EGVE Symposium (2008)
B. Mohler and R. van Liere (Editors)

Interactive and Accurate Collision Detection in Virtual Orthodontics

Maria Andréia F. Rodrigues†, Rafael S. Rocha, Wendel B. Silva

Universidade de Fortaleza (UNIFOR)
Av. Washington Soares, 1321
60811-905 Fortaleza–CE, Brazil

Abstract

An interactive computer-based training tool for using in Orthodontics is aimed at students and experienced professionals who need to predict orthodontic treatment outcomes, including the determination whether and where the teeth are coming into contact. Fundamental to achieve the best possible fit is the relative position of the teeth within the dental arch and with the opposing arches. In this paper we present the implementation and analysis of different types of discrete and continuous collision detection algorithms that meet performance goals, suitable for moving teeth simulation. The obtained results show that the collision detection algorithms implemented using Sphere-Trees provide quite acceptable accuracy while maintaining interactive visualization.

Categories and Subject Descriptors (according to ACM CCS): I.3.8 [Computer Graphics]: Applications I.3.7 [Three-dimensional Graphics and Realism]: Virtual Reality

1. Introduction

Interactive visualization of virtual applications represents a promising area with high potential of enhancing the training of health-care professionals. In this regard, virtual reality can enable medical personnel to practice surgical procedures on simulated individuals. In Orthodontics, for example, interactive computer-based training tools are aimed at students and experienced professionals who need to perform orthodontic treatment. Actually, the orthodontist and patient have a strong need for methods that enable them to compute realistic pictures of the expected teeth positioning to circumvent unexpectedly situations that may occur in practice.

Usually, treatment planning and the choice of a proper appliance model are based exclusively on clinician expertise [AMG*98]. A treatment is commonly used to obtain the proper position of the teeth within the dental arch and relative to the opposing dental arch, thus giving the correct occlusion with the best functional and aesthetic features [Mar93]. This procedure is driven by both the location and shape of the teeth. The determination of the fit between teeth and their proper position within the dental arch and with the opposing arches can be best estimated by defining the contact points between them. In our previous work [RSBN*07], we have proposed and implemented a prototype 3D simulation system for training and treatment planning in Orthodontics. The system includes simple teeth models and a graphical interface that can be used to assist clinicians in patient diagnosis, appliance design, and treatment planning. However, when operating the virtual treatment planning system for patients with a malocclusion, we need to accurately detect collisions between teeth to prevent them from penetration through each other. Moreover, when a user interacts with the 3D virtual environment, the computer-generated graphics display should be able to generate precise and realistic-looking views of the simulated tooth movement quickly enough to guarantee that the interaction feels responsive and natural.

Building upon our previous published work, this paper is devoted to the implementation and analysis of different types of collision detection algorithms for rigid objects that meet high performance goals, suitable for moving teeth simulation. For our experiments we used a Pentium Core Duo 3.0 GHz with 512 MB of RAM memory and an accelerator

† Corresponding author: mafr@unifor.br

graphics card. Two of the algorithms we have implemented support progressive refinement detecting collision (by using hierarchies of spheres) between successively tighter 3D approximations to teeth surfaces. In a first attempt to collision detection, we have used discrete tests among bounding circles to detect collisions between teeth. This approach has important limitations, for example, it does not take into account the shape of the teeth's root (it is common to have difficulties to move the crown of a tooth caused by a premature occurrence of collision between two adjacent teeth's roots). To improve upon this approach, we then used discrete intersection tests between Sphere-Trees [BO04, Bra03]. We have additionally applied interval arithmetic to bound the intersection tests between spheres as recommended in [Red04] to find out whether there exists an intersection within the time interval of the current time step in the simulation. We use this test to traverse the Sphere-Trees and, once the leaves are reached, the time of contact between the spheres is calculated through the linearly interpolation of the initial and final positions of the spheres.

An initial data set representing the teeth at their initial positions and respective constraints of repositioning are received as input of our 3D computer system, that includes a 3D mesh generator which is responsible for building a virtual customized model of the patient [RSBN*07]. A set of rules are applied to detect teeth collisions, which occur as the patient's tooth moves along the dental arch, from its initial position to its final position, in accordance with the movement constraints and treatment planning. Traditional times laws of physics are also applied to determine how to respond to a collision. Based on conservation of linear momentum, angular momentum and energy, the velocities after the teeth impact are calculated as well as the loss of kinetic energy. The results show that we have been able to demonstrate a behavior that closely replicates teeth movements in presence of collisions using Sphere-Trees. Further, the proposed collision detection algorithms accurately report the contact points and objective measurements, while guarantee interactive visualization (on average, 40 fps).

The rest of the paper is organized as follows. Section 2 covers related work. Section 3 describes the simulation of teeth movement. Section 4 presents the surface mesh representation of the teeth implemented in our prototype and some details about normal occlusion. Section 5 presents the main features (advantages and limitations) of the collision detection algorithms implemented. The results obtained with the computer simulations and their interpretations are presented in Section 6. Finally, Section 7 concludes the paper with some suggestions for future work.

2. Related work

Interactive visualization, surgical simulation, augmented and virtual reality systems are some examples of computer based applications that have made significant advances in treatment planning, computer-assisted diagnosis, and health care training. Similar to our work, some projects aim at developing simulation systems in which the anatomy of a patient (3D objects) can be viewed on a computer screen, and the user interface components can be interactively manipulated. The advantage of our system [RSBN*07] is that teeth movements are dynamically visualized instead of statically, with the ability to modify interactively, through the user interface panel, the loadings on a selected tooth or a group of teeth, and evaluate the results of changes in 3D for different orthodontic setups.

Some researchers have presented 3D computer-based orthodontic treatment tools, and mandible (lower jaw bone) movement simulators [BBV02, EMF*03]. These works are related to ours in that tooth movement is represented by a functional model composed of geometric restrictions on displacements in three-dimensions. However, none of the systems include interactive and accurate 3D collision detection methods, and some of them have the further disadvantage of being quite expensive commercial tools.

It is worth noting that when two or more objects interact with each other, collisions occur. Many algorithms have been proposed and developed for collision detection between 3D objects [Eri05]. Nevertheless, collision detection is still one of the bottlenecks of many interactive systems. To speed up the collision detection, a traditional approach is to separate the problem into two distinct phases: broad and narrow. The broad phase collision detection aims at efficiently cull out pairs of non-intersecting objects, discarding as many pairs of objects as possible. Simple geometric primitives, such as boxes and spheres, can be used as bounding volumes to quickly reject pairs of objects that do not collide. The broad phase collision detection can be further optimized using spatial partitioning data structures to decompose the virtual environment into groups of near objects. In the context of this work, the broad phase can be carried out using the spatial arrangement of teeth, that is, only neighbor teeth need to be tested for collision.

The collision detection process is refined in the narrow phase, when the remaining pairs of objects are analyzed more accurately. In general, bounding volume hierarchies are firstly used to narrow in on the regions of contact. These structures approximate the geometry of the objects in levels of detail. Then, an exact test may be performed. Sphere-Tree is a well-known example of bounding volume hierarchy [Hub96]. Hubbard described one of the first algorithms for the automatic generation of Sphere-Trees. Recently, Bradshaw and O'Sullivan have improved on Hubbard's work and developed a very efficient method for building Sphere-Trees that combines a set of sphere reduction algorithms [BO04]. In this work, the Sphere-Trees for the teeth were built using Bradshaw's Sphere-Tree Construction Toolkit [Bra03].

Collision detection algorithms can be classified into the following categories: discrete and continuous [Red04]. Dis-

crete collision detection algorithms sample the trajectory of the objects and report interpenetrations at the instants of sampling. Alternatively, continuous collision detection algorithms consider the real trajectory of the objects (or an approximation of it) to compute the first time of contact between colliding objects, as well as the contact state (collision points and normals). The problem with this approach is the difficulty of expressing the trajectories of objects as a closed function of time. Despite discrete algorithms fit better in interactive environments with high scalability, such as those composed of a large number of objects, continuous algorithms are more accurate and robust. As proposed in [Red04], we have applied interval arithmetic to bound the intersection tests between spheres to find out whether there exists an intersection within the time interval of the current time step in the simulations.

During the past years a number of approaches have been proposed for collision detection, a fundamental requirement for many varied applications, including computer applications in health care. However, few papers have been published on the level of interactivity achieved during the 3D simulations, and on the efficient implementation of data structures and algorithms to solve accurate collision detection problems in real-time virtual reality applications.

3. Simulation of tooth movement

The current implementation of the system simulates 3D teeth movements through time with respect to a fixed Cartesian frame located in the middle of the dental arch. It is designed and implemented to displace and rotate the teeth interactively to obtain the correct dental occlusion, based on the measurements and system of forces defined by the user in the interface panel. Any tooth (or a group of teeth) can be interactively "extracted" or selected through the user interface panel to apply loadings. Using the current and next tooth position in the dental arch, as well as the force system set up by the user, our system automatically calculates those new tooth positions (trajectories) that best fit the geometry of the virtual patient dental arch through linear interpolations, during the evolution of the orthodontic treatment (Figure 1). Each tooth has its own system of coordinates. A director vector between two neighboring teeth in the x direction can be found easily. In our simulator, a tooth can be also displaced (translated) into extraction spaces, following the direction of the dental arch, by calculating the director vector. For the z direction, the orthogonal vector is calculated via the cross product. This is useful, for example, when the user wants to perform inclinations on the teeth of the patient.

The simplest orthodontic movements [Mar93], such as tipping, occur about the centre of tooth resistance ($\frac{1}{3}$ from the root apex). Translations are modelled in our simulator by applying a bodily movement where the whole tooth structure is uniformly loaded. It is expected that during rotations, tipping also may occur. Extrusions and intrusions are both

Figure 1: *From the left to the right: tooth positions during movement in the dental arch.*

modelled by vertical movements where forces are used to move the tooth down or up, respectively. The center of resistance and center of rotation of the tooth are calculated automatically and they help to evaluate the effect of the force system on the tooth movement. A comprehensive description of the system can be found in [RSBN*07].

Collision detection is fundamental to interactive simulation systems and ensures that the properties of the solid real world are maintained. Therefore, we have also implemented collision effects among dental surfaces caused by tooth movements. The computer simulation imposes several restrictions that have a direct impact on our collision detection method. These include how many objects there are in a given orthodontic scenario, their relative sizes and shapes, level of detail, positions, if and how they move, and whether they are rigid or not. The problem of detecting when teeth collide, modeling the contact between them and determining an appropriate response, are all critical operations requiring careful coding to achieve a high and constant frame rate. The whole process has to operate under strict time and size restrictions to guarantee the correct real-time teeth collisions [Eri05]. Given the usual time and space trade-off, the detection accuracy must be balanced against computation time to meet performance requirements.

Based on the dental model occlusion determination, the final position of the teeth can be ascertained. In each frame of the simulation, we verify whether there is some force being applied to any tooth (the original user-defined force or the one that was propagated along the dental arch due to collisions). In the affirmative case, we first verify the orientation of applied force. Based on the displacement orientation found and current tooth position, we identify the "next tooth" (visible or not) in the dental arch. Following the orientation of the tooth displacement, we then search for the nearest visible tooth in the dental arch. The direction vector is then obtained from the center of mass of a tooth (which is being subjected to a force) toward the center of mass of the "next tooth" in the dental arch.

While not detected a collision between the geometry (bounding circles or Sphere-Trees) of the moving tooth and the next visible tooth, the moving tooth can still be translated and/or rotated, and the orthodontic wire model used is identified. If we change the orthodontic arch model used, the behavior of the tooth changes accordingly and this fact is

taken into account in our tooth movement simulator. Therefore, if it is necessary to apply a torque, the rectangular wire model should be used and the tooth rotates with the center of rotation in this case located at its crown. However, if the wire model is rectangular and the tooth is subjected to a force, this tooth can translate in the x, y, and z-directions. Finally, if the wire model is spherical and the tooth is subjected to a force, it can only rotate. The center of rotation, in this case, is taken to be a position located at $\frac{1}{3}$ of its root length. In the negative case (ie, a teeth collision is detected), the moving tooth stops and the force being applied to it is partially propagated to the next visible tooth (that suffered the collision), by simulating a collision effect among teeth and thus successively, while there is enough force to move any tooth as well as colliding teeth in that direction.

4. Surface mesh representation of the teeth

By creating surface meshes (on the left of Figure 2), we have implemented the four different types of teeth present in the mouth of an adult human: incisors (central and lateral), canines, premolars (first and second), and molars (first, second, and third). In our system, each tooth consists of a crown and one or more roots. The crown is the functional part of the tooth that is visible above the gum. The root is the unseen portion that supports and fastens the tooth in the jawbone. Roots vary from single to quadruple and were also implemented due to their importance as they move during treatment (tooth movements are also restrained by root shape and position). Approximately 125 vertices are used in our interactive 3D system to model the skeletal shape of each tooth.

According to Angle's Classification Method [Mar93], in a normal occlusion the cusp of the upper first molar should rest on the groove of the mandibular first molar. Also, in the normal condition the teeth in the maxilla (superior maxillary bone) slightly overlap those of the mandible (inferior maxillary bone) both in front and at the sides (normally a value around 2mm). Any variations from this, results in malocclusion types.

5. Collision detection

In this work we have rigid objects (teeth), each of which is composed on average of approximately 180 polygons. Thus, when two teeth collide, a brute force collision detection algorithm would be executed 180x180 times in most cases (polyhedron-polyhedron collision test). One way to speed up it is to use bounding volumes. In the next sections, three different approaches using bounding volumes are implemented and analyzed to detect teeth collisions: two based on discrete algorithms, and one based on a continuous algorithm.

5.1. Discrete Algorithms

The first discrete algorithm implemented is a very simple one. Initially, in a pre-processing time, we assign a bounding circle to each tooth. These circles are defined in the $x-z$ plane (see the coordinate system in Figure 1, where the $x-z$ plane is horizontal and the y-axis points up) and can be thought of as being upper bounds to the projection of the teeth in that plane. Every time the moving tooth simulator updates the position of a tooth, the position of its bounding circle is also updated, thus, the bounds are always valid. As long as we assume the teeth move only in the $x-z$ plane, this approach guarantees that an overlap between two teeth will take place only if an overlap between its bounding circles also takes place. Although cubes can be thought as a closer shape of some teeth crowns, bounding circles have proved to be more accurate than bounding boxes for teeth collision detection (in our previous work [RSBN*07], aligned bounding boxes were implemented and, although they were able to guarantee interactive visualization, they did not produce accurate results for teeth collision detections, particularly for those which occur in the curved areas of the dental arches).

Once the simulation system has been forwarded after taking one time step, we check for the existence of collisions. The collision detection is a simple matter of testing all the bounding circles for intersection. The overlap between two circles (one with center c_1 and radius r_1, and another with center c_2 and radius r_2) occurs when the distance between their centers is less than or equal to the sum of their radii:

$$\|c_1 - c_2\| \leq (r_1 + r_2)^2 \tag{1}$$

One possible optimization, to avoid testing all pairs of teeth (during the broad phase), is to test only the moving tooth against its next neighbor tooth, in the direction of its velocity. In this first attempt to interactive collision detection we have not implemented a narrow phase, so we issue the teeth whose bounding circles overlap as colliding teeth. Although this approach produces interactive performance, it has also important limitations. First, the bounding circles give a limited geometric accuracy, making some simulations not fully accurate. Further, it does not take into account the root of the teeth during collision checking (sometimes the tooth stop moving not because there is not space available between two adjacent crowns, but due to the occurrence of a collision between the teeth's roots). Another problem is that bounding circles are not useful for testing the upper jaw's teeth against the lower jaw's ones in presence of collisions. This kind of test can be used for verifying malocclusion problems, for example. An even more serious problem is the discrete nature with which this approach detects collisions. One may have observed that the collision checks are done once we have sampled all the teeth positions. This approach can sometimes cause somewhat large interpenetrations between the bounding circles, and, therefore, between the teeth, leading to inaccurate simulations. One solution to the latter problem is to decrease the time step, making the teeth move small amounts and increase the

collision detection accuracy. Another possible solution is to use backtracking to go back in time and find the first time of collision. One drawback of both solutions is that they slow down the simulation and do not fully eliminate the problem.

The second discrete algorithm we have implemented overcomes some of the problems mentioned previously, through the use of bounding volume hierarchies. A first way to speed up the previous collision algorithm is to replace the bounding circles with Sphere-Trees [BO04]. These are hierarchies of bounding spheres used to approximate the 3D teeth's geometry. The hierarchy begins with the tooth's bounding sphere as the root of the tree, and it better approximates the tooth shape as one goes down the tree structure. We have chosen Sphere-Trees due to the following reasons: the intersection method between spheres is simple and fast, with low memory requirement; spheres have rotational invariance; there are a number of published algorithms for automatically building Sphere-Trees [Hub96, Bra03, WZS*06]; and Sphere-Trees have proved capable of approximating the teeth geometry in this work very accurately. In particular, we have used Bradshaw's Sphere-Tree Construction Toolkit [Bra03] to automatically build Sphere-Trees for the teeth. All the Sphere-Trees were built using a branch factor of 8 and a tree depth of 4, totalizing 422 spheres on average in the 4^{th} level. The resulting Sphere-Trees are depicted in Figure 2.

To detect a collision between two Sphere-Trees it suffices to traverse both trees narrowing in on the pairs of overlapping leaf spheres (one of each tree). One way of collecting all overlapping leaf spheres is to traverse both trees simultaneously. Every time two nodes overlap, the children of one node are tested against the other node. This test is repeated down to the leaves. If there exists at least one overlapping leaf pair, it will be found and the teeth will be considered to be in a collision state. To test whether a pair of spheres overlaps, the same discrete test presented in Eq. 1 can be used.

The replacement of the circle-circle intersection test with the intersection test for Sphere-Trees allowed us to carry out more accurate collision detections and, hence, to obtain more accurate simulated results. Additionally, this approach can also handle collisions among the upper jaw's teeth and the lower jaw's teeth. Furthermore, if we highlight the overlapping leaf spheres only, it can be useful to visually spot the regions of contact between a pair of overlapping teeth. Despite this latter approach be very accurate, it is still a discrete collision detection method and, therefore, teeth interpenetrations generally occur. The next section presents an extension of the latter method to continuously detect collisions between Sphere-Trees.

5.2. Continuous Algorithm

In the discrete method described previously, overlapping between two spheres can be missed if the spheres are too small and their velocity is relatively high. One possible solution to this problem is to determine a lower bound to the left term used in the sphere-sphere intersection test $\|c_1 - c_2\|^2$ for the time interval. If this lower bound is less than the right term of the inequality $(r_1 + r_2)^2$, which does not need to be bounded because it is constant, then it is guaranteed that the spheres will not overlap within the time interval. Otherwise, an intersection may or may not occur (it is important to notice that false positives may occur, but never the opposite). In this way, the spheres are conservatively tested for intersection (we do not miss overlaps). We use interval arithmetic to calculate this lower bound as suggested in [Red04].

We have replaced the discrete sphere-sphere intersection test with the test based on interval arithmetic, consequently, the algorithm that traverses the Sphere-Trees uses the latter test to search for collisions. Whenever this algorithm encounters two leaves, it linearly interpolates the initial and final positions of the spheres to determine the first time of contact between them [Len04] (see Eq. 2).

$$t = \frac{-(AB) - \sqrt{(AB)^2 - B^2[A^2 - (r_1 + r_2)^2]}}{B^2} \qquad (2)$$

Where A and B are the position vectors calculated as follows: $A = c_1 - c_2$ and $B = (c_1^{'} - c_1) - (c_2^{'} - c_2)$. The variables c_1 and c_2 are the initial positions of sphere 1 and 2, respectively; and $c_1^{'}$ and $c_2^{'}$ are the final positions of sphere 1 and 2, respectively. This ultimate test effectively discards spheres that do not overlap, as opposed to the previous test using interval arithmetic. We collect the computed time t for all the overlapping leaf spheres. The least time within the range [0, 1) indicates the first time of contact between the Sphere-Trees being tested. We use this time then to approximate the first time of contact between the teeth. If the times found all lie outside the range [0, 1), then the spheres do not collide within the time interval. One important thing to notice is that when a collision is detected, the tooth movement simulator receives back the first time of contact between the colliding teeth. With this information, the simulator is able to reposition the teeth in a state where they have just touched each other. Using this approach we have obtained even more accuracy in the collision detection and, consequently, in the resulting animations.

As a final remark, it is interesting to know that the spheres' size of the Sphere-Trees generated by the Bradshaw's algorithm [BO04] is unpredictable. If the aim is just the visualization of the teeth contact area, then a better approach would be to use Sphere-Trees generated from octrees [Hub93], essentially because spheres at the same level are equally sized due to the octree structure. Therefore, the sphere leaves can be shown to visually approximate the 3D contact area, as shown in the end of the next section.

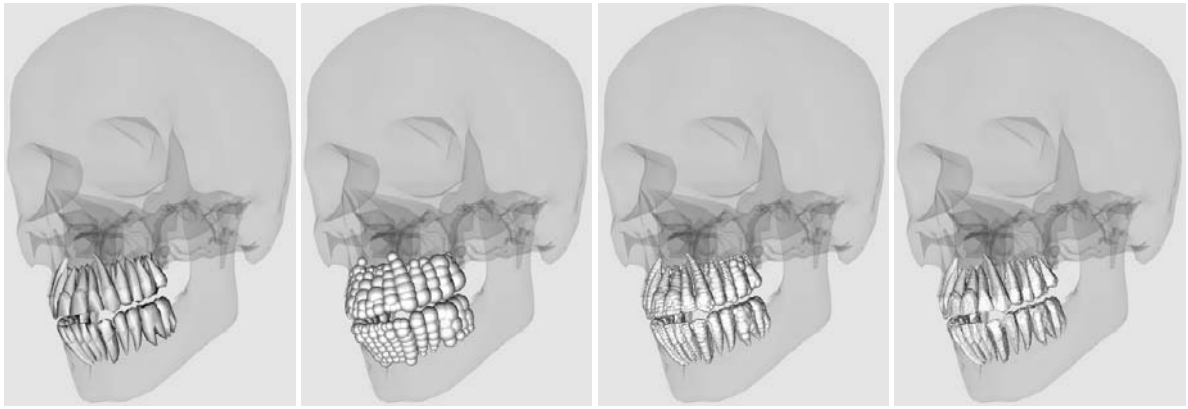

Figure 2: *From the left to the right: mesh representation of the teeth in the mouth and depth levels 1, 2, and 3, respectively, of the Sphere-Tree generated for teeth collision detection in our prototype (level 0 is not shown in this figure).*

6. Analysis of results

Analysis of results is based on the degree of fit between experimental orthodontic treatments and simulated case studies. We used cephalometric measurements and dental cast data taken during one-year follow-up orthodontic treatments to pre-validate our simulator. Collision effects acting on different teeth were observed and their behaviors were simulated. After the teeth collisions were detected and the final teeth positions were decided, we evaluated the relationship between the teeth's contact area in different regions of the dental arch by using a simple method that provides an objective measure of tooth-to-tooth distances. The horizontal tooth-to-tooth distances are calculated as the distance between the geometric centers of each adjacent tooth crown (say, c_1 and c_2) in the plane $x-z$, where $c_1 = (x_1, y_1, z_1)$ and $c_2 = (x_2, y_2, z_2)$, as follows: $d_{c_1,c_2} = \sqrt{(x_1 - x_2)^2 + (z_1 - z_2)^2}$. Figures 3, 4, and 5 show the resulting tooth-to-tooth distances calculated from discrete intersection tests between bounding circles, with discrete tests between Sphere-Trees, and with continuous intersection tests between Sphere-Trees, respectively.

More specifically, Figures 3 and 6(a) show the simulated teeth distances and collision results, respectively, for the treatment that required the extraction of the 1st premolars on both sides of the maxilla and mandible (Case 1). The loading force was applied to the 1st upper and lower molars. The 1st molars and 2nd premolars moved in the dental arch, but the incisors and lateral incisors teeth did not. This fact was also observed in our simulations due to occurrence of collisions, leaving no more space left for teeth movement. Figures 4 and 6(b) show the distances and collision results, respectively, for the treatment that required the extraction of the 2nd premolars on both sides of the maxilla and mandible (Case 2). Likewise Case 1, our simulations demonstrated that the 1st molars and the 1st premolars moved in the dental arch, and the canines and lateral incisors stayed at the same initial position. Finally, Figures 5 and 6(c) show the teeth distances for the treatment that required the space-filling between the upper and lower central incisors (Case 3). In this case, the loadings were applied to all four incisors teeth, but in opposite x directions. In the same way as Case 1 and 2, our graphics simulation succesfully positioned the teeth closely to each other in the dental arch in the presence of collisions. Based on the teeth distance measurements and collision results obtained in our simulations, in the following we compare the accuracy of the three implemented collision detection algorithms.

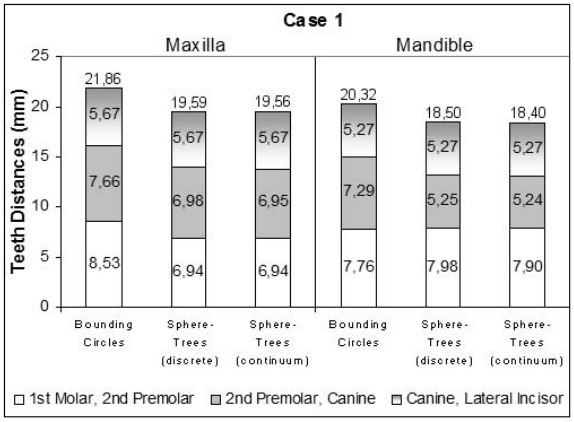

Figure 3: *Distances between the geometric centers of each adjacent tooth crown for Case 1.*

Most orthodontists assume in their routine practice that a distance between two adjacent teeth greater than 1mm cannot be neglected. Assuming this fact, in the simulations shown in Figures 3, 4 and 5, the collision detection algorithms using Sphere-Trees (discrete and continuous algo-

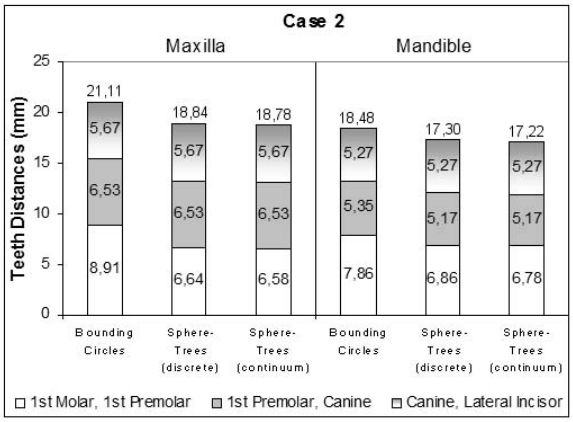

Figure 4: *Distances between the geometric centers of each adjacent tooth crown for Case 2.*

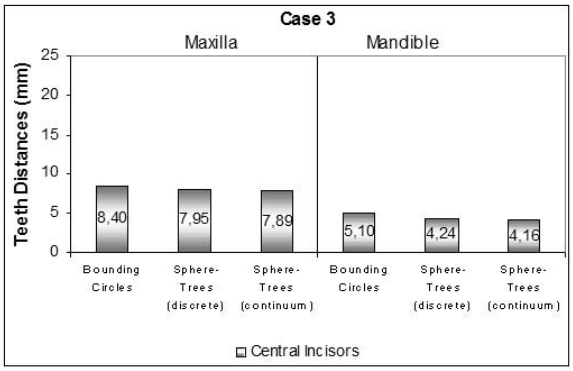

Figure 5: *Distances between the geometric centers of each adjacent tooth crown for Case 3.*

rithms) achieved higher accuracy (smaller tooth-to-tooth total distance values) than the discrete algorithm using bounding circles (remind that the teeth distances are calculated between the geometric centers of each adjacent crown). Additionally, although the tooth-to-tooth total distance values calculated using discrete tests between Sphere-Trees are very similar to the ones calculated from continuous tests, the latter improves visual accuracy as shown in (a), (b), and (c) of Figure 6.

Inspired by the recent work of Wang *et al.* [WZS*06], who have presented a new approach for computing sets of spheres to approximate solid objects based on what the authors call the sphere outside volume (volume inside the sphere, but outside the solid object, named SOV), we have implemented an approach for approximating solid objects which, similarly to their work, uses the sum of all SOVs, given a specific number of spheres. Although Bradshaw's Sphere-Tree Construction Toolkit has been used to build the

(a) (b) (c)

Figure 6: *Graphical results for Cases 1, 2 and 3.*

Sphere-Trees, we have applied the sum of all SOVs of the Sphere-Trees' leaves to measure how accurate the geometric teeth approximation is, by calculating a geometric error that is measured by the relative outside volume (the sphere set's outside volume divided by the original mesh's volume).

One limitation of this metric is that it sums the outside volume in the intersection of two spheres twice. This is not a problem for minimizing the sum of SOVs and finding good approximations, but it makes difficult to use the resulting value *per se* to measure how good the approximation is. Thus, we compare the geometric accuracy between Sphere-Trees built using octrees with 6 depth levels and using Bradshaw's algorithm with 4 levels (the latter is shown in the right-side of Figure 2). The octree algorithm results in 1.25, on average, with 0.89 and 2.12 as minimal and maximal values, respectively. On the other hand, Bradshaw's algorithm results in 0.16, on average, with 0.07 and 0.51 as minimal and maximal values, respectively. This demonstrates the superiority of Bradshaw's algorithm even when deeper octrees are used. Recall that Figure 2 shows Sphere-Trees built by Bradshaw's algorithm for all teeth. Note that, also visually, these Sphere-Trees rapidly converge to the teeth geometry.

Besides a substantial increase in terms of accuracy (Sphere-Trees are computed during pre-processing and used to approximate the whole 3D teeth geometry), while providing highly interactive visualization, the additional benefit of our collision algorithms based on Sphere-Trees is that dental occlusions between the meshes of the upper and lower dental arches can be visualized, by highlighting the 3D contact points of the teeth. In our simulations, Sphere-Trees with 3 and 4 depth levels generated interactive visualization and accurate collision results. Moreover, the Sphere-Tree with 4 levels achieved a very high degree of fit for teeth shape representation and, consequently, has demonstrated to be more recommended for representing realistic teeth geometry and their 3D interactions in Orthodontics.

Finally, to assess the quality of occlusion between the teeth of the maxillary and mandibular arches, the orthodontist has to estimate distances between specific points located on the teeth of both arches [AMG*98]. Distance measurement is currently based on the observation by the orthodontist of a plaster cast model of the mouth. There are many

problems with cast analysis: manual measurements are both inaccurate and one-dimensional, so the estimation of occlusion is generally biased; the diagnosis does not rely on objective measurements but rather on a subjective interpretation of the shape of the teeth model; and the influence of tooth root collisions can not be analyzed and visualized. In our prototype, when teeth's roots collide, the teeth models should be rotated until their roots are in the proper vertical position and orientation. The two dental arches are positioned together, and the teeth models are moved slightly to analyze whether (and how) the upper and lower arches fit together (occlusion). The teeth models are then observed from the side and the occlusion's type can be verified. In this way, the occlusal contact areas on any teeth are calculated and the contact points are depicted using a different colour. Figure 7 shows an example we generated for representing the dental occlusion, focusing on the contact areas on the teeth.

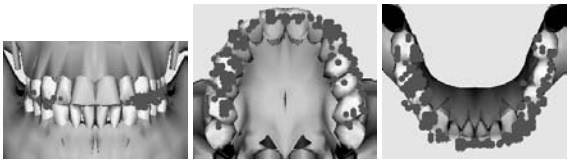

Figure 7: *The 3D occlusal contact areas on the teeth are highlighted (frontal, maxillary, and mandibular views on the left, middle, and right, respectively).*

7. Conclusion and future work

We have successfully implemented and tested different types of collision detection algorithms that meet performance goals, suitable for moving teeth simulations. The obtained results show that the collision detection algorithms using Sphere-Trees provide quite acceptable accuracy (smaller tooth-to-tooth distance values) while maintaining interactive visualization (on average, 40 fps, which makes possible to be used in a training system for dental practice), whereas the one using bounding circles is to be rejected. In particular, Sphere-Trees with 4 levels obtained a very high degree of fit for teeth shape representation and, consequently, a more accurate method for collision detection and for generating realistic teeth animations. Besides providing a fast method for collision detection, the continuous collision detection algorithm using Sphere-Trees achieved even more accuracy and can be used to evaluate other problems in orthodontics like whether (and how much) abrasion occurs when the teeth collide during the occlusion.

For future work we aim at improving the system by developing more robust teeth collision responses doing continuous simulations. Extensive clinical evaluation will be a slow process because of the long time of orthodontic treatment and the fact that, for ethical reasons, it is not possible to experiment with the treatment regimes.

8. Acknowledgments

Maria Andréia F. Rodrigues is supported by the Brazilian Agency CNPq under grant No 303046/2006-6 and Rafael S. Rocha by CAPES. We are also grateful to Central Geradora Termoelétrica de Fortaleza - ENDESA for its financial support.

References

[AMG*98] ALCAÑIZ M., MONTSERRAT C., GRAU V., CHINESTA F., RAMON A., ALBALAT S.: An advanced system for the simulation and planning of orthodontic treatment. *Medical Image Analysis 2*, 1 (1998), 61–79.

[BBV02] BISLER A., BOCKHOLT U., VOSS G.: The virtual articulator-applying VR technologies to dentistry. In *Proceedings of the 6th IEEE International Conference on Informatics and Visualization* (2002), pp. 600–602.

[BO04] BRADSHAW G., O'SULLIVAN C.: Adaptive medial-axis approximation for sphere-tree construction. *ACM Transactions on Graphics 23*, 1 (2004), 1–26.

[Bra03] BRADSHAW G.: Sphere-tree construction toolkit. http://isg.cs.tcd.ie/spheretree/, 2003.

[EMF*03] ENCISO R., MEMON A., FIDALEO D. A., NEUMANN U., MAH J.: The virtual craniofacial patient: 3d jaw modeling and animation. In *Proceedings of the 11th Medicine Meets Virtual Reality* (2003), pp. 65–71.

[Eri05] ERICSON C.: *Real-Time Collision Detection*. Morgan and Kaufmann Publishers, 2005.

[Hub93] HUBBARD P. M.: Interactive collision detection. In *Proceedings of the IEEE Symposium on Research Frontiers in Virtual Reality* (1993), pp. 24–31.

[Hub96] HUBBARD P. M.: Approximating polyhedra with spheres for time-critical collision detection. *ACM Transactions on Graphics 15*, 3 (1996), 179–210.

[Len04] LENGYEL E.: *Mathematics for 3D Game Programming and Computer Graphics*. Game Development Series, 2nd Edition, 2004.

[Mar93] MARCOTTE M. R.: *Biomechanics in Orthodontics*. BC Decker Inc, 1993.

[Red04] REDON R.: Continuous collision detection for rigid and articulated bodies. In *ACM SIGGRAPH Course Notes* (2004).

[RSBN*07] RODRIGUES M. A. F., SILVA W. B., BARBOSA-NETO M. E., GILLIES D. F., RIBEIRO I. M. M. P.: An interactive simulation system for training and treatment planning in orthodontics. *Computers and Graphics 31* (2007), 688–697.

[WZS*06] WANG R., ZHOU K., SNYDER J., LIU X., BAO H., PENG Q., GUO B.: Variational sphere set approximation for solid objects. *The Visual Computer: International Journal of Computer Graphics 22*, 9 (2006), 612–621.