Eurographics Workshop on Visual Computing for Biomedicine (2008)
C. P. Botha, G. Kindlmann, W. J. Niessen, and B. Preim (Editors)

A Haptic Rendering Algorithm for Molecular Interaction

M. B. Stocks and S. D. Laycock

School of Computing Sciences, University of East Anglia, Norwich, NR4 7TJ, UK
Matthew.Stocks@uea.ac.uk, sdl@cmp.uea.ac.uk

Abstract

Haptic rendering is the process of calculating and displaying physical forces to a user. Used concomitantly with a virtual environment it can further enhance a user's immersive experience whilst they interact with computer graphics. Haptic Feedback has been applied to the study of molecular systems for several years, however, computation requirements have hampered progress. Most popular representations of molecules comprise of primitive shapes like spheres. Many molecules, especially proteins, potentially contain thousands of atoms each of which can be represented as a single sphere and will need to be processed for collision in the haptic rendering loop. Current systems often simulate stiff contacts with a proxy system based on tracking a point over planar surfaces. In this paper a novel method for the haptic rendering of a space filling molecule representation is presented. The technique reduces the time taken to detect and respond to the collisions and improves the overall spherical feel of the molecule by using the implicit description of spheres to track the surface as opposed to a polygonal approximation.

Categories and Subject Descriptors (according to ACM CCS): I.3.6 [Computer Graphics]: Interaction Techniques

1. Introduction

Haptic rendering allows physical interaction with virtual environments through the use of three dimensional input and force feedback. However, in order to achieve stable feedback, refresh rates of the haptic frames must be 1 KHz or greater. Conversely, visualisation techniques can allow a frame rate of 30 Hz. For this reason archetypal haptic systems use separate threads for the haptics and graphics rendering loops. Collision detection and response will all be processed in the haptics thread and the information is typically read by the graphics thread to provide the visual cues to the user.

Accepted representations of molecules are mostly constructed from primitive geometric shapes. The history of these shapes date back to before electronic computers existed. Originating as paper drawn renders, tangible molecular models constructed from wood and plastic quickly became popular. However, the complexity of large molecules like proteins meant computer graphics was the best way forward. Molecular Graphics (MG) is used primarily for research into molecular structure. However, research into haptic feedback for molecular simulation is still in its infancy. There are three main areas where haptic feedback is applied to molecular graphics. The first is the teaching of structural biology. It is traditionally done using computer renderings of molecules, however, with a haptic system a user can fully explore all areas of the geometry using the extra sense of touch afforded from a haptic interface device [SWS*03]. Particle steering is another application for haptic systems with the device used to manipulate real world molecular structure [BKB*07]. The majority of research, however, is into the docking of proteins and ligands [SB08].

The calculations required for physical interactions between atoms are complex and often require prohibitive amounts of computing power preventing real time performance. To alleviate the computation of atomic interactions a Molecular Dynamics (MD) simulation can be executed in a preprocessing stage. Many proteins are constructed from thousands of atoms, potentially meaning several thousand collision checks are also required each haptic frame. Haptic rendering for virtual environments generally employs one of the various constraint-based rendering algorithms. These techniques solve the problems found with rendering convex and concave shapes by allowing the haptic interface point (HIP) to penetrate the surface of a polygon, then constraining a visual point to the surface (Surface Contact Point,

SCP), see Section 3.3. As the HIP moves the SCP is tracked over the features of the polygonal model. Therefore, after the initial collision has been recorded a significant reduction in computation is required to perform the tracking. This is because only the neighbouring polygons to the current contact need to be considered. Basdogan's work revealed that despite large increases in the model's complexity the time taken to render a haptic frame increased only slightly or not at all as the amount of neighbour searches remained approximately constant [HBS99].

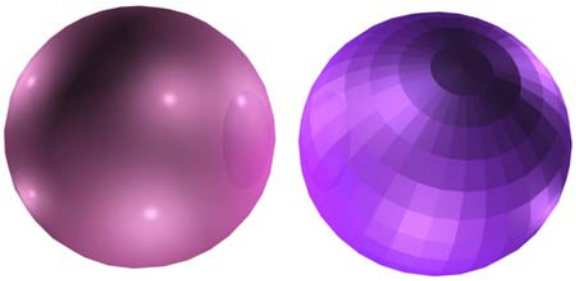

Figure 1: *Two spheres are constructed with an equal amount of polygons. Modern rendering techniques allow techniques such as interpolating normals to trick the eye into perceiving a smooth sphere. However, if this same information is used for the haptic collision detection and response the sphere will feel polygonal as it appears on the right.*

When attempting to render spherical objects using a constraint-based technique designed for polygonal models the amount of polygons required to construct the shape must be considered. In Figure 1 there are two spheres constructed from the same amount of polygons. This scene comprises of 1920 separate triangles (960 per sphere) all of which would need to be processed when the HIP is not in collision with any polygon. After a collision has taken place the surface would be tracked as normal. Additionally Figure 1 illustrates how graphical rendering tricks can cause the spheres themselves to appear smooth and spherical, as seen on the left. Some of these tricks may be employed in haptic rendering such as interpolation of surface normals to generate smoother forces. However, the sphere will still feel polygonal due to the geometry of the sphere, as illustrated on the right of Figure 1. One way to represent a smoother surface would be to add more polygons which would increase the complexity of the collision detection stage.

2. Our Contribution

The mathematical properties of the three dimensional objects used in MG can be utilised to decrease the amount and complexity of the computations required for constraint-based haptic rendering algorithms. This paper presents a novel approach to the haptic rendering of molecules represented in space filling mode utilising the implicit equations of spheres and planes. The space filling model is constructed from overlapping spheres with radii taken from various sources such as Van der Waals [A.B64]. Generally each atom in the periodic table is allocated its own radius and colour in order to provide a clear image. The algorithm presented shows how in addition to providing a smooth spherical feel to molecular models it is also possible to further reduce the amount and complexity of computations required when detecting and handling collisions with spheres. In addition, as with the polygonal approach, the issue of incorrect force vectors found when the HIP is allowed to penetrate a concave region can also be solved with the approach.

3. Previous Work

This section surveys the previous work related to haptic rendering and specifically the application of haptic rendering for molecular visualization. This includes the processing of collisions and the force computations required for stable haptic systems. The main application for the new approach detailed in this paper is in the exploration and teaching of molecular structure by enabling a user to feel the surface of simulated atoms. The research by Sankaranarayanan et al gives a full review of the role haptics plays in the teaching of structural biology [SWS*03]. However, the main body of research lies within the field of protein docking. The main approaches to force calculations in these systems are detailed in Sections 3.1 and 3.2. The current main approach to surface tracking is provided in Section 3.3.

3.1. Offline force computation

Offline force computation involves precomputing the forces using a Molecular Dynamics simulation. Typically the calculations will require execution on a supercomputing cluster. This is due partly to the complexity of the algorithms used in the various simulations. Mainly the amount of computing power required is directly related to the complexity of the molecular structures being studied. The complexity of the algorithms required for accurate simulation will scale up as the amount of atoms involved increases. As proteins contain large amounts of atoms, MD simulations involving them often become too complex to be executed in real time.

A recent implementation of an offline force calculation technique was developed by Wollacott et al. and a similar approach was taken by Ho et al. [HBS99, AK06]. Wollacott's implementation involves pre-processing the possible forces into a grid, then as the haptic interface point moves through the grid the forces are read from memory and returned accordingly. Due to the complications involved in these types of calculations a maximum of 50 objects can be rendered as haptic objects. Ho et al. use a system of offline computation coupled with a real time environment. A successful binding

between protein and ligand involves finding the configuration of both molecules that results in the lowest interaction energy coupled with the largest surface contact. This is calculated with the Molecular Dynamics simulation. However, there are other local minimums over the surface of a protein that could cause the ligand to become trapped. Ho et al. first run a separate program to determine the potential binding sites. To acheive this the haptic workspace must be defined around the binding site to form an area referred to as the Active Haptic Workspace (AHW). This area is a magnified high resolution chunk of the protein which can then easily be explored using the haptic device. When a rough configuration is determined based on the Van der Waals interactions between atoms the Molecular Dynamics simulation then calculates the correct configuration in offline time-steps.

3.2. Real-time force computation

Attempting to calculate the forces in real-time imposes obvious challenges in terms of efficiency. Perhaps the most widely used piece of MG software incorporating real-time force calculation is Visual Molecular Dynamics (VMD) [HDS96]. This program is used as a graphical front end for a MD program designed for parallel super computers called NAMD [LRM*99]. The two programs communicate with each other through TCP/IP and are interfaced with a haptics device in a package called Interactive Molecular Dynamics [SGGS01]. This allows for real-time calculation of all areas of the Molecular Dynamics simulation and the processing of coordinate updates each time the simulation refreshes a frame. However, this type of system requires an efficient connection between the desktop computer and a system able to perform large amounts of computing at high speed. Other related areas of research in the area of force feedback in molecular graphics include the nanomanipulator [TRC*93] and the Docker [FPBOYBK90] both from UNC.

3.3. Haptic Rendering of Polygonal Objects

To simulate stable and stiff forces a haptic rendering algorithm must achieve an update rate of 1 KHz. Most systems use a polygonal approach to collision detection and then employ a tracking method to track a point constrained to the surface, (SCP), whilst the HIP moves internal to the geometry. A force can then be calculated based on a virtual spring between the HIP and SCP. These algorithms were first developed by Zilles and Salisbury in their work on constraint-based haptic rendering in 1995 [ZS95]. Basdogan et al. also developed a constraint-based rendering system based on a procedural approach [HBS99]. Depending on the structure of the program each triangle in the scene will need to be tested for collision with the haptic point every frame (for more complex scenes employing spatial partitioning methods will reduce the amount of tests required). After a collision has been detected the SCP can be tracked along the surface of a polygon using Equation (1) shown graphically as Case 1, in Figure 2.

$$\begin{aligned} PHIP &= \text{Previous HIP} \\ PSCP &= \text{Previous Surface Point} \\ \vec{V} &= (HIP - PHIP) \\ C &= \vec{V} \cdot \hat{N} \\ \vec{M} &= \vec{V} + (\hat{N}C) \\ SCP &= PSCP + \vec{M} \end{aligned} \tag{1}$$

The haptic rendering algorithm must track the SCP over the surface of the polygonal object by considering the distances to the features of the mesh, namely the edges, vertices and polygons to ensure the SCP remains on the closest feature to the HIP. The algorithm must then test if the HIP has left the surface of the current feature [ZS95]. The algorithm will then return a force using standard spring equations based on the vector between the HIP and the SCP. The next section discusses the haptic rendering of concave objects used in standard constraint-based haptic rendering algorithms for polygonal objects.

4. Haptic Rendering of Concave geometry

As previously stated this algorithm is inspired by a constraint-based haptic rendering algorithm for polygonal objects. The next two sections describe in detail the steps taken to handle concave geometry in a polygonal model, as both cases presented here have counterparts in the new algorithm detailed in this paper.

4.1. Two Intersecting Plane

When two intersecting planes form a concave section a haptic rendering algorithm must deal with two cases, either the SCP lies on one of the planes or it should be constrained to the line of intersection. Figure 2 illustrates both of these cases. Case 1 is the simple initial phase where the SCP is tracked along the primitive based on the position of the HIP. Case 2 is where the SCP is constrained to the line of intersection. The image illustrates two positions for the HIP and SCP. After the initial contact the SCP must be tracked along the plane using Equation (1). If the HIP moves into the volume defining the locus of points closer to the line than either plane then the SCP must be constrained to the intersection of the two planes. Then as the HIP moves closer to the right hand plane the state will revert back to Case 1 where the SCP can be tracked along the surface of the new primitive.

Once the HIP moves out of collision, the algorithm must revert to its initial form of checking for intersection with all the planes in the scene. The next section provides an overview of how the polygonal algorithm handles three planes intersecting in a concave formation.

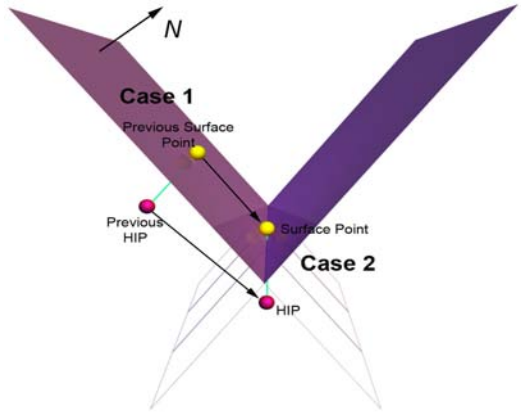

Figure 2: *The polygonal approach to haptic rendering of two intersecting polygons forming a concave angle. Case 1 shows the initial position after contact, as the HIP moves the SCP tracks relative to it. As it moves under both primitives it is constrained to the line between the two polygons until it moves closer to the right primitive when it reverts to Case 1.*

4.2. Three Intersecting Planes

It is apparent from Figure 3 that the three planes are intersecting at a single point. If the HIP moves into the volume in which every point is closer to the intersection of the three planes as opposed to one of the planes then the SCP must be constrained to the intersection point. Once the HIP moves closer to one of the planes the state will revert to Case 1 to allow the algorithm to continue tracking the SCP on the new primitive.

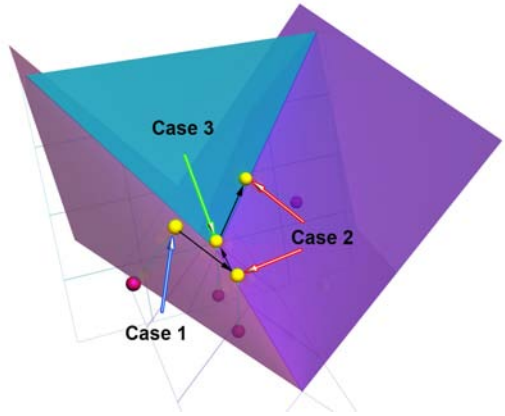

Figure 3: *The polygonal approach to haptic rendering of three overlapping planes. As before Case 1 shows the initial position. As it moves under all three planes the SCP is constrained to the point where all three primitives intersect. If the HIP moves closer to one of the planes the state of the algorithm updates to the appropriate case.*

The polygonal approach is restricted to simulating faceted objects. Force interpolation similar to Phong shading used in lighting could be employed, however, forces more faithful to the spherical shape of molecules can be generated by utilising the implicit equations of spheres. The next section details a novel algorithm to efficiently render spheres suitable for a Molecular Graphics application.

4.3. A Haptic Rendering Algorithm for Spheres

The algorithm is analogous to the polygonal rendering algorithm discussed in the previous section. The space filling representation of a molecule represents the input to the algorithm. The polygons or planes of the previous algorithm are replaced with spheres in the new input. The intersection of two planes discussed previously is analogous to the intersection circle formed from two intersecting spheres. The intersection of three planes is analogous to the intersection of three spheres intersecting at two points. Figure 4 shows a trace through the three cases of the algorithm which are required to ensure the SCP can reliably track the HIP over the surface of a cluster of spheres. Case 1 illustrates the simple case resulting after initial contact with a sphere. At this point the algorithm simply calculates the closest point on the surface of the sphere. Case 2 depicts the state of the algorithm where the SCP must be defined as the closest point from the HIP to the circle formed by the intersection of the two spheres. Finally, Case 3 shows the SCP constrained to one of the intersection points resulting from the intersection of three spheres.

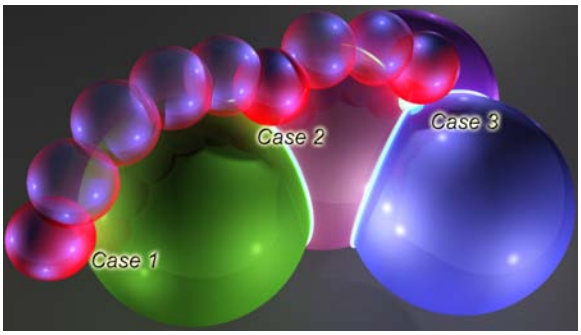

Figure 4: *A trace through all three states of the new algorithm. Case 1 illustrates the case after initial contact. Case 2 depicts the state of the algorithm when the HIP is closest to the circle of intersection. Case 3 shows the SCP being constrained to a single point when the HIP is closest to one of the intersection points.*

The parallels between this system and the polygonal approach are clear. However, the transitions between the cases can be calculated simply using the implicit equations defining the spheres forming the space filling representation of

the molecule. In this representation the features that must be tracked include spheres, circles and points. When the HIP penetrates a sphere the closest point on the surface of the sphere can be calculated and this forms the SCP. As the HIP moves the SCP can be recalculated as the closest point to the sphere the HIP initially penetrated. If the HIP moves out of the current sphere it is assumed to be no longer in collision. However, if the HIP remains in collision and moves into areas such that the SCP should be constrained to the intersection areas of two or more spheres additional computation is required. The following sections illustrate how the SCP can be tracked over the features of the space filling representation relative to the HIP.

4.4. Two Overlapping Spheres

In Case 2 in the new approach the SCP will be constrained to a circle formed by the intersection of two spheres, as opposed to the line in the polygonal approach. Figure 5 illustrates two overlapping spheres in two dimensions with one sphere placed at the origin and the other translated along the x axis. The intersection of these two primitives forms a circle, denoted by a vertical line in the figure.

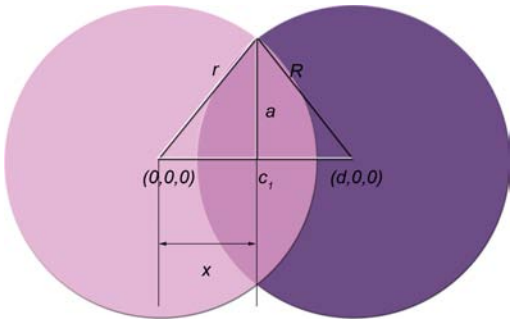

Figure 5: *A two dimensional representation of intersecting spheres with radii r and R.*

Equations (3) and (4) are used to calculate the radius and centre of the intersection circle. They are based on solving the implicit equations of the two spheres simultaneously. From this solution the value x can be obtained, which is the length of the line labelled in Figure 5. Also the value a can be determined as the radius of the intersecting circle.

$$x = \frac{d^2 - r^2 + R^2}{2d} \tag{2}$$

$$a = \frac{1}{2d}\sqrt{4d^2R^2 - (d^2 - r^2 + R^2)^2} \tag{3}$$

$$\vec{S} = (S_1 - S_0)$$
$$C_1 = S_1 + (\hat{S}x) \tag{4}$$

The transition from Case 1 to Case 2 occurs when the HIP is closer to the intersection circle than the sphere it initially collided with. The test for transition is performed utilising cones constructed from the circle and the centres of the intersecting spheres. The bases of two cones are defined by the intersection circle and each cone has height, x and $d-x$. Figure 6 illustrates the two intersecting spheres with the cones visible in three dimensions.

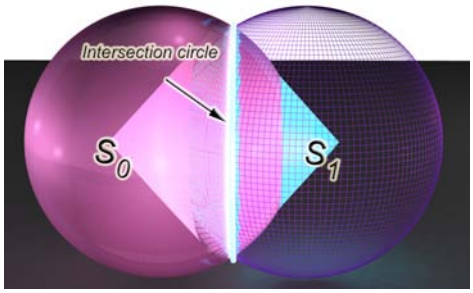

Figure 6: *The two intersecting spheres defined with centres S_0 and S_1. When the HIP is inside the cone the SCP becomes the closest point on the intersection circle.*

To determine if the HIP is interior to a cone the HIP is projected onto the plane that contains the intersection circle. If the distance between the projected point and C_1 is greater than the radius of the circle, a, then the test for transition is terminated. However, if the distance is less then a dot product can be used to determine if the HIP lies inside the cone. If it is then the SCP becomes constrained to the circle formed by the intersection of the two spheres. As the HIP moves it can move closer to two features either the neighbouring cone or returning to the previous sphere. If it is returning to a sphere then the algorithm reverts to Case 1. Alternatively if the HIP moves into a neightbouring cone then Case 2 still applies and the SCP remains constrained to the intersection circle. However, the data structures must be updated to ensure the current sphere is the neighbour of the previously contacted sphere. Ensuring the closest features are updated enables the SCP to track over the features of the space filling representation without performing any collision detection functions.

In the space filling representation multiple spheres can be intersecting each other. The next section describes how the algorithm deals with this situation.

4.5. Multiple Overlapping Spheres

In a pre-processing stage the intersection points between several intersecting spheres must be calculated and added to the data structure utilised in the haptic loop. In order to determine the intersection points between three spheres the intersection points between the circle calculated in Section 4.3 and a third sphere can be obtained. The plane P the circle

lies in can be defined by a normal vector n passing through point, C_1. A new circle is formed where this plane cuts the third sphere being considered. Calculating this new circle involves using the radius of the third sphere r and the constant value taken from the implicit plane equation, d. If the value for d is less than r then we find a new circle lying in the same plane as the first intersection circle with squared radius $r^2 - d^2$ and with a centre at $n * d$ which is referred to as C_2. From these two circles we must then determine the points of intersection as these will be the points at which the SCP must be constrained to when the algorithm is in Case 3. These points are shown in Figure 7.

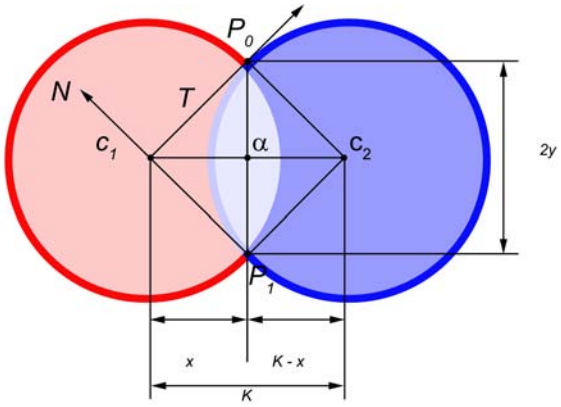

Figure 7: *The intersection of the third sphere and the intersection circle. P_0 and P_1 define the intersection points.*

Equations (5) and (6) are used to calculate the intersection points. x can be calculated using Equation (2). C_2 and C_1 define the centres of the new circles.

$$\begin{aligned} \vec{V} &= (C_2 - C_1) \\ y^2 &= R^2 - x^2 \\ K &= \left|\vec{V}\right| \\ \alpha &= C_1 + (\hat{V}x) \\ \vec{F} &= \hat{n} \times \vec{V} \\ P_0 &= \alpha + (\hat{F}y) \qquad (5) \\ P_1 &= \alpha - (\hat{F}y) \qquad (6) \end{aligned}$$

The transition to Case 3 occurs when the HIP lies within the volume defined by planes. One plane is constructed for each intersection circle. In the case of three intersecting spheres the planes are described with the points of intersection P_0 or P_1 and a normal $\vec{N}$ calculated in Equation (7) with the vector $\vec{T}$ describing a vector that lies on the plane itself.

$$\begin{aligned} \vec{T} &= \begin{cases} P_0 - C_1, & \text{for } P_0 \\ P_1 - C_1, & \text{for } P_1 \end{cases} \\ \vec{N} &= \vec{T} \times n \qquad (7) \end{aligned}$$

It is possible for the HIP to lie under both points and therefore the algorithm could become confused as to which point the HIP should be constrained to. In order to counteract this a Euclidean distance test is performed between the SCP and points P_0 and P_1. If the distance is less than a specified threshold for either of the intersection points then the algorithm must start testing the planes attached to that point. As the HIP moves the algorithm must track to the next feature. The initial test is to determine if the HIP has moved past any of the planes attached to the current intersection point. However, it is necessary to know which feature to constrain to after this event. To solve this the solution is to keep unique identifiers to the cones that were used to create the planes. When the HIP moves to the opposite side of the plane it will query that plane to determine which intersection cone it should constrain the SCP to.

5. Implementation and Results

The system was implemented and tested using the following hardware: Intel Core 2 Duo Quad Edition, 3GB RAM and an nVidia 8800GTX graphics card. The haptic interface device used was the Phantom Omni from Sensable technologies [Pha]. The program was implemented in the C++ programing language utilising the OpenGL API for the graphics and the HD version of the Open Haptics Toolkit for the haptics. To validate the utility of the algorithm the polygonal approach to haptic rendering was implemented using the Chai3d toolkit [FFDC05], both implementations employ similar spacial partitioning methods to improve the servo times.

Haptics programs typically must run at a rate higher than 1KHz. As graphics programs need only refresh at a rate of around 60Hz most implementations utilise several separate threads for the haptics servo loop and the graphics. The HD API used for the implementation of the algorithm allows low level interaction with the Omni device and requires a programmer to input each step required for haptic feedback but allows accurate timing of the thread. Chai3d however, is designed for the rapid development of haptics programs using various techniques (including polygonal) but still gives a programmer direct access to the servo loop allowing timing in the same way as the HD API.

Table 1 details the computation times to check for collisions whilst the HIP moves in free space around the molecule. Table 2 includes the same input data and illustrates the average computation times when the HIP is in contact. The first two test cases involve two files containing a random

Algorithm 1 The new surface tracking algorithm

```
if Not in collision with an atom then
  for all Atoms in the molecule do
    if The HIP is interior to Atom then
      Set collision flag to true
      Start tracking on the current Atom
    end if
  end for
else
  Constrain SCP to surface of the current Atom
  if Not in collision with a cone then
    for all Cones in the current Atom do
      if The HIP has collided with a Cone then
        Set in cone flag to true
        Start tracking on the current Cone
      end if
    end for
  else
    Constrain SCP to surface of the current Cone
    if The HIP is closer to the neighbour Cone then
      Update the Cone to become the neighbour Cone
      Update the Atom to become the Cones parent
    end if
  end if
  if |SCP - Intersection Point| < ε then
    for all intersection points in the current Atom do
      if |HIP - Intersection Point| < ε then
        Set in sphere flag to true
        Start tracking on the current Point
      end if
    end for
  else
    if The HIP moves inside the Point Volume then
      Constrain SCP to surface of the Point
    end if
  end if
end if
```

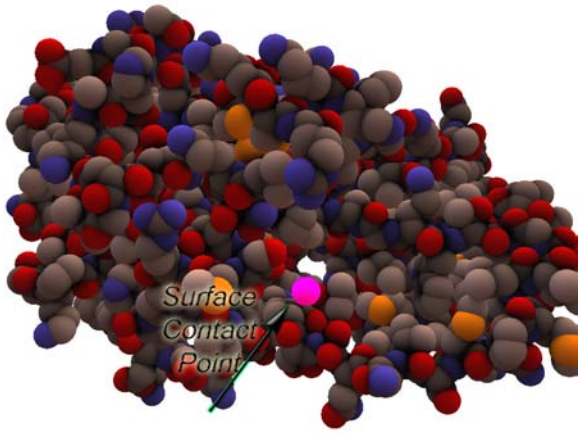

Figure 8: *A screen shot taken from the program showing a representation of Alcohol Dehydrogenase being touched with a water molecule.*

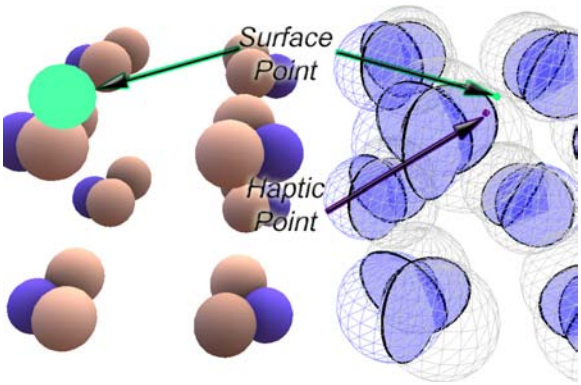

Figure 9: *A screen shot taken from the program illustrating the actual positions and sizes of the spheres. The haptic shapes are rendered in wireframe to show the HIP penetrating the surface.*

arrangement of spheres and the latter three are molecules obtained from the Protein Data Bank [HKHJ07] and are labeled with their pdb code in the table.

Figure 8 is a screen shot taken from the program used for both implementations. The file being used in this example contains 2785 atoms.

Figure 9 is an illustration of the haptic algorithm with the actual haptic data rendered on the right of the image. This shows how adding the radius of the simulated water molecule allows the system to remove the need to compute multiple points of contact on the surface of the water molecule.

The results show that our system is able to handle considerably large atomic coordinate files and is able to maintain stable rates during collision.

Table 1: *Comparison of computation times for a Polygonal Approach and the Proposed Algorithm while the HIP is in free space around a simulated molecule*

Name	Amount	Polygonal (Hz)	New system (Hz)
Test 1	50	2.19×10^{-6}	5.29×10^{-7}
Test 2	150	5.86×10^{-6}	5.39×10^{-7}
5ADH	2785	0.00015	5.32×10^{-7}
8TIM	3735	0.00033	5.47×10^{-7}
1KBW	13751	N/A	0.008

Table 2: *Comparison of computation times for a Polygonal Approach and the Proposed Algorithm while the HIP is in contact with a surface of a molecule*

Name	Amount	Polygonal (Hz)	New system (Hz)
Test 1	50	4.92×10^{-5}	5.07×10^{-6}
Test 2	150	9.46×10^{-5}	9.89×10^{-6}
5ADH	2785	0.0013	6.30×10^{-6}
8TIM	3735	0.0024	1.49×10^{-6}
1KBW	13751	N/A	1.56×10^{-6}

6. Conclusion

In this paper a novel haptic rendering algorithm to efficiently render molecules represented in space filling mode has been presented. Through experimental results it is shown that our algorithm is a significant improvement over the polygonal approach to tracking the surface of spheres during haptic rendering. It allows for dramatically higher frame rates and therefore allows for a larger amount of atoms to be processed in real-time. The user's perception of the primitives being rendered is also improved since the implicit equations of the spheres are utilised instead of a polygonal approximation. This system works in a similar way to the constraint-based rendering algorithm for polygonal objects. Due to this, the algorithm could easily be extended to incorporate surface properties such as friction and texture. However, the main focus of future work aims to invesitgate the rendering of molecules undergoing deformations.

References

[A.B64] A.BONDI: van der waals volumes and radii. *The Journal of Physical Chemistry 68*, 3 (1964).

[AK06] A.M.WOLLACOTT, K.M.MERZ.JR: Haptic applications for molecular structure manipulation. *Journal of Molecular Graphics and Modeling 25*, 6 (2006), 801–805.

[BKB*07] BASDOGAN C., KIRAZ A., BUKUSOGLU I., VAROL A., DOANAY S.: Haptic guidance for improved task performance in steering microparticles with optical tweezers. *Optics Express 15* (2007), 11616–+.

[FFDC05] F. C., F. B., D. M., C. S.: Chai: An open-source library for the rapid development of haptic scenes. IEEE World Haptics.

[FPBOYBK90] FREDERICK P. BROOKS J., OUH-YOUNG M., BATTER J. J., KILPATRICK P. J.: Project gropehaptic displays for scientific visualization. In *SIGGRAPH '90* (New York, NY, USA, 1990), ACM, pp. 177–185.

[HBS99] HO C.-H., BASDOGAN C., SRINIVASAN M. A.: Efficient point-based rendering techniques for haptic display of virtual objects. *Presence: Teleoper. Virtual Environ. 8*, 5 (1999), 477–491.

[HDS96] HUMPHREY W., DALKE A., SCHULTEN K.: VMD – Visual Molecular Dynamics. *Journal of Molecular Graphics 14* (1996), 33–38.

[HKHJ07] H.BERMAN, K.HENRICK, H.NAKAMURA, JL.MARKLEY: The worldwide protein data bank (wwpdb): ensuring a single, uniform archive of pdb data. *Nucl. Acids Res. 35*, suppl1 (2007), D301–303.

[LRM*99] L. K., R. S., M. B., R. B., A. G., N. K., J. P., A. S., K. V., K. S.: Namd2: Greater scalability for parallel molecular dynamics. *Journal of Computational Physics 51* (1999), 283–312(30).

[Pha] Phantom haptic devices. http://www.sensable.com/haptic-phantom-desktop.htm.

[SB08] SUBASI E., BASDOGAN C.: A new haptic interaction and visualization approach for rigid molecular docking in virtual environments. *Presence: Teleoper. Virtual Environ. 17*, 1 (2008), 73–90.

[SGGS01] STONE J., GULLINGSRUD J., GRAYSON P., SCHULTEN K.: A system for interactive molecular dynamics simulation. In *2001 ACM Symposium on Interactive 3D Graphics* (New York, 2001), Hughes J. F., Séquin C. H., (Eds.), ACM SIGGRAPH, pp. 191–194.

[SWS*03] SANKARANARAYANAN G., WEGHORST S., SANNER M., GILLET A., OLSON A.: Role of haptics in teaching structural molecular biology. In *HAPTICS '03* (Washington, DC, USA, 2003), IEEE Computer Society, p. 363.

[TRC*93] TAYLOR R. M., ROBINETT W., CHI V. L., FREDERICK P. BROOKS J., WRIGHT W. V., WILLIAMS R. S., SNYDER E. J.: The nanomanipulator: a virtual-reality interface for a scanning tunneling microscope. In *SIGGRAPH '93* (New York, NY, USA, 1993), ACM, pp. 127–134.

[ZS95] ZILLES C. B., SALISBURY J. K.: A constraint-based god-object method for haptic display. In *IROS '95* (Washington, DC, USA, 1995), IEEE Computer Society, p. 3146.