones calculated by many colliders from each client.

But this approach is not perfect. The collider server introduces a network delay that decreases the maximum velocity of vehicles in order to catch and characterize the collisions successfully. The tradeoff between performance and network delay are explored further in section 7.

4. COLLISION DETECTION

In the real world collision is a very common physical phenomena and very easy to understand. For a simple example, consider a pencil dropping onto a desk; the pencil collides with desk, bounces, and then comes to rest on top of the desk. It happens so naturally that we never worry about the pencil penetrating the desk and falling to the floor. For the digital world, things are different.

This section shows how collision happens between a vehicle and the scene and between moving vehicles in a virtual world. It also shows how these collisions can be represented. The goal is to model realistic behavior when a vehicle hits the scene or another vehicle. Although the collision detection algorithms used in this research are not complicated, they build on some basic common ideas of various well-developed algorithms to provide a reasonable compromise between real-time performance and realistic physical behavior.

4.1. Collision model

Collision detection is fundamental in any physical-based simulation. Simulation in a large-scale virtual environment requires collision detection in a large-scale scene with thousands of polygons, and many moving objects. Any collision algorithm needs a specific collision model and query type. The collision model is the representation of objects that are collision candidates. The query type refers to how the algorithm reports collision results.

An actual vehicle is comprised of numerous parts characterized by very complicated geometry. It is impossible for a complex geometric model to be used in real time simulation of collision detection. A simplified collision model that is a rectangle with the same width and length as the real vehicle is used for collision detection in this research. The model also incorporates pitch, roll and yaw angles as well the vehicle cg height and dynamics properties, such as position, velocity and acceleration.

Since the roll-over collision cases are not considered in this research, the simplified 2D rectangle collision model is sufficient for collision detection. The adoption of 2D collision model does not necessarily mean the simulations have to be done on the 2D ground. It can also be done on the 3D terrain since the vehicle model has 3D orientations and translations.

4.2. Vehicle-to-Scene Collision Detection

For vehicle-to-scene collision detection, the vehicle model is decomposed into four line segments. By using OSG's internal ray to surface intersection functionality, vehicle-to-scene collision detection can be done easily and efficiently with four successive intersection tests. Each intersection test returns an intersection point and a normal of intersection surface if the ray intersects any scene surface.

For computing the collision responses; forces, moments, etc, a collision surface normal and a collision point are needed. But the repeated intersection tests for the four edges of vehicle collision model may return zero to four intersection points and surface normals. If there are two or more intersection points, then a single representative intersection point and normal must be computed from them. Figure 4-1 shows the possible cases for vehicle-to-scene collisions.

![Figure 4-1 Vehicle-to-Scene Collision Cases](image)

For cases 3, 4 and 5, which have two intersection points, the mid-point of the two computed intersection points is taken as the collision point. The collision normal is the vector that is perpendicular to the line connecting two intersection points and opposite the approaching velocity vector of the vehicle to the barrier.

A four-intersection-point case, such as Case 6, is transformed into a two-intersection-point case by merging the two closest points. Then the collision point and surface normal are calculated by using the two-intersection-point approach.

4.3. Vehicle-to-Vehicle Collision Detection

In order to reduce unnecessary calculations a two-step collision detection technique is employed in vehicle-to-vehicle collision detection. First, to quickly eliminate cases in which the two vehicles are far apart, a bounding circle proximity filter is implemented. If the testing passes the first step, then the second step is taken. In the second step, the Cohen-Sutherland Clipping Algorithm [20] is used to simplify the intersection test and calculate intersection points. Cohen-Sutherland is a very efficient method for computing the intersection of a line segment with a rectangle. In this research, one vehicle is treated as the rectangle and the other is treated as four line segments. Then the Cohen-Sutherland clipping algorithm is applied four times. If a line segment is clipped, then intersection is detected, and an intersection point list is returned.

Similar to vehicle-to-scene, a single collision point and surface normal are needed for vehicle-to-vehicle collision response calculation. Vehicles collide with each other from any possible angle. For various different cases, different
approaches are taken to yield reasonable collision points and collision surface normals. Figure 4-2 shows the possible cases for vehicle-to-vehicle collision.

![Diagram of Vehicle-to-Vehicle Collision - General Cases](image)

Figure 4-2 Vehicle-to-Vehicle Collision – General Cases

For two-intersection-point cases, such as cases 1, 2, and 3, the same approach used in vehicle-to-scene collision detection is used here to compute the collision point and the impact surface normal. Case 4, although not physically impossible, has one collision point, but almost never happens. The approach for dealing with this case is to wait until two intersection points appear. Case 5, happens only when the time step between two continuous frames is too large. If it happens then the system fails to catch the collision. This condition will be discussed in section 7. It is also possible that the system catches the collision cases like Case 6 which has four intersection points. Similar to the four-intersection-point collision cases for vehicle-to-scene collision, it is transferred into a two-intersection-points case to get the single collision point and surface normal.

5. COLLISION RESPONSE CALCULATION

Based on information obtained from collision detection, this section presents algorithms for determining realistic dynamic collision response in real time. The vehicle-to-vehicle collision response algorithms, which consist of basic dynamics equations and coefficient of restitution, are introduced in the first subsection. The two core models of coefficient of restitution are then explained in detail. Finally the vehicle-to-scene collision response algorithm is described briefly as a specialization of the general vehicle-to-vehicle collision response algorithms.

5.1. Vehicle-to-Vehicle Collision Algorithms

The goal of the collision response algorithm is to make the collision simulation reasonable and correct for a collaborative virtual environment, but not necessarily accurate in engineering detail. As explained in section 4, the vehicle model is simplified as a rectangle consisting of four line segments. Further assumptions are made in this research to simplify vehicle collision simulation, namely:

- There is only one collision at a time for one vehicle.
- Collision force is constant throughout the collision duration and lasts only one frame time.
- Collision forces and moments are applied in the vehicle yaw plane.
- The collision surface normal is in the vehicle yaw plane.
- Table 5-1 shows all the symbols used in deriving the collision response equations. Some of them will be associated with specific vehicles. This association is indicated by adding subscripts and superscripts, as defined below.
- All the variables used take sign.
- The subscript ‘1’ refers to Vehicle 1; ‘2’ refers to Vehicle 2, e.g., \( m_1 \) refers to the mass of Vehicle1, and \( m_2 \) refers to the mass of Vehicle2.
- The subscript ‘x’ indicates the x component; ‘y’ indicates the y component, in the corresponding coordinate system, e.g., \( V_{1x} \) means the “x” component of the velocity of Vehicle 1.
- The superscript ‘p’ means post collision, ‘a’ means pre (ante) collision, e.g., \( V_{2a} \) means y component of velocity of vehicle 2 at the moment before collision.

<table>
<thead>
<tr>
<th>No.</th>
<th>Symbol Name</th>
<th>Unit</th>
<th>Full Name</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>V</td>
<td>Feet/sec</td>
<td>Velocity of vehicle at cg</td>
</tr>
<tr>
<td>2</td>
<td>r</td>
<td>rad/sec</td>
<td>Vehicle yaw rate</td>
</tr>
<tr>
<td>3</td>
<td>p</td>
<td>Feet/sec</td>
<td>Approaching velocity at collision point of two impact bodies</td>
</tr>
<tr>
<td>4</td>
<td>e</td>
<td>N/A</td>
<td>Coefficient of Restitution</td>
</tr>
<tr>
<td>5</td>
<td>( \mu )</td>
<td>N/A</td>
<td>Coefficient of collision friction</td>
</tr>
<tr>
<td>6</td>
<td>m</td>
<td>lbs</td>
<td>Mass of vehicle</td>
</tr>
<tr>
<td>7</td>
<td>( I_{zz} )</td>
<td>lbs\cdot feet</td>
<td>Moment of inertial</td>
</tr>
<tr>
<td>8</td>
<td>I</td>
<td>lbs\cdot sec</td>
<td>Impulse of collision</td>
</tr>
<tr>
<td>9</td>
<td>W</td>
<td>Feet</td>
<td>Width of vehicle</td>
</tr>
<tr>
<td>10</td>
<td>H</td>
<td>Feet</td>
<td>Cg height</td>
</tr>
</tbody>
</table>

In order to avoid the use of the time variable in deriving the collision response algorithms, impulse is used instead of force. Basically, impulse is the integration of force over time. That is \( I = \int F dt \). Since F is assumed constant throughout collision, then \( I = F \times \Delta t \), where \( \Delta t \) is the collision duration. The advantage of using the impulse is that it does not require the collision duration in the collision response calculation to have the same value as the integration time step during vehicle dynamics integration.

5.1.1. Basic Dynamics Properties

If two objects are not totally elastic, then any collision between them causes energy loss. But the total momentum is always maintained throughout the collision.

As shown in Figure 5-1, suppose collision occurs between two vehicles (Vehicle 1 and 2), and the collision point and collision normal vector are all known. \( \rho_1(a_1, b_1) \) and \( \rho_2(a_2, b_2) \) represent vectors from the cg’s of the corresponding vehicles to the collision point. Many other variables are known before a collision. They are: \( m_1, I_{zz_1}, m_2, I_{zz_2}, V_{1a}, V_{1y}, V_{2a}, V_{2y}, \)
The unknown variables are: \( I_{1y}, I_{1y^*}, I_{2x}, I_{2y}, V_{1x}^*, V_{1y}^*, V_{2x}^*, V_{2y}^*, r_1^* \), and \( r_2^* \). 

**Figure 5-1 Vehicle-to-Vehicle Collision**

The linear momentum equations for Vehicle 1 and Vehicle 2 are:

\[
m_1(\vec{v}_1^* - \vec{v}_1) = \vec{I}_i \quad (5-1)
\]

\[
m_2(\vec{v}_2^* - \vec{v}_2) = \vec{I}_i \quad (5-2)
\]

The angular momentum equations for Vehicle 1 and Vehicle 2 are:

\[
l_{1z}(r_1^* - r_1) = \vec{I}_i \times \vec{\rho}_1 \quad (5-3)
\]

\[
l_{2z}(r_2^* - r_2) = \vec{I}_i \times \vec{\rho}_2 \quad (5-4)
\]

The relationship between the impulses applied on two vehicles is:

\[
\vec{I}_2 = \vec{I}_i = I_{1x} + \mu l_{1x} \quad (5-5)
\]

Where \( \mu \) is the coefficient of the vehicle collision friction model that varies with different collision situations (described in the next subsection).

The above equations can be further decomposed into their components.

\[
V_{1x}^* = V_{1x}^* + I_{1x} / m_1 \quad (5-6)
\]

\[
V_{1y}^* = V_{1y}^* + \mu l_{1y} / m_1 \quad (5-7)
\]

\[
r_1^* = r_1^* + (I_{1y}, b_1 - \mu l_{1y}, a_1) / l_{1z} \quad (5-8)
\]

\[
V_{2x}^* = V_{2x}^* - I_{2x} / m_2 \quad (5-9)
\]

\[
V_{2y}^* = V_{2y}^* - \mu l_{2y} / m_2 \quad (5-10)
\]

\[
r_2^* = r_2^* + (-I_{2y}, b_2 + \mu l_{2y}, a_2) / l_{2z} \quad (5-11)
\]

Thus, there are six equations, but seven unknown variables. Another equation, developed in the next subsection, is needed to obtain a solution.

### 5.1.2. Coefficient of Restitution

The coefficient of restitution method is a collision simulation algorithm based on the theory that the separating velocity of two collided objects at the collision point is always less than the approaching velocity at the collision point. Here, the approaching velocity and separating velocity of two objects means the normal component of relative velocity of the two objects.

If \( p \) represents the approaching velocity, then the relationship between pre-collision and post-collision states is:

\[
p^* = -e p^\circ \quad (5-12)
\]

Where \( e \) is coefficient of restitution which generally is positive and less than 1, but in this research, it can be negative in some special cases. These special conditions for \( e \) are described in the next subsection.

As shown in Figure 5-1, using the vehicle velocity and yaw rate to represent \( p \) yields:

\[
p^* = V_{2x}^* - V_{1x}^* + r_2^* b_2 - r_1^* b_1 \quad (5-13)
\]

\[
p^* = V_{2y}^* - V_{1y}^* + r_2^* b_2 - r_1^* b_1 \quad (5-14)
\]

Combining Eq. (5-6) through (5-14) gives:

\[
I_{1x} = \frac{(1 + e)(VP_{2x}^* - VP_{1x}^*)}{M_1 + M_2} \quad (5-15)
\]

Where \( M_1 = \frac{1}{m_1} + \frac{(b_1 - \mu a_1) l_{1z}}{1}, \)

\( M_2 = \frac{1}{m_2} + \frac{(b_2 - \mu a_2) l_{2z}}{1}, \)

\( VP_{1x}^* = V_{1x}^* + r_1^* b_1, \) and \( VP_{2x}^* = V_{2x}^* + r_2^* b_2 \)

In Eq. (5-15), the term \( VP_{2x}^* - VP_{1x}^* \) is actually the approaching velocity of the two vehicles, and the term \( M_1 + M_2 \) is related to these vehicles' mass. So in the final equation it turns out that impulse is related only to approaching velocity and the vehicles' mass. Equation (5-15) is very simple to implement for computing the collision impulse.

### 5.2. Collision Coefficient Models

As mentioned in the previous subsection, two collision coefficients are used for deriving the collision algorithms. One is for the collision friction model, and another is for the coefficient of restitution model. Both of these models are introduced in this subsection.

#### 5.2.1. Collision Friction Model

According to the classic Coulomb friction model, if the relative tangent velocity of two contacted objects is not zero, the friction force between them is proportional to the normal force \( (\mu N) \). If the relative tangent velocity is zero, the friction force is equal to the tangential force that is less than \( \mu N \), where \( \mu \) is the coefficient of static friction.

For vehicle collision simulation, it is impossible to implement static collision friction force by using Coulomb’s friction model since there is no way to get the external
tangential force. The collision calculation is done in one single frame time - no acceleration is involved. So if there is no relative tangent velocity between two objects the friction force is set to zero. However, if there is no static friction force in the collision simulation the vehicle response may be unrealistic. Consider, for example, a head-to-head collision. If there is no relative velocity along the impact surface, the friction force is zero. However, the relative tangent velocity is generally not zero. Even for a very small relative tangent velocity the friction force is \( \mu N \), where \( N \) can be very large. If this happens during simulation, the vehicles spin apart from each other very quickly and the resulting behavior appears unrealistic.

To address this challenge, a new collision friction model is introduced as shown in Figure 5-2. The new model introduces a linear ramp so that small values of approaching angle \( (\alpha) \) yield a relatively small friction force contribution. As shown in Figure 5-2, \( \alpha \) is the angle between relative velocity of the two objects and the collision normal. The relative velocity is the difference of the velocity of Vehicle 2 from the velocity of Vehicle 1. In this research, trial and error testing indicated a reasonable threshold value at which the approach angle ramp stops is 10 degrees. The ramp in this model simulates the tangential impulse. The tangential impulse increases gradually along with the approaching angle.

\[
\begin{align*}
\text{Collision Surface} \\
\text{Absolute Approaching} \\
\text{Relative Velocity}
\end{align*}
\]

\[
0 \quad 10^\circ \quad 90^\circ \quad \text{Angle}
\]

Figure 5-2 Collision Friction Model

5.2.2. Coefficient of Restitution Model

The coefficient of restitution is another coefficient used in the vehicle collision response algorithm. This coefficient directly determines how vehicles behave after collision. Vehicles may bounce back from each other, or go through each other. The response velocity may be large or small compared to the velocity before the collision. All of these behaviors are determined by the coefficient of restitution, \( e \). In this research, the \( e \) value is determined by the approach angle \( (\alpha) \), the relative velocity and the collision point. The approach angle and relative velocity have been defined in the previous subsection.

According to Macmillan [21], typical vehicle collisions have coefficients of restitution of between 0.05 and 0.3. For a small approach angle, the lower value is a better approximation. For a large approach angle, the larger value is better. Based on this, it is useful to set the coefficient of restitution as a function of the approach angle. It is defined here as:

\[
e(\alpha) = 0.175 - 0.125 \cos(2\alpha)
\]  

(5-16)

The model described above is for general collision cases in which the vehicles bounce back after collision. It is also very useful to implement collision cases in which the vehicles pass through each other, as in a glancing blow, when the collisions are not major.

Figure 5-3 shows how such a minor collisions are defined. A central box is defined inside the vehicle. If the relative velocity vector of one vehicle, starting from the collision point, intersects the central box of another vehicle, it is considered a major collision. Otherwise, it is a minor collision. For a minor collision, the \( e \) value is set to a negative value, that way, the two collided objects will not separate from each other. Instead, they will move closer to each other, but at a smaller approach speed.

\[
\begin{align*}
\text{Relative Velocity} \\
\text{Central Box} \\
\text{Collision Point} \\
\text{Vehicle}
\end{align*}
\]

Figure 5-3 Major or Minor Collision

5.3. Vehicle-to-Scene Collision

The coefficient of restitution method is a convenient way to simulate different collision scenarios, such as the impact of completely elastic objects or totally inelastic objects and head-to-head vehicle collisions. In this subsection, it is used for formulating the vehicle-to-scene collision scenario.

Because any scene object is stationary it can be treated as a vehicle whose mass is infinite and velocity is zero. By treating scene object as Vehicle 2, a special case of Eq. (5-15) is (vehicle-to-scene collision response):

\[
M_2 = 0 \text{, and } V_2u = 0 \text{, so } \]

\[
I_{1s} = \frac{-V_1u(1 + e)}{M_1} \]  

(5-17)

5.4. Collision Responses

After computing the normal impulse for Vehicle 1 \( (I_{1s}) \), it is simple to calculate the forces and moment applied at the cg point for each vehicle. Hereinafter, subscript 'i' is used to represent either Vehicle 1 or Vehicle 2. Suppose the integration time step for the VDM is \( \Delta t \), then the forces will be:

\[
F_{ix} = I_{1s} / \Delta t \]  

(5-18)

\[
F_{iy} = I_{2s} / \Delta t = \mu F_{ix} \]  

(5-19)

\[
M_{iz} = (F_{ix} \beta_{iz} - F_{iy} \mu_iz) / \Delta t \]  

(5-20)

All the other post collision state values, such as velocity and yaw rate of both vehicles, can be calculated using Eq. (5-6) though Eq. (5-11). After completing the collision response calculation for all the vehicles, the collider server sends the
6. VEHICLE DYNAMICS IMPLEMENTATION

The vehicle dynamics model (VDM) is a central component of any vehicle driving simulation. The VDM integrates user inputs and the collision packets from the collider server based on a model of vehicle parameters to produce the real time response of the simulated vehicle. The choice of dynamics engine is the central factor in balancing the realism and performance of the simulation. Conceptually, there is no limitation imposed by the other system components on the choice of which VDM to use. In this work, two VDMs have been implemented and tested. They are a simple 7 DOF model and a commercial 17 DOF dynamics simulation package known as Vehicle Dynamics Analysis Nonlinear (VDANL) [23].

VDANL is a commercial simulation software tool produced by Systems Tech, Incorporated. VDANL provides an application programmers interface (API) that facilitates integration of VDANL with a user’s program. The application programmer can influence the simulation frame through means of a user defined module (UDM) that is called by VDANL at a specific time during each simulation frame.

The simplified VDM was written in C++ specifically for the present work. Its architecture is far simpler than VDANL and as a result is much easier to integrate with the other system components. It is desirable to use the simplified VDM for networking and usability testing due to limited availability of the commercial VDM and the high performance needed from the VDM.

The detailed information about the specific communication mechanisms among the VDM, IG, and collider server, and about the internal working of the VDMs is described in [25].

7. USABILITY ANALYSIS AND TESTING

In order to evaluate the usability of the collision detection and simulation scheme, it is important to consider the system’s performance with respect to the response delay. This section addresses usability and the tests of the system performed to explore any limits based on network delay, number of vehicles, and vehicle velocity.

7.1. Usability Analysis

One of the most important considerations of the system is whether collisions are successfully detected when they occur, and if they are caught in a timely manner. In order to give a more specific meaning to these two reference criteria, let’s consider two cases.

First, consider the case of an undetected collision. As shown in Figure 7-1, this happens when the frame rate of the VDM is so low that the gap between subsequent packet updates is too large relative to the velocity of the vehicle. At the \( n^{th} \) frame, the vehicle has not reached the barrier. But at the \( (n+1)^{th} \) frame, the vehicle has completely passed the barrier. In this case, the collision detection algorithm will fail. If this case occurs when vehicle velocity is within expected limits (less than 150 miles/h), then the VDM’s update rate is insufficient to support real time vehicle collision simulation. This outcome is more likely with more complex VDMs since the complexity of the dynamic model is a major factor in frame rate. Since the VDMs used in this research are fast enough, the undetected collision is not a problem in the paper.

In addition to missed collisions, we must also consider collisions that are detected late. Consider Figure 7-2. A collision occurs at time \( t_i \) when the vehicle collides with the barrier. But the vehicle does not receive the collision message until time \( t_2 \) when it has already passed completely through the barrier. We refer to the elapsed time between when the client sends out a packet to the time when it receives a collision packet back as the response delay. Late collisions occur when the response delay is too large relative to the vehicle velocity.

![Figure 7-1 Failed Collision Detection](image)

Figure 7-1 Failed Collision Detection

As shown in Figure 7-2, if \( V \) represents vehicle velocity, and \( \Delta t_{resp} \) represents response delay, then the vehicle travel distance due to response delay can be represented as:

\[
D = \Delta t_{resp} \times V
\]

(7-1)

![Figure 7-2 Late Collision Detection](image)

Figure 7-2 Late Collision Detection

In order to maintain \( D \) at an acceptable threshold, (say less than a half of vehicle width), the response delay and the maximum vehicle velocity must be limited. Basically the response delay includes the network delay, packet manipulation time that depends on number of clients and collision detection time, which also depends on the number of vehicles. Based on this, we can formulate the response delay as Eq. (7-2):

\[
\Delta t_{resp} = \Delta t_{net} + f(NoV)
\]

(7-2)

Where \( \Delta t_{net} \) is network delay, and \( NoV \) represents number of vehicles. Merging Eq. (7-2) and (7-1) gives:
\[ D = (\Delta_{net} + f(NoV)) \times V \]  

(7-3)

Define the reference value for D as half of a vehicle width. If the vehicle travel distance due to response delay is not greater than half of a vehicle’s width, collisions can be detected on time. If we conservatively imagine a standard car width as 5 feet, then the condition under which we can guarantee that collisions can be detected on time is:

\[ (\Delta_{net} + f(NoV)) \times V \leq 2.5 \]  

(7-4)

This analysis is, of course, a simplification in that it presumes that network delay is a constant. A more accurate model of network delay is as a stochastic distribution of times categorized by a mean and variance. If the variance of delay is small relative to the mean, then the above analysis holds. For cases where network delay is highly variable, the maximum value for network delay would provide the limit.

### 7.2. Response Delay vs. Velocity

Both network delay and NoV contribute to the response delay. In this subsection, the relationship between allowed maximum response delay and vehicle speed is carried out based on formula (7-4). As shown in Figure 7-3, the maximum response delay for which we can assure on time collision detection at vehicle speed of 150 miles/h is 12ms. 150 miles/h is a generous maximum speed for all but the most extreme situations involving automobiles.

Figure 7-3 Max Response Delay vs. Speeds

### 7.3. Response Delay in a LAN

Due to the short distance of the 100GB LAN in which the system was tested, the natural network delays were negligible relative to the response time limits shown in Figure 7-3. So for our purposes, the network delays can be ignored compared to the server time contribution to response delay. In other words, in testing the system on the dedicated LAN, server time can be treated as equal to response delay. In this subsection, the relationship between response delay in the LAN and number of vehicles (NoV) will be established based on the testing data shown in Table 7-1.

<table>
<thead>
<tr>
<th>Number of Vehicles</th>
<th>1</th>
<th>2</th>
<th>3</th>
<th>4</th>
<th>5</th>
<th>6</th>
<th>8</th>
</tr>
</thead>
<tbody>
<tr>
<td>Average res. delay (ms)</td>
<td>0.4</td>
<td>1.25</td>
<td>1.5</td>
<td>1.75</td>
<td>2.0</td>
<td>2.4</td>
<td>3.0</td>
</tr>
</tbody>
</table>

Figure 7-4 shows the linear regression line of response delay vs. NoV. If a reasonable maximum velocity of an automobile is considered as 150 miles/h, this allows a maximum response delay of 12ms. Based on Figure 7-4, we can predict the maximum NoV, which can be reliably simulated to be 32, by extrapolating the regression line. The prediction may not be accurate because larger NoV may introduce network traffic that may increase the network delay to significant levels, but it does allow us to say that up to 10 vehicles can be easily simulated on a high-speed LAN with great confidence.

Figure 7-4 Response Delay in LAN vs. Number of Vehicles

If the simulation is carried out on a long distance network, then the network delay would contribute to the response delay significantly. Then the number of vehicles or the maximum vehicle velocity would drop accordingly.

### 8. CONCLUSIONS AND FUTURE WORK

This paper presents the design, implementation and evaluation of a network-based vehicle collision detection and simulation system. It supports multi-vehicle simulation in multiple places on a network. It supports vehicle-to-scene and vehicle-to-vehicle collision detection and response simulation. Additionally, it supports connection of heterogeneous vehicle dynamics models and image generators.

The OSG built-in hierarchy bounding volume and line segment to surface collision detection functionalities were used for vehicle-to-scene collision detection. This functionality was easily applicable and proved quite efficient for the vehicle collision model used in this paper. Two stages of collision detection, bounding circle and Cohen-Sutherland clipping, were utilized to facilitate efficient vehicle-to-vehicle collision detection.

A simple, but robust and realistic vehicle-to-vehicle, including vehicle-to-scene, collision simulation algorithm was developed, based on the coefficient of restitution method presented by Macmillan. A friction model was added to simulate the static friction force. An adapted coefficient of restitution model was created to extend the functionality of the coefficient of restitution method so that it could be used for simulating a glancing collision scenario in addition to a receding scenario.

A new collision server was introduced, together with a network architecture to accommodate vehicle-to-vehicle collision detection and response simulation. The network-based collision algorithm presented is capable of reproducing the collision behavior of two colliding vehicles. More
importantly, it supplies consistent collision results for each collided vehicle based on the same collision. So vehicle responses in vehicle-to-vehicle collision are consistent, reasonable and realistic.

Test and analysis showed that the network-based vehicle collision detection and simulation system was capable of simulating up to 32 vehicles simultaneously on a dedicated 100Gb LAN with vehicle speeds of up to 150 miles/h. These results would be substantially reduced in the presence of network delays where the variance of the delay is significant in comparison to the mean delay.

For the future work, the current vehicle collision simulation algorithms can be extended to incorporate the vertical collision forces to simulate more vehicle collision scenarios such as roll-over collision simulation. The 3D collision model, instead of 2D collision model, would be necessary for doing those collision simulations with vertical collision responses in order for the system to receive more collision queries.

REFERENCES