

A Unified Framework for Collision Detection, Avoidance, and Response

Kevin L. Steele and Parris K. Egbert
Computer Science Department
Brigham Young University
Provo, Utah USA 84602

ABSTRACT

Collision detection and collision avoidance algorithms are necessary for accurate and realistic animation, but are presently implemented separately and independently. This is a disadvantage when designing some simulations or animations which would otherwise benefit from the availability of both algorithms during run-time. We propose a unified algorithm incorporating both collision detection and collision avoidance simultaneously during an animation. Our algorithm benefits animations whose moving objects generally require collision-free movement, but under certain circumstances may collide with other objects within the scene. The algorithm utilizes vector fields as its basis, and we present a supporting algebra that facilitates the design of a simulation's behavioral interaction. Two sample simulations are presented, and their implementation and performance is discussed.

1 INTRODUCTION

Applications utilizing computer graphics and computer animation are becoming more abundant each year. Such systems take advantage of the communicative power of three-dimensional graphics to more effectively present information to the user, frequently taking advantage of advanced modeling, rendering, and animating techniques to achieve their end.

One important group of animation techniques often required by an application to deliver realistic object movement is collision detection, collision avoidance, and collision resolution or response. These strategies are essential for the correct animation of objects in a scene that would otherwise intersect one another and yield unrealistic results. Collision detection anticipates the exact or approximate time and location in which two objects will collide and begin to inter-penetrate, collision avoidance seeks to avoid such contact through path planning, and collision response directs object motion in response to a collision.

Previous work on these strategies has produced workable solutions [MOO88, GAR94], but many are cumbersome and special-cased. The problem is further compounded when real-time performance is desired, since most proposed solutions cannot be computed in the short interval available between the frames of a real-time system [HUB95].

In this paper we propose a unified framework of vector fields that encapsulates solutions to both collision detection and collision avoidance. With the framework in place, both collision detection and avoidance can be incorporated into the same simulation or animation, providing a simple and general interface to object interaction rules. Other systems [CAM90, LIN95, HUB96] generally incorporate either collision avoidance or detection, but not both. However, some animations can be more effectively executed with both present. For instance, birds flying in a flock tend to avoid one another, but under special circumstances, a strong wind perhaps, they may collide with one another or other objects. Both collision avoidance and detection are necessary to resolve all possible interactions. Rather than require the animator to pre-define the paths of all the birds and then check for collisions on those paths, our unified collision avoidance/detection algorithm can be applied to such animations and will handle all possible interactions without user intervention.

2 VECTOR FIELDS AS INTERACTIVE CONSTRAINTS

The basis of the unified collision analysis framework is the vector field. Typical use of vector fields in past computer graphics applications has been one of passive visualization. Data from natural or physically-based phenomena are gathered into sets or generated from mathematical models, and are collectively represented by vector fields. These vector fields model the actual phenomena, and are rendered to aid in visualization. Weather patterns [WIL91], ocean currents [JOH91], fluid dynamics [SHI91], and general 3D vector fields [CAB93, GLO91, SCH91] have all been made easier to visualize with the aid of advanced three-dimensional rendering techniques.

Recent work [EGB96, HIL94, WEJ91] demonstrates the usefulness of using vector fields as active constraints in polygonal model construction and manipulation, virtual sculpting, interactive terrain generation, animation, and collision avoidance. These methods employ a user-defined vector field utilized as an on-screen tool to interact directly with the model to be modified. As the user passes the vector field over portions of the model, those graphic elements (polygons, vertices, etc.) coming into contact with vectors within the vector field are modified relative to the strength and orientation of the vectors. If these vectors are interpreted as forces or loads, and the polygons of the model are treated as physically-based objects subject to such forces, then those elements of the model can be manipulated or constrained in a predictable and controllable manner.

Using this technique, the modeler can sculpt "hand-crafted" objects, generate realistic mountainous terrain, or define the paths of animated objects—all based on the contours and vector strengths of user-definable vector fields. Vector fields used in this context can be chosen from a library of commonly used fields for modeling, or can be defined using a procedural interface such as a programming language.

2.1 Dynamic Vector Fields

To meet our particular needs, we extend the classic definition of a vector field [STE91] by expanding its function domain to be a subset of R^n rather than R^3 . Thus we define a vector field as:

Let G be a subset of R^n . A vector field on R^n is a function $\mathbf{F}(x_1, x_2, x_3, \dots, x_n)$ that assigns each point $(x_1, x_2, x_3, \dots, x_n)$ in G a three-dimensional vector $\mathbf{F}(x_1, x_2, x_3, \dots, x_n)$.

Expressed in terms of its component functions, this vector field is

$$\mathbf{F}(x_1, x_2, x_3, \dots, x_n) = \mathbf{F}_1(x_1, x_2, x_3, \dots, x_n)\mathbf{i} + \mathbf{F}_2(x_1, x_2, x_3, \dots, x_n)\mathbf{j} + \mathbf{F}_3(x_1, x_2, x_3, \dots, x_n)\mathbf{k}$$

Note that the vector field function maps $\mathbf{F}: R^n \rightarrow R^3$. Having an n -space function domain allows the vector fields to be dependent on any number of variables in the system, not just the three spatial dimensions. We thus think of the vector field as dynamic in nature, and refer to it as a *dynamic vector field*, a vector field in three-space that changes with respect to an arbitrary number of user-defined variables. These variables can represent time, proximity of one object to another, etc.

3 UNIFIED FRAMEWORK

With the dynamic vector field definition in place, we propose a common framework that will unify collision detection, avoidance, and response computations. The basis for the framework is our interpretation of dynamic vector fields as physical force fields, similar to that proposed in [HIL94]. In this approach, each object in a simulation is treated as a particle that can move freely in Euclidean space subject to external forces. The objects maintain their own positions and, depending on the type of simulation, velocities and accelerations as well. Vector fields are then introduced into the simulation, and any particle found within the boundary of a vector field is given an external force vector equivalent to the vector field function evaluated at the position of the particle. Such particles' positions, velocities, and/or accelerations are modified based on the external forces.

A brief description of the framework and associated algorithm follows; a more detailed explanation is given in section 3.2. Assume a simulation or an animation in which collision detection and/or avoidance occur. Each independently moving object in the scenario maintains its own mass, position, velocity, and acceleration, and is surrounded by a dynamic vector field. As objects approach one another and intersect the critical set of another's field, vectors from the fields are sampled and interpreted as forces acting upon the opposing objects. Since the objects maintain mass, position, velocity, and acceleration, their motion can be predicted and controlled via the field that "hit" them. In this way, either collision avoidance, collision detection, or both can be used between any two objects depending on how their field functions are defined. Additionally, the field function determines the collision response to each object involved.

3.1 Vector Field Algebra

An important aspect of our proposed framework is the interaction of vector fields with other vector fields. In a physically-based simulation, overlapping force vectors are usually added together. However, a less restrictive rule would be advantageous, allowing for a more flexible modeling environment. For this reason, we define a vector field algebra to govern the interaction of two or more fields in the same vicinity.

To simplify the situation, we split vector fields into two components—a spatial component, defining the critical set, or spatial bounds outside of which the function yields zero vectors, and an algebraic component, defining the vector field function itself. For example, a vector field function that defined

vectors all pointing away from the origin, and whose lengths were constant within a unit's distance from the origin and zero outside that distance, would have a critical set whose loci of points comprised a unit sphere centered at the origin. Under our split definition of vector fields, the spatial component of that function would be the unit sphere, and the algebraic component would be a function returning constant-length vectors emanating from the origin.

This is arguably unnecessary, since a "traditional" vector field function includes its critical set boundary. We found the split definition enormously helpful, however, in optimizing the interactive treatment of fields using raster graphics algorithms. For instance, since vector fields can be interactively dragged around a simulation, it made sense to provide a geometric primitive such as a sphere or cube that encapsulates the vector field for the user to manipulate and drag. With a visual icon representing the vector field's boundary, the user then intuitively understands the critical set of the field.

We define the spatial operations using the regularized boolean set operations of union (\cup^*), intersection (\cap^*), and difference ($-^*$), and the algebraic operations using the vector operations of addition (+), cross product (\times), and replacement (\rightarrow). This duality of operations in the algebra requires that we include both types of operations in our descriptions and definitions. For instance, we cannot take the cross product (\times) of two vector fields; rather, we take the cross product of the union ($\times \cup^*$) of two vector fields. Figure 1 shows the possible combinations that are defined in the algebra, along with their corresponding operators. The next two subsections formally define the two types of operations.

	Union	Intersection	Difference
Addition	$+\cup^*$	$+\cap^*$	$-^*$
Cross Product	$\times\cup^*$	$\times\cap^*$	$-^*$
Replacement	$\rightarrow\cup^*$	$\rightarrow\cap^*$	$-^*$

Figure 1: Combinations of algebraic operations with boolean set operations to resolve vector field interactions, given with their corresponding operators. The difference operation has no algebraic component because it is exclusively sufficient to resolve the interaction. That is, after the difference operation, there is only one vector field function left, and hence no need for a functional component.

3.1.1 Spatial Operations

Since dynamic vector fields have an $\mathbf{F}: R^n \rightarrow R^3$ mapping, their domains must occupy common dimensions for any spatial operation to be applicable to them. Given two domains R^m and R^n , we define the regularized boolean set operations for the subsets E of R^m and G of R^n , which respectively bound separate vector field functions, to be intuitively the same as in [FOL90], with the following extensions to allow for operations on subsets of differing dimensions:

Let A , B , and C be sets of linearly independent coordinate axes where

$$A = \{\mathbf{a}_1, \mathbf{a}_2, \mathbf{a}_3, \dots, \mathbf{a}_m\},$$

$$B = \{\mathbf{b}_1, \mathbf{b}_2, \mathbf{b}_3, \dots, \mathbf{b}_n\}, \text{ and}$$

$$C = \{\mathbf{c}_1, \mathbf{c}_2, \mathbf{c}_3, \dots, \mathbf{c}_k\}.$$

Let E and G be subsets of R^m and R^n respectively, whose coordinate axes are given by A and B respectively, and $E \bullet G$ be any regularized boolean set operation on E and G . The operation $E \bullet G$ yields a subset F of R^k , $k \leq m + n$, where the coordinate axes of R^k is $C = A \cup B$. The subset F consists of the union, intersection, or difference of $E' = S(E)$; $S: R^m \rightarrow R^k$ with $G' = T(G)$; $T: R^n \rightarrow R^k$, such that for any slice H_1 in R^m (of coordinate axes A) of F , $H_1 = E$, and any slice H_2 in R^n (of coordinate axes B) of F , $H_2 = G$.

The slice operation in the above definition is intuitively defined as taking a subset of the object by holding invariant some of the coordinates of the object while allowing the rest to vary. For example, taking a 2D slice in z of a 3D object is simply extracting a plane out of the 3D object perpendicular to the z axis. The plane extracted is selected by fixing the z -coordinate to a user-defined value. Also, note that the \rightarrow symbol in $\mathbf{S}: R^m \rightarrow R^k$ and $\mathbf{T}: R^n \rightarrow R^k$ represents linear transformations from R^m and R^n to R^k , rather than the replacement operation used previously in the algebra.

To describe the extended boolean set operations in simple terms, each m -dimensional and n -dimensional volume is transformed up into the least k -space possible, with all coordinate axes of R^m and R^n included in R^k , and then combined using the union, intersection, or difference operator. The transformation makes the original volumes invariant across each new dimension. For example, if a two-dimensional image is extruded along the z -axis to create a three-dimensional volume, the new volume is invariant, or constant across z because any two-dimensional slice taken along z is the original image.

The following is an example of an extended boolean set operation: A 3D vector field that is also dependent on time and another 3D vector field that is also dependent on δ (perhaps a distance) are combined using the sum of union operation. The least k -space possible is R^5 , where the fourth and fifth dimensions are time and δ . The first vector field is transformed up to R^5 , and the new object is constant across δ . The second vector field is transformed up to R^5 , and the new object is constant across time. The union of the two new objects is an object of dimension R^5 whose volume includes both of the original two.

3.1.2 Algebraic Operations

Now that we have defined the interaction of vector field volumes, we can define the vector field algebraic operations of addition, cross product, and replacement:

Let A , B , and C be sets of linearly independent coordinate axes where

$$A = \{\mathbf{a}_1, \mathbf{a}_2, \mathbf{a}_3, \dots, \mathbf{a}_m\},$$

$$B = \{\mathbf{b}_1, \mathbf{b}_2, \mathbf{b}_3, \dots, \mathbf{b}_n\}, \text{ and}$$

$$C = \{\mathbf{c}_1, \mathbf{c}_2, \mathbf{c}_3, \dots, \mathbf{c}_k\}.$$

Let E and G be subsets of R^m and R^n respectively, whose coordinate axes are given by A and B respectively. Let \mathbf{U} and \mathbf{V} be vector fields in the subsets E of R^m and G of R^n respectively, and $E \bullet G$ be any regularized boolean set operation on E and G . An algebraic operation $\mathbf{U} \circ \mathbf{V}$ is the vector field \mathbf{V}' in the subset $E \bullet G$ of R^k in which the following three conditions hold:

1. Every point $(c_1, c_2, c_3, \dots, c_k)$ in $(E \cup G)$ is assigned the vector $\mathbf{U}(a_1, a_2, a_3, \dots, a_m)$,
2. Every point $(c_1, c_2, c_3, \dots, c_k)$ in $(G \cup E)$ is assigned the vector $\mathbf{V}(b_1, b_2, b_3, \dots, b_n)$,
3. Every point $(c_1, c_2, c_3, \dots, c_k)$ in $(E \cap G)$ is assigned the vector $\mathbf{U}(a_1, a_2, a_3, \dots, a_m) + \mathbf{V}(b_1, b_2, b_3, \dots, b_n)$ for addition, $\mathbf{U}(a_1, a_2, a_3, \dots, a_m) \times \mathbf{V}(b_1, b_2, b_3, \dots, b_n)$ for cross multiplication, and $\mathbf{V}(b_1, b_2, b_3, \dots, b_n)$ for replacement,

where, for each a_j , $a_j = c_i$, where i is the coordinate axis of $E \bullet G$ that corresponds to the j^{th} coordinate axis of E , and similarly for b .

Dynamic vector fields are closed under addition, cross multiplication, and replacement. Addition is both commutative and associative, and replacement is associative:

\mathbf{U} , \mathbf{V} , and \mathbf{W} are vector fields.

$$\mathbf{U} + \mathbf{V} = \mathbf{V} + \mathbf{U} \text{ (commutative)}$$

$$(\mathbf{U} + \mathbf{V}) + \mathbf{W} = \mathbf{U} + (\mathbf{V} + \mathbf{W}) \text{ (associative)}$$

$$(\mathbf{U} \rightarrow \mathbf{V}) \rightarrow \mathbf{W} = \mathbf{U} \rightarrow (\mathbf{V} \rightarrow \mathbf{W}) \text{ (associative)}$$

Since union and intersection, the only regularized boolean set operations that require further algebraic operations (the difference operation requires no algebraic operations), are both commutative and associative, all regularized boolean set operations can be performed independent of any algebraic field operations. This means that all the algebraic properties mentioned above hold whether the set operation to combine the vector fields be union, intersection, or difference. Figure 2 shows examples of both the spatial operations and algebraic operations.

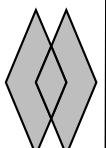
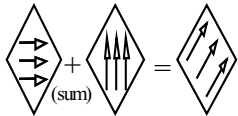
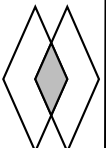
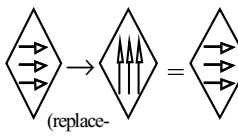
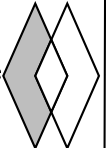
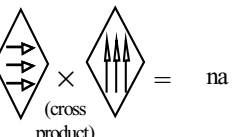
Spatial Operations	Algebraic Operations
	
	
	

Figure 2: Examples of the spatial operations of union, intersection, and difference, and the algebraic operations of addition and replacement. Since cross multiplication only applies to three dimensions, the 2D example is not shown. The result in three dimensions would be a vector field whose vectors are cross products of the operands.

3.2 Collision Avoidance

In [EGB96] Egbert, et. al., described a collision avoidance scheme utilizing vector fields as forces as described in section 3. As objects approach vector fields in their simulations, an opposing force is exerted on the object, which in turn modifies its trajectory to minimize the opposition. The objects in such simulations behave as if there were magnetic repulsive fields between them. The strength of the “magnetic field” is determined by the vector field function.

These vector fields are not dynamic but are static, in that their functions are only dependent on location in the field, providing the mapping $F: R^3 \rightarrow R^3$. The algorithm is discrete, since it freezes objects at regular, or discrete intervals to check proximity to a field and update the object's position and velocity accordingly.

Our framework uses Egbert's technique for collision avoidance, and subsumes and expands it for collision detection. Egbert uses a simple scheme for resolving vector field interactions by taking the sum of the union of any overlapping fields. We instead incorporate the full algebra given in section 3.1, using operations appropriate for the needs of the simulation. For instance, the spatial CSG operations can be used to construct a layered composite vector field as shown in figure 3.

3.3 Rigid-Body Collision Detection

Collision detection seeks to determine the time and point of impact at which a collision between two rigid bodies occurs. We again use a discrete algorithm, checking object proximities at regular intervals. However, the vector fields surrounding each object for collision detection are dynamic, their functions depending on location within the field as well as the type and velocity of object entering the field.

As a simple example of how the algorithm works, consider two spheres moving independently in a closed volume with the possibility of colliding. Each sphere has an associated position and velocity, and for purposes of this example can bounce elastically off the walls of their enclosing volume. Additionally, each sphere has a dynamic vector field surrounding it which maintains the same position and velocity as the sphere. The dynamic vector field has a spherical bounding volume with a radius of at least the length of the sum of the two largest sphere's radii in the simulation (the *only* spheres in this simple example).

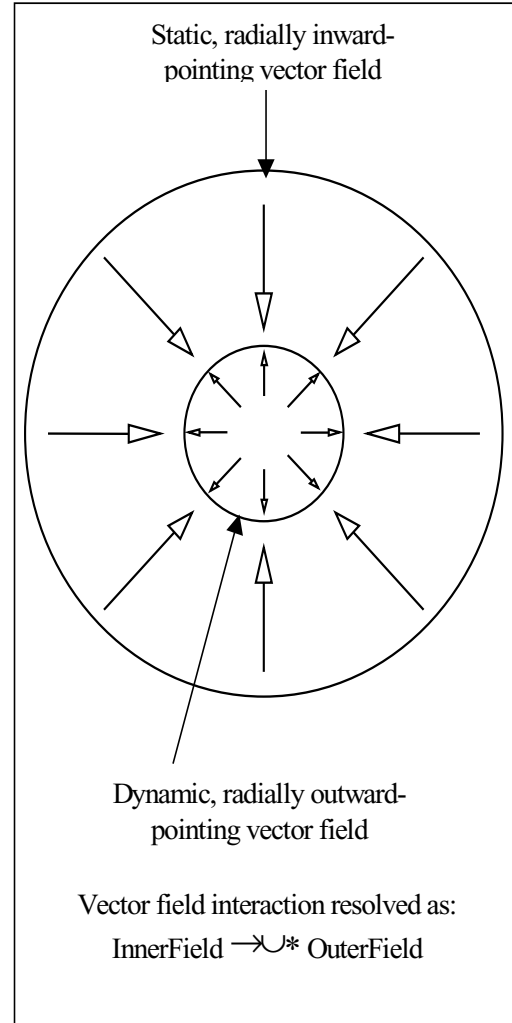


Figure 3: Example of a composite vector field. The outer field is a static, radially inward-pointing field simulating a gravitational field. An object entering the field is pulled toward the center until it comes into contact with the inner field, a dynamic, radially outward-pointing field simulating the surface of another object. With the spatial/algebraic boolean operator, the combination is considered one vector field.

To optimize performance, all sampling of the dynamic vector field function is done at the center of any approaching spheres. A field function yields a vector according to the following rules:

- If no spheres (besides the one it's surrounding) are within its boundary, the function returns the zero vector.
- Otherwise, for each approaching sphere within its boundary:

- If the approaching sphere does not intersect the approached sphere (determined from each sphere's radius, hence the need for a dynamic vector field), the function returns the zero vector.
- Otherwise, the function returns a force vector in the direction of the approaching sphere whose length is the magnitude of the velocity vector of the approaching sphere.

Spheres that receive force vectors can change their acceleration, velocity, and position accordingly. If these variables are modified according to Newtonian laws of motion, a physically-based simulation is produced.

A more complex simulation would be one in which objects are comprised of multiple parts, each having its own surrounding vector field. These objects could additionally experience torque forces, exhibit angular velocity, and preserve angular momentum. In such a simulation, the repulsive vector fields behave exactly the same as in the simple case; the only difference is force vectors are applied to angular as well as translational motion. Section 4 describes an example of this in which we construct composite vector fields around more complex objects that spin and at times collide.

3.4 Unification

Given the previous algorithms for collision avoidance and detection, it is straightforward to incorporate them both into the same simulation. Essentially, each object that has the potential of colliding with other objects is assigned its vector field based on how it will interact with the rest of the scene. If an object is to be avoided, a static repulsive vector field of the appropriate strength is assigned to it. If an object is allowed to actually collide with other objects (that aren't trying to avoid it), a dynamic vector field is assigned based on the rules in section 3.3. The same object can even be given both types of fields. Since all vector field interactions are resolved using the algebra under its binary operations, all objects' surrounding vector fields are inserted into a tree (see figure 4) prior to starting the simulation. While the simulation is running, collisions are resolved using one algorithm that simply evaluates the composite vector field and updates motion variables accordingly.

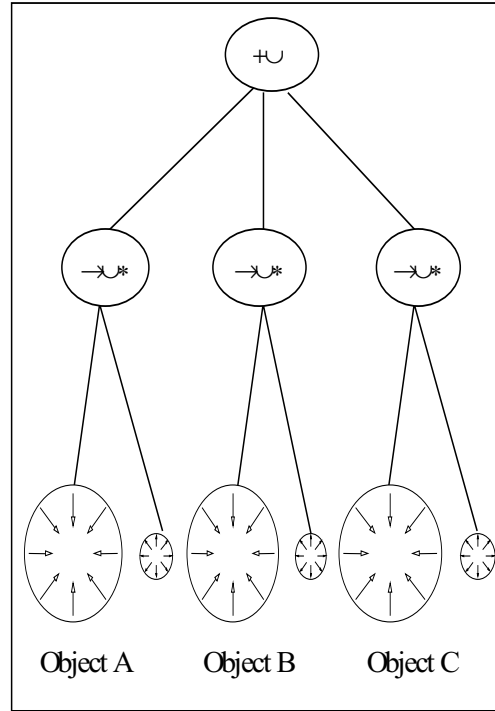


Figure 4: Tree structure of a composite vector field. The leaves of the tree are individual vector fields, and the nodes are spatial/algebraic operators. The tree is interpreted as a single composite vector field. In this example, three objects A, B, and C each have a large attractive field and a small repulsive field as in figure 3 which are combined using a replacement of union operator. These three compositions are then combined with a sum of union operator. If the three objects were placed in a simulation, they would attract one another until a collision occurred, at which time they would bounce in a physically accurate manner. Note that binary operators typically must be placed in a binary tree. Since the sum of union operator is associative, however, it can have any number of operands, so its node can have any number of children.

4 IMPLEMENTATION AND PERFORMANCE

We implemented our simulations and supporting vector field algebra under Open Inventor^{TM1} running on Silicon Graphics platforms. Under its strictest definition, the algebra calls for vector field boundaries of arbitrary shape and dimension. Our implementation, however, utilizes only four geometric primitives provided by Open

¹Open Inventor is a trademark of Silicon Graphics, Inc.

Inventor—sphere, cone, cube, and cylinder—as shape boundaries. Additionally, we have allowed only the algebraic domain, rather than both the algebraic and spatial domains, to have more than three dimensions. These two restrictions made implementation much easier and more efficient, while still permitting us adequate freedom in defining our vector fields. As an example of these restrictions, consider a dynamic spherical vector field whose field strength, i.e., vector magnitude, varies inversely proportional to the field's distance to a wall (increases as it approached the wall, etc.). The algebraic domain is four-dimensional, comprised of x , y , z and distance, but the spatial domain, the spherical boundary, remains three-dimensional, being comprised only of x , y and z .

Given these restrictions in our algebra, our first simulation was a proof-of-concept for the dynamic vector fields and algebra and involved collision detection among a group of equally-sized spheres. Each sphere was surrounded by a dynamic vector field, as described earlier, and collisions were checked each frame. On collision, the spheres received forces and reacted in a physically accurate manner. Running on an SGI O2 with a single MIPS R5000 processor, we achieved reasonable interactive performance with up to about twenty spheres.

Our second simulation utilized both collision detection and avoidance to demonstrate the interaction of ammonia molecules in a closed container. Each ammonia molecule, comprised of three hydrogen atoms and one nitrogen atom, was surrounded with two vector fields per atom—a dynamic field for handling the physical interactions of collisions with other molecules, and a static field for simulating the electrical attractions and repulsions between molecules (see color plate 1). Collision detection is seen in the physical interactions, and collision avoidance in the electrical interactions. Translational and angular velocities are computed on a frame-per-frame basis with forces from the fields computed as described above; color plate 2 shows a short sequence of the animation where two molecules are colliding. Running on an SGI O2 with a MIPS R5000 processor, our simulation performed adequately, maintaining interactive frame rates with up to about ten ammonia molecules.

5 CONCLUSION

We have designed the vector field algebra to be general purpose. This is done to provide an algebraic framework for many diverse applications, each requiring its own subset of the algebra's operations for its activities. Hence, each implementation of the

algebra is generally a subset of its full capability, and can be tailored to the needs of the application. For instance, as mentioned earlier, the vector field definition calls for virtually any type of 3D geometry to be available for use as the bounding volume of the vector field. If an application needs only simple geometric primitives for its vector field volumes, there is no need to provide it with computationally expensive and time consuming NURBS surfaces. Likewise, another application may require more flexible surface definitions, which it utilizes at the cost of performance.

As shown by the second simulation, our framework of vector fields effectively intermixed collision detection and avoidance with successful results. During simulation run-time, only one straightforward evaluation algorithm is necessary to resolve collisions, reducing overhead and simplifying implementation. The framework is general enough to permit many physically-based properties and elements to be implemented within it, including gravity, light, electromagnetism, and rigid-body collisions.

REFERENCES

- [CAB93] Cabral, B. and Leedom, L. Imaging Vector Fields Using Line Integral Convolution. Proceedings of SIGGRAPH 93 (Anaheim, California, August 1-6, 1993). In *Computer Graphics Proceedings, Annual Conference Series, 1993*, ACM SIGGRAPH, New York, 1993, pp. 263-270.
- [CAM90] Cameron, S. Collision Detection by Four-Dimensional Intersection Testing. *IEEE Transactions on Robotics and Automation*, 6, 3 (June 1990), 291-302.
- [EGB96] Egbert, P. and Winkler, S. Collision-Free Object Movement Using Vector Fields. *IEEE Computer Graphics and Applications*, 16, 4 (July 1996), 18-24.
- [FOL90] Foley, J. D., van Dam, A., Feiner, S. K., and Hughes, J. F. *Computer Graphics, Principles and Practice (Second Edition)*. Addison-Wesley, Reading, MA, 1990.
- [GAR94] Garcia-Alonso, A., Serrano, N., and Flaquer, J. Solving the Collision Detection Problem. *IEEE Computer Graphics and Applications*, 14, 3 (May 1994), 36-43.
- [GLO91] Globus, A., Levit, C. and Lasinski, T. A Tool for Visualizing the Topology of Three-

- Dimensional Vector Fields. In *Visualization '91*. (October 1991), IEEE Computer Society Press.
- [HIL94] Hilton, T. L. and Egbert, P. K. Vector Fields: An Interactive Tool for Animation, Modeling and Simulation with Physically Based 3D Particle Systems and Soft Objects. In *Proceedings of Eurographics 94, Computer Graphics Forum*, 13, 3 (1994), 329-338.
- [HUB95] Hubbard, P. Collision Detection for Interactive Graphics Applications. *IEEE Transactions on Visualization and Computer Graphics*, 1, 3 (September 1995), 218-230.
- [HUB96] Hubbard, P. Approximating Polyhedra with Spheres for Time-Critical Collision Detection. *ACM Transactions on Graphics*, 15, 3 (July 1996), 179-210.
- [JOH90] Johnson, M. A., and O'Brien, J. J. Modeling the Pacific Ocean. *The International Journal of Supercomputer Applications*, 4, 2 (Summer 1990), 37-47.
- [LIN95] Lin, M. C. and Manocha, D. *Fast Interference Detection Between Geometric Models*. The Visual Computer, 11, (c) Springer-Verlag 1995, 542-561.
- [MOO88] Moore, M. and Wilhelms, J. Collision Detection and Response for Computer Animation. *Computer Graphics*, 22, 4 (August 1988), 289-298.
- [SCH91] Schroeder, W. J., Volpe, C. R., and Lorensen, W. E. The Stream Polygon: A Technique for 3D Vector Field Visualization. In *Visualization '91*. (October 1991), IEEE Computer Society Press.
- [SHI90] Shirayama, S. and Kuwahara, K. Flow Visualization in Computational Fluid Dynamics. *The International Journal of Supercomputer Applications*, 4, 2 (Summer 1990), 66-80.
- [STE91] Stewart, J. *Multivariable Calculus (Second Edition)*. Brooks/Cole, Pacific Grove, CA, 1991.
- [WEJ91] Wejchert, J. and Haumann, D. Animation Aerodynamics. *Computer Graphics*, 25, 4 (July 1991), 19-22.
- [WIL90] Wilhelmson, R. B., et al. A Study of the Evolution of a Numerically Modeled Severe Storm. *The International Journal of Supercomputer Applications*, 4, 2 (Summer 1990), 20-36.