DSC North America 2003 Proceedings, Dearborn, Michigan, October 8-10, 2003 (ISSN 1546-5071).

SIMULATION OF VEHICLE COLLISIONS IN REAL TIME

Mark Knight
Los Alamos National Laboratory
MS C927
Los Alamos, NM 87545
Phone: 505-667-6378
Email: mknight@lanl.gov

Jim Bernard
Virtual Reality Applications Center, Iowa State University
2274 Howe Hall, Room 1620
Ames, IA 50011-2274
Phone: 515-294-3092
Fax: 515-294-5530
Email: bernard@iastate.edu

October 8, 2003

ABSTRACT

Typical vehicle simulations require numerical integration at an integration time step no larger than 0.01 seconds, usually less than half that time. This does not leave enough time to carry out the complex calculations required for detailed collision calculations in real time. This paper presents a method that strikes a compromise, which, although not carrying all the detail necessary for very accurate collision calculations, allows useful simulations to proceed in real time. The method has three parts: collision detection, estimation of the momentum transfer expected to result from the collision, and application of forces to provide the desired momentum transfer. The method uses a common scene graph for collision detection, which allows the system to work with most of the common scene database formats without the need of specialized preprocessing. All of the collision detection and response calculations employ open-source code and are designed to work well at speeds required by real-time vehicle simulation. Examples based on the VDANL vehicle dynamics simulation illustrate the utility of the methodology.

INTRODUCTION

Our interest in collision in real time follows from our interest in driving simulation, which demands real time calculations. References [1,2,3,4] present descriptions of a several driving simulators.

Reference [5] describes four components common to all driving simulators:

- A simulation of the physics of the vehicle model and the road surface
- A simulation of the surrounding environment
- Video and audio displays to display state output to the operator
- Input control devices for the operators

Reference [6] discusses these components of vehicle simulation and adds two additional components, collision interaction and networking management, for a collaborative driving simulation application.

This paper focuses on simulating vehicle collisions in real time. The motivation is to allow driving simulations to continue in the face of glancing impacts with barriers rather than distract the user by going through the barrier or by causing the simulation to stop. There are two key challenges, detecting that a collision has occurred, and computing the effects of the collision.

BACKGROUND

Detailed collision simulation has been an active area of research since the 1960's. Various lumped mass spring models were presented in the early 1970's [7,8] as a method to evaluate crashworthiness of vehicles in a more cost effective way. In 1973 McHenry [9] presented the Simulation Model of Automobile Collisions (SMAC) computer program as a tool for accident reconstruction. All of these methods were intended to simulate somewhat detailed collision events, and in the accident reconstruction cases, were often used in an iterative way to match physical evidence. They were not concerned with real time performance.

In 1983 Macmillan [10] presented rigid body impulse response calculations specifically for vehicle collisions. The method assumed the pre and post collision velocities at the impact point were governed by a coefficient of restitution. This method of calculation is appealing for real-time simulation because of the simplicity and speed of the calculations.

Hahn [11] in 1988 presented rigid body impulse response calculations for more general rigid bodies in computer animations. About that same time Moore and Wilhelms [12] discussed the topics of collision detection and collision response. They presented two response methods, a spring based penalty method and an impulse based solution. The impulse method was typically faster to compute, especially in violent collisions, and had an added benefit in that the resulting system of equations need only be solved once per collision instead of every time step as required by the spring based methods.

This paper implements the impulse methods given by Macmillan and presents a method based on the loss of kinetic energy during the collision. Both these response methods give reasonable looking results in real-time. While this paper focuses on collision response calculations, it will also demonstrate that the real-time performance is highly dependent on the collision detection algorithm speed. Lin [13], Jiménez [14], and Kim [15] have provided recent surveys of collision detection methods.

The following sections address the issues of real-time collision detection and response and address the challenge of real time implementation.

COLLISION DETECTION

Typically collisions are detected by searching a database of collidable objects to find the interference. If a collision is detected, the collision point and collision plane normal are saved to enable calculation of the resulting response.

There are several ways to perform the detection operation [13, 14, 15]. We selected Open Scene Graph [17] to perform collision detection against a visual scene graph database. This option provides a great deal of flexibility in database formats while still maintaining ample computation speed. In addition, Open Scene Graph is free, open source, and can run on several computer platforms. Other methods may have faster detection speeds, particularly with databases containing a high number of polygons, but often they require specialized file formats and substantial preprocessing [15].

To test for a collision against the scene graph, the vehicle is represented as a set of line segments that generate a horizontal 2D rectangular plate around the vehicle at the CG height. For each integration time step, the bounding rectangle is tested against the scene for intersections. Figure 1 shows the front-right corner of the bounding rectangle intersecting with a wall. A collision is detected when any segment intersects an object.

To enable calculation of the vehicle response to the collision, simple algorithms typically require specification of the point of force application and the direction of the force. Here we find the point of application of the force by taking the midpoint of the line of interference between the two objects. As an example, in Figure 2 the point of application would be the midpoint of the line segment a-b. Since we are concerned in this paper with flexible vehicle bodies hitting rigid barriers, the collision plane normal will be the surface normal of the rigid barrier.

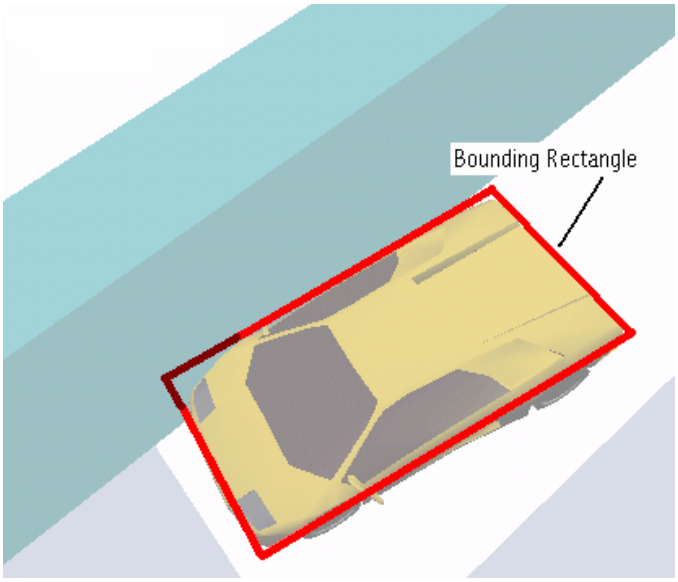

FIGURE 1 Vehicle collision bounding rectangle in a front right side collision.

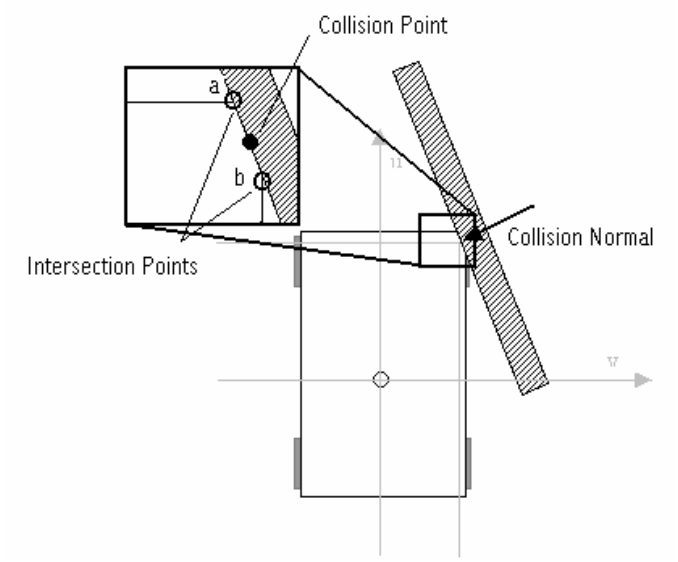

FIGURE 2 Average intersection point and collision normal. The callout shows how the two segment intersection points are averaged to find the final collision point. The collision normal vector is shown perpendicular to the rigid barrier.

COLLISION RESPONSE

Our goal is to simulate collisions in a useful way that makes sense but is not intended to be correct in engineering detail. Several simplifying assumptions enable real time calculations:

1. The impact is between a vehicle and a rigid wall.
2. The normal vector to the wall is in the yaw plane of the vehicle.
3. The forces of impact are in the yaw plane and at the mass center height of the vehicle.

4. The time duration of the impact is small compared to the time scale of yaw plane vehicle motion.

5. There is only one impact force, and this impact force remains fixed relative to the vehicle throughout the impact.

This section presents two methods for computing the collision response. Both methods begin with the application of momentum relationships.

Linear and Angular Momentum

First apply linear momentum:

$$\int_{t_0}^{t_0+\Delta t} \mathbf{F} dt = m\left(\mathbf{V_2} - \mathbf{V_1}\right) \qquad (1)$$

where the collision occurs at t_o, the duration of the collision is Δt, **F** is the force vector applied to the vehicle by the wall, m is the mass of the vehicle, $\mathbf{V_1}$ is the yaw plane velocity vector of the vehicle's mass center just before the force is applied, and $\mathbf{V_2}$ is the yaw plane velocity vector of the vehicle's mass center just after the force is applied. We presume the time interval Δt is so small that other forces, forces from the road on the tires for example, do not affect the momentum transfer. We have restricted our analysis here to one impact force. The extension to more impact forces, at least from a momentum point of view, is straightforward.

A similar relationship applies to angular momentum, namely,

$$\boldsymbol{\rho} \times \int_{t_0}^{t_0+\Delta t} \mathbf{F} dt = I_{zz}\left(r_2 - r_1\right) \qquad (2)$$

where I_{zz} is the vehicle's total yaw moment of inertia about its mass center, $\boldsymbol{\rho}$ is the vector from the total vehicle mass center to the impact, and r_1 and r_2 are the vehicle's yaw rates before and after the collision respectively.

Equations (1) and (2) have a total of three unknowns, the velocity of the mass center after the impact, the angular velocity after the impact, and the impulse. For the third equation needed to solve the system, we rely on a combination of intuition and experience.

Coefficient of Restitution Method

First consider the methods of Macmillan [10], who calls for the normal velocity change of the vehicle at the point of impact to be a function of a coefficient of restitution e.

$$\mathbf{Vp_2} \cdot \mathbf{N} = -e \cdot \left(\mathbf{Vp_1} \cdot \mathbf{N}\right) \qquad (3)$$

$\mathbf{Vp_1}$ and $\mathbf{Vp_2}$ are the velocity vectors of the vehicle at the point of impact before and after the impact and **N** is the unit normal to the rigid surface.

The coefficient of restitution e is a function of the details of the collision. Macmillan suggests values in the range of 0.0 to 0.3. Equations (1), (2), and (3) yield the impulse and the post collision angular velocity and velocity of the mass center.

The algorithm implementation defines the collision frame of reference about the point of collision with the primary force along the surface normal of the collision and a frictional force along the tangent direction. Macmillan's presentation illustrates the general two-vehicle collision, but the presentation here is limited to a single vehicle hitting a static rigid body.

Given the vehicle velocity vector projected into the collision normal and tangential directions, the following momentum equations relate the pre and post collision velocities with the impulse.

$$m(vn_2 - vn_1) = \text{Impulse} \tag{4}$$

$$m(vt_2 - vt_1) = \mu \cdot \text{Impulse} \tag{5}$$

Where *vn* is the mass center velocity normal to the collision surface, *vt* is mass center velocity tangential to the collision surface, μ is the coefficient of friction, and Impulse is normal to the collision surface,

$$\text{Impulse} = \int_{t_0}^{t_0+\Delta t} F_n dt \tag{6}$$

The change in angular momentum is given by Equation (2), which expands to the form,

$$I_{zz} \cdot (r_2 - r_1) = \int_{t_0}^{t_0+\Delta t} F_n dt \cdot \mu \cdot x - \int_{t_0}^{t_0+\Delta t} F_n dt \cdot y \tag{7}$$

Combining Equations (6) and (7) gives the form,

$$I_{zz} \cdot (r_2 - r_1) = \text{Impulse} \cdot (\mu \cdot x - y) \tag{8}$$

Where r is the yaw rate and x and y are the distances from the total vehicle mass center to the point of collision as illustrated in Figure 3.

The pre- and post-collision velocities are related by the coefficient of restitution as stated in Equation (3). To simplify the notation we use *p* to represent *Vp·N*, which is the velocity at the point of impact and normal to the collision surface.

$$p_2 = -e \cdot p_1 \tag{9}$$

It is clear from Figure 3 that

$$p_1 = vn_1 - r_1 \cdot y \tag{10}$$

$$p_2 = vn_2 - r_2 \cdot y \tag{11}$$

where y is the moment arm from the vehicle's mass center to the collision normal.

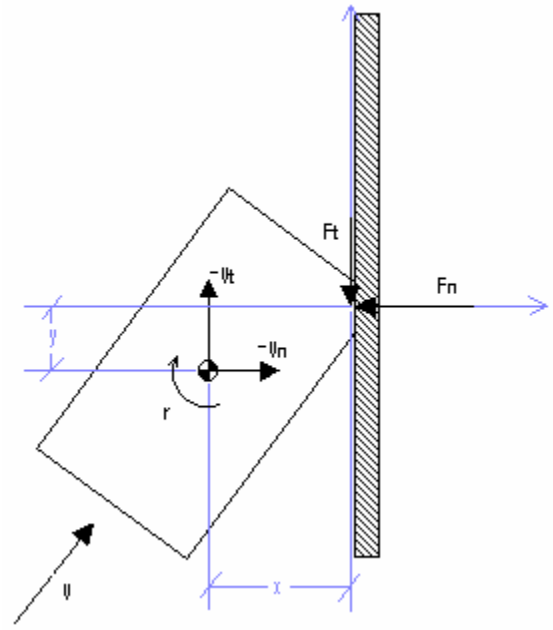

FIGURE 3 Restitution method collision diagram.

According to Macmillan, typical vehicle collisions have coefficients of restitution of between 0.0 and 0.3. Since we need to be prepared for a wide range of collision angles, we have found it useful to use a cosine shaped function, as shown in Figure 4, to interpolate the coefficient of restitution as a function of the angle of attack between the vehicle and the wall, from 0.3 for small angles to 0.05 at a 90-degree impact. When the angle of attack is small, a coefficient of 0.3 yields a small loss in kinetic energy and the vehicle is simply redirected away from the wall. For collisions near ninety degrees angle of attack, the coefficient of 0.05 leads to a loss of almost all of the kinetic energy, an expected result of a head on collision.

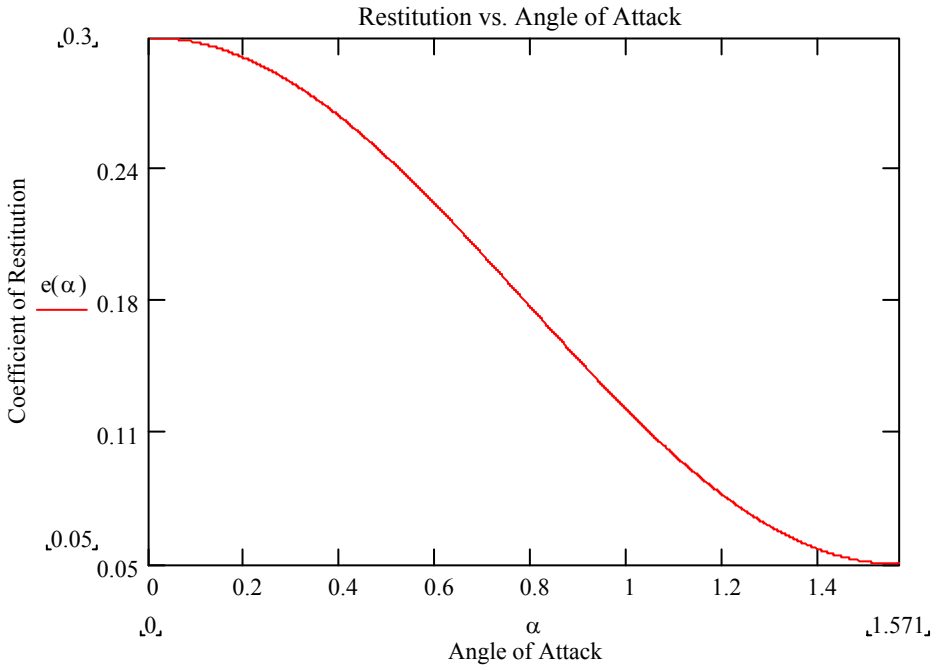

FIGURE 4 Coefficient of Restitution vs. Angle of Attack

The coefficient of restitution relationship and momentum equations yield the impulse. Substituting Equation (11) into Equation (9) yields:

$$vn_2 - r_2 \cdot y = -e \cdot p_1 \tag{12}$$

where, p_1 is computed from Equation (10) and the pre-collision velocities.

Using the momentum equations, rearranging and substituting them into Equation (12) yields

$$\frac{m \cdot vn_1 + \text{Impulse}}{m} - y \cdot \left[r_1 + \frac{\text{Impulse} \cdot (\mu \cdot x - y)}{I_{zz}} \right] = -e \cdot p_1 \tag{13}$$

Simplifying,

$$p_1 + \text{Impulse} \cdot \left[\left(\frac{1}{m} + \frac{y^2}{I_{zz}} \right) - \mu \left(\frac{x \cdot y}{I_{zz}} \right) \right] = -e \cdot p_1 \tag{14}$$

Now, if,

$$h = \frac{1}{m} + \frac{y^2}{I_{zz}} \tag{15}$$

and,

$$k = \frac{x \cdot y}{I_{zz}} \tag{16}$$

then the impulse is

$$\text{Impulse} = -\frac{1+e}{h - \mu \cdot k} \cdot p_1 \tag{17}$$

Kinetic Energy Loss Method

For certain special applications, going through a barrier for example, we have found it useful to stipulate the energy loss during the collision rather than call for a coefficient of restitution. In particular, consider the parameter P that indicates the yaw plane energy before and after the collision.

$$P = \frac{m(\mathbf{V_2} \cdot \mathbf{V_2}) + I_{zz} \cdot r_2^{\,2}}{m(\mathbf{V_1} \cdot \mathbf{V_1}) + I_{zz} \cdot r_1^{\,2}} \tag{18}$$

Again, we find it useful to make P a cosine function of the angle of attack between the vehicle and a rigid wall, ranging from 0.92 at small angles of attack to 0.04 at 90-degree impacts. These energy loss values appear to generate reasonable responses.

The collision force can be broken into components along the vehicle's local coordinate system.

$$\mathbf{F} = F_0(a\mathbf{i} + b\mathbf{j}) \tag{19}$$

where **i** and **j** are unit vectors in the vehicle's coordinate system as illustrated in Figure 5 and F_0 is the magnitude of the force.

Following the procedure of the restitution method, we apply linear momentum for the system. Referring to Figure 5, Equation (1) can be written in scalar form.

$$F_0 \cdot a = \frac{m(u_2 - u_1)}{\Delta t} \tag{20}$$

$$F_0 \cdot b = \frac{m(v_2 - v_1)}{\Delta t} \tag{21}$$

where u and v are the longitudinal and lateral components of the mass center velocity.

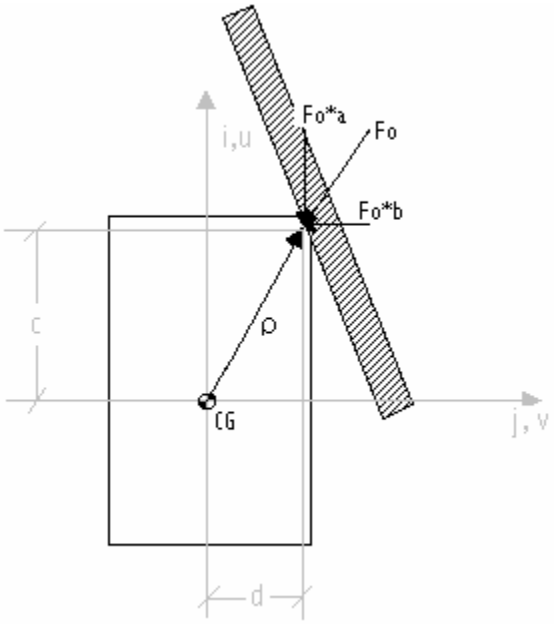

FIGURE 5 Collision with a static barrier on the front right corner of the vehicle.

Similarly, the change in angular momentum about the mass center given by Equation (2) can be computed in scalar form as

$$F_0(-a \cdot d + b \cdot c) = \frac{I_{zz}(r_2 - r_1)}{\Delta t} \tag{22}$$

where r is the yaw rate, I_{zz} is the yaw moment of inertia, and the vector from CG to collision point is

$$\boldsymbol{\rho} = (c\mathbf{i} + d\mathbf{j}) \tag{23}$$

The relationship between pre- and post-collision kinetic energy is given by

$$E_2 = P \cdot E_1 \tag{24}$$

where, P is the fraction of energy remaining after the collision and,

$$E_1 = 0.5 \cdot m\left(u_1^{\,2} + v_1^{\,2}\right) + 0.5 \cdot I_{zz} \cdot r_1^{\,2} \tag{25}$$

$$E_2 = 0.5 \cdot m\left(u_2^{\,2} + v_2^{\,2}\right) + 0.5 \cdot I_{zz} \cdot r_2^{\,2} \tag{26}$$

Equations (20, 21, and 22 through 26) yield the magnitude of the force F_0 required to achieve the impulse across the specified collision time step Δt.

Begin by relating post-collision velocity and yaw rate to the magnitude of the force of impact:

$$u_2 = \frac{F_0 \cdot a \cdot \Delta t + m \cdot u_1}{m} \tag{27}$$

$$v_2 = \frac{F_0 \cdot b \cdot \Delta t + m \cdot v_1}{m} \tag{28}$$

$$r_2 = \frac{F_0 \cdot \Delta t \cdot b \cdot c - F_0 \cdot \Delta t \cdot a \cdot d + I_{zz} \cdot r_1}{I_{zz}} \tag{29}$$

Now Equation (26) yields

$$\begin{aligned} \frac{E_2}{0.5} &= \frac{m\left(F_0^{\,2} \cdot \Delta t^2 \cdot a^2 + 2 \cdot F_0 \cdot \Delta t \cdot a \cdot m \cdot u_1 + m^2 \cdot u_1^{\,2}\right)}{m^2} + \ldots \\ \ldots &+ \frac{m\left(F_0^{\,2} \cdot b^2 \cdot \Delta t^2 + 2 \cdot F_0 \cdot b \cdot \Delta t \cdot m \cdot v_1 + m^2 \cdot v_1^{\,2}\right)}{m^2} + I_{zz} \cdot r_2^{\,2} \end{aligned} \tag{30}$$

which can be rewritten as

$$\begin{aligned} \frac{E_2}{0.5} &= F_0^{\,2}\left[\frac{\Delta t^2\left(a^2 + b^2\right)}{m} + \frac{\Delta t^2\left(a^2 \cdot d^2 + b^2 \cdot c^2 - 2 \cdot a \cdot d \cdot b \cdot c\right)}{I_{zz}}\right] + \ldots \\ \ldots &+ F_0\left[2 \cdot \Delta t \cdot \left(a \cdot u_1 + b \cdot v_1\right) + 2 \cdot \Delta t \cdot r_1 \cdot \left(b \cdot c - a \cdot d\right)\right] + \ldots \\ \ldots &+ m\left(u_1^{\,2} + v_1^{\,2}\right) + I_{zz} \cdot r_1^{\,2} \end{aligned} \tag{31}$$

The solution for F_0 is:

$$F_0 = \frac{-B \pm \sqrt{B^2 - 4 \cdot A \cdot C}}{2A} \tag{32}$$

Where:

$$A = \frac{\Delta t^2\left(a^2 + b^2\right)}{m} + \frac{\Delta t^2\left(a^2 \cdot d^2 + b^2 \cdot c^2 - 2 \cdot a \cdot d \cdot b \cdot c\right)}{I_{zz}} \tag{33}$$

$$B = 2 \cdot \Delta t\left(a \cdot u_1 + b \cdot v_1\right) + 2 \cdot \Delta t \cdot r_1\left(b \cdot c - a \cdot d\right) \tag{34}$$

$$C = m\left(u_1^{\,2} + v_1^{\,2}\right) + I_{zz} r_1^{\,2} - \frac{E_2}{0.5} \tag{35}$$

This method yields two solutions. We typically implement the solution with the larger of the two forces because it prevents the vehicle from going through the barrier. Note that the lower of the two forces may also be valuable if we want the vehicle to lose energy while breaking through the barrier.

DYNAMICS IMPLEMENTATION

The methods presented here can be used with any vehicle dynamics platform. There are two choices for implementation:

- Upon collision detection, calculate the momentum change using the methods presented here, reset the state variables of the simulation with the post collision velocity and yaw rate, and continue integration.
- Upon collision detection, calculate the momentum change using the methods presented here. Then apply a force over a short time period to cause the desired momentum change.

The first method, resetting the state variables, is only suitable for modelers who have control of the vehicle dynamics software. It is not very practical for modelers using commercial code because in the context of such code the modeler does not typically have the ability to overwrite the state variables and restart the motion quickly enough to meet real time constraints. This paper uses the second method, namely, applying the force required to cause the desired momentum change.

We implemented both the coefficient of restitution method from [10] and the energy method given here in the commercial software VDANL [16]. We assumed a constant force between the rigid barrier and the vehicle for a small period of time to yield the desired momentum change, and we used VDANL's User Defined Module option to apply the collision forces and moments to the vehicle model.

For both methods, the force of impact included a force normal to the surface of collision and a frictional force tangent to the surface. We used a coefficient of friction μ to define the friction force as

$$F_t = \mu \cdot F_n \tag{36}$$

Following Macmillan, who suggested coefficient of friction values from 0.0-0.3 for typical collisions, we found it useful to vary the coefficient of friction from 0.0 for head on collisions to 0.3 for small angles of attack. We arrived at this approach on pragmatic grounds – the coefficient 0.3 makes good sense for small angles of attack and seems to work well in the models. At high angles of attack, near-head-on collisions that characteristically lead to very high normal forces, we lowered the magnitude of the friction coefficient to prevent unrealistically high post-collision yaw rates.

Initialization

Prior to starting the integration, the user must initialize the collision scene graph and supply vehicle parameters for the collision testing and response calculations. The collision scene file can be any visual database format supported by Open Scene Graph [17].

The following parameters are required:

- Vehicle mass
- Vehicle yaw moment of inertia
- Distance from the CG to the front of the vehicle body
- Distance from the CG to the rear of the vehicle body
- Vehicle body width
- CG height

These dimensions are used to generate a 2D bounding rectangle, which is set at the CG height.

Collision Testing and Force Computation

For each dynamics time step the simulation provides the collision algorithm with the simulation time, linear position and velocity, and angular position and velocity of the vehicle. This places the bounding rectangle in the scene to test for interference with any of the scene's elements. If a collision is detected, the collision point and collision surface normal are stored to compute the forces and moments for application to the sprung mass.

Application of Forces and Moments

Equations (1) and (2) plus equation (3) or (18) enable the calculation of the impulse $\int \mathbf{F} dt$ and the angular impulse $\boldsymbol{\rho} \times \int \mathbf{F} dt$. We have implemented these impulses in the context of VDANL by assuming the force F is constant. Thus,

$$\int_{t_0}^{t_0+\Delta t} \mathbf{F} dt = \mathbf{F} \cdot \Delta t \qquad (37)$$

We verified that the fixed step integrator of VDANL provides accurate results with one integration time step,

$$\Delta t = 0.005 \text{ seconds} \qquad (38)$$

The performance testing focused on the two primary objectives, results that look reasonable and follow rigid body momentum and energy relationships correctly, and algorithms that work in real time.

TESTING FOR FACE VALIDITY

We tested the algorithms in several scenarios, from oblique side impacts to head on collisions, and the performance in all cases was satisfactory in the sense that the results seemed to make good sense.

Figures 6 and 7 provide an illustration, which was carried out using the database of Figure 8 and the coefficient of restitution-based calculations. As Figure 6 indicates, the vehicle begins driving straight ahead; the velocity is 60 mph. The right front corner impacts the side wall at about 1.5 seconds, causing a spike in negative yaw rate. This is followed almost immediately by the right rear corner of the vehicle hitting the wall, reversing the direction of the yaw rate, but with the new velocity of the mass center directed slightly away from the wall. At about 2.5 seconds, the left front corner of the vehicle hits the end wall effectively stopping forward motion and resulting in residual negative yaw rate, which gradually dies out.

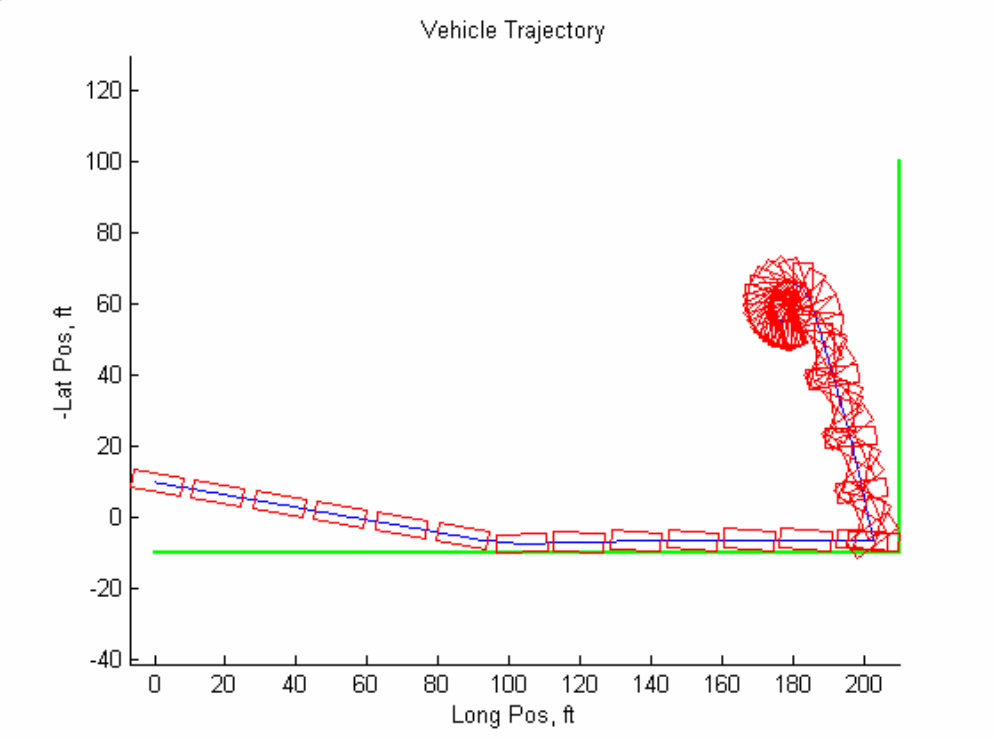

FIGURE 6 Vehicle trajectory of a side collision followed by a front collision.

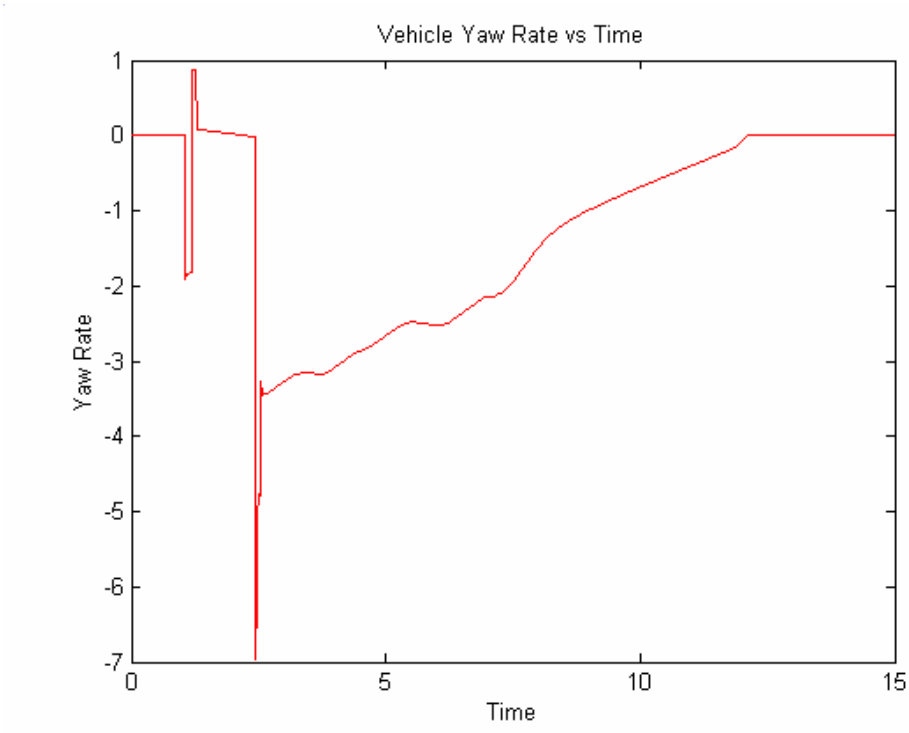

FIGURE 7 Vehicle yaw rate for the side collision followed by a front collision.

TESTING FOR REAL TIME PERFORMANCE

We tested the real-time performance of the algorithm using two databases, one quite simple and one a bit more complex. The complexity of a database is primarily measured by polygon count. Figure 8 presents the simple database, a test scene containing an L-shaped wall and a flat driving surface. The database contains 612 triangles, all of which were used to test for collisions. Figure 9 presents part of the more complicated Watkins Glen racetrack scene. The entire visual scene consists of 8780 triangles, but we created a version of the database with only the vertical elements for collision testing. This removed unnecessary terrain and sky polygons and brought the collision triangle count to 1838.

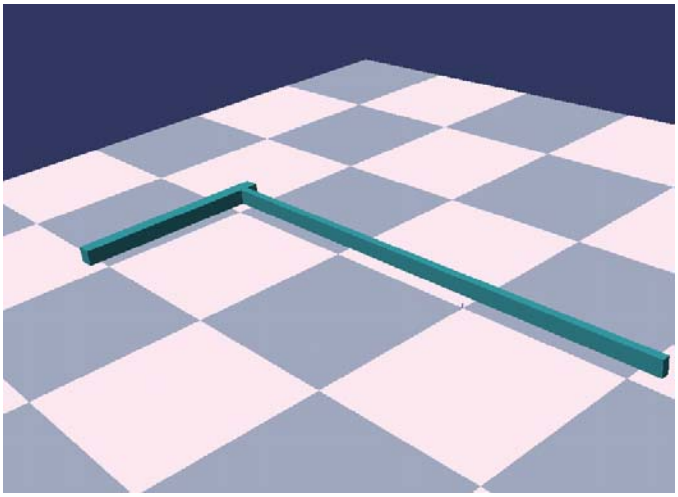

FIGURE 8 L-shaped test scene.

Numerical experiments verified our expectation that the collision computation speed is mainly dependent on collision scene complexity. All of the experiments were run on a 300 Mhz Pentium II PC with 192 MB of RAM. Both the restitution and kinetic energy based methods yield fast collision response computation times averaging 0.013 ms per integration time step independent of the complexity of the database. This shows the response calculation time is negligible as it is only 0.26% of the VDANL integration time step of 5 ms.

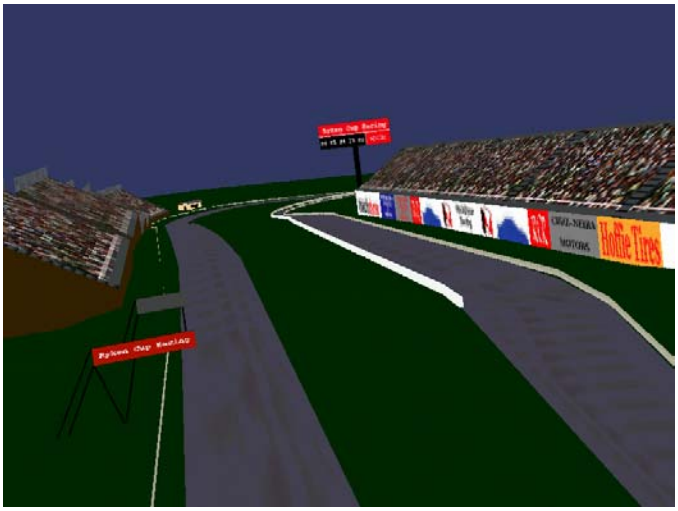

FIGURE 9 Watkins Glen Track

The collision detection calculations and numerical integration can take the remainder of the 5 ms time step. Table 1 presents the average update times for the numerical experiments. These update times include the normal vehicle dynamics calculations as well as all collision detection and response calculations. Since the dynamics and collision response calculation speeds are relatively constant, this table presents a good measure of relative detection speeds of different scenes. The table indicates that the more complicated scene requires higher query times and thus longer update times.

Both scenes were created with Multigen Creator, which can optimize a scene graph hierarchy to improve the collision detection speed [18]. Table 1 shows average VDANL update times for typical collisions with both scenes in a non-optimized state. The table also provides results for the Watkins Glen database after Multigen Creator optimization, which reorganizes the polygons spatially so collisions are detected more quickly by only querying polygons in close proximity to the vehicle. Optimizing the Watkins Glen database improved the overall performance by over 15%.

TABLE 1 VDANL update method times using different collision scenes.

	Kinetic Energy Method	Coefficient of Restitution Method
Test Rail	2.52 ms	2.60 ms
Watkins Glen	4.69 ms	4.64 ms
Watkins Glen Optimized by MultiGen Creator	3.88 ms	3.90 ms
No Collision Calculation	0.99 ms	

CONCLUSIONS AND FUTURE WORK

This paper presents an algorithm that enables real-time collisions in driving simulations. The algorithm includes collision detection, computation of the resulting impulse, and the application of forces and moments to a vehicle dynamics model. The VDANL based implementation illustrates real-time collision in simple and moderately complex databases.

In the future we plan to extend the library to include collisions with moving objects. We are primarily interested in vehicle-to-vehicle collisions in collaborative driving simulations. Modification of this code to allow for moving collidable bodies will be challenging, particularly when implemented for vehicles simulated on different dynamics engines but interacting in the same environment. We expect continued improvement in computation speed will help in the extension of this work to include additional vehicles and more complicated databases.

ACKNOWLEDGMENTS

Thanks to Ole Balling and Systems Technology Inc. for their help with VDANL and sample library code. The Watkins Glen database is courtesy of Fakespace Systems Inc.

REFERENCES

1. Romano, R.A., Stoner, J.W., Evans, D.F., "Real Time Vehicle Dynamics Simulation: Enabling Tool for Fundamental Human Factors Research", SAE Paper 910237, 1991.

2. Greenberg, J.A., Park, T.J., "The Ford Driving Simulator", SAE Paper 940176, 1994.

3. Bertollini, G.P., et al., "The General Motors Driving Simulator", SAE Paper 940179, 1994.

4. Chen, L.D., Papelis, Y., Watson, G., Solis, D., "NADS at the University of Iowa: A Tool for Driving Safety Research", Paper presented at the 1st Human-Centered Transportation Simulation Conference, Iowa City, IA, 2001.

5. Gruening, J., Bernard, J., Clover, C., Hoffmeister, K. "Driving Simulation", SAE Paper 980223, 1998.

6. Balling, O., Knight, M., Walter, B., Sannier, A. "Collaborative Driving Simulation", SAE Paper 2002-01-1222, 2002.

7. Kamal, M.M, "Analysis and Simulation of Vehicle to Barrier Impact", SAE Paper 700414, 1970.

8. Greene, J.E., "Computer Simulation of Car-To-Car Collisions", SAE Paper 770015, 1977.

9. McHenry, R.R., "Computer Program for Reconstruction of Highway Accidents", SAE Paper 730980, 1973.

10. Macmillan, R. H., "Dynamics of Vehicle Collisions", Channel Islands, UK: Inderscience Enterprises Ltd., 1983.

11. Hahn, J.K., "Realistic Animation of Rigid Bodies", Computer Graphics, Volume 22, Number 4, 1988.

12. Moore, M. and Wilhelms, J., "Collision Detection and Response for Computer Animation", Computer Graphics, Volume 22, Number 4, 1988.

13. Lin, C.M., Gottschalk, S., "Collision Detection Between Geometric Models: A Survey", University of North Carolina, 1998.

14. Jiménez, P., Thomas, F., and Torras C., "3D Collision Detection: A Survey", Institut de Robòtica i Informàtica Industrial, Barcelona, Spain, 2000.

15. Kim, C., "Collision Detection Algorithms", Virtual Reality Applications Center, Iowa State University, 2002.

16. Allen, W.R., Rosenthal, T.J., Klyde, D.H., Chrstos, J.P., "Vehicle and Tire Modeling for Dynamic Analysis and Real-Time Simulation", SAE Paper 2000-01-1620

17. Open Scene Graph Webpage www.openscenegraph.org, August 1, 2002.

18. "MultiGen Creator Users Guide", MultiGen-Paradigm, Inc. 1999.