Interactive Repositioning of Bone Fracture Segments

Michael Scheuering[1,2], Christof Rezk-Salama[1],
Christian Eckstein[2], Kai Hormann[1],
Gnther Greiner[1]

[1]University of Erlangen-Nuremberg, Computer Graphics Group
Am Weichselgarten 9, D-91058 Erlangen, Germany
Email: scheuering@informatik.uni-erlangen.de

[2]Siemens Medical Solutions
Henkestr. 127, D-91054 Erlangen, Germany

Abstract

This paper presents an application for semi-automatic repositioning of bone fractures that allows the merging of several fragments. This application has been developed with regard to orthopaedic surgeons who want to simulate the position and orientation of bone fragments preoperatively.

The interactive algorithm features volumetric collision detection for intuitive navigation and coarse manual positioning. Additionally, an optimization process for the mathematically exact repositioning of the bone fragments is implemented.

In order to accelerate the volumetric collision detection, octree structures are used that are efficiently implemented as an hierarchy of oriented bounding boxes (OBB). The collision test uses the *separating axis theorem* for a fast traversal of the octree.

To improve the manual part of the repositioning process, the principal axes of each fragment are precalculated initially. Subsequently, the fragments are pre-justified by the user. Finally, an optimization process is performed based on Powell's algorithm for multidimensional minimization. The optimal position of the bone fragments is determined by the use of a voxel-based metric, that exploits the same bounding box hierarchy.

1 Introduction

The surgical treatment of bone fractures is a typical problem in medical practise. For complicated fractures, however, the repositioning of several bone fragments during a surgical intervention is a complex task, which has extremely negative influence on the duration of the intervention as well as on the patient's convalescence. In order to circumvent this, the idea is to simulate the surgery in a purely virtual environment based on tomographic data, which is individually acquired for each patient.

Taking into account the possibilities of virtual reality applications, one can assist the surgeon preoperatively by the use of a semi-automatic repositioning system that allows to glue together single fragments. This problem is very similar to 3D puzzle applications as presented in [13]. Here, the authors describe to semi-automatically assemble the Parthenon at the Acropolis of Athens by the use of scanned stone fragments.

In Section 2, we give an overview of the process of the repositioning of bone fragments. This includes the segmentation of the fragments (Section 3) and their visualization techniques (Section 4). In Section 5 we describe the interaction of volumetric datasets by the use of collision detection techniques. Afterwards, the whole process of repositioning of the fragments is automatized by introducing Powell's algorithm (Section 6). At the end, we present the results (Section 7) and conclude in Section 8.

2 Repositioning of Bone Fragments

The repositioning procedure can be described as a three tier approach. As an initial step a semi-automatic segmentation of the data must be performed in order to classify individual bone fragments. This segmentation is the basis for a manual repositioning by the physicist. This requires a method for interactive visualization of the volume data, that allows to pick and move individual sub-

volumes. A fast and robust algorithm for collision detection is necessary for intuitive navigation within 3D space. After manual repositioning an optimization process is performed that efficiently computes the optimal position. In the following each step of this procedure will be explained in detail.

3 Segmentation

Manual repositioning requires the ability to independently move several bone fragments. This implies the necessity to divide the original CT data set into sub-volumes, that can be transformed separately. In addition to this coarse subdivision an explicit voxel-based segmentation is required in order to efficiently perform collision detection (see Section 5.2). Since bone fragments are easily classified from CT data by applying a simple threshold to the Hounsfield scale, an inexpensive volume growing technique was chosen. This allows a separation of the bone structure from surrounding tissue. However, the results obtained by this semi-automatic approach must again be controlled and corrected manually.

4 Visualization

A variety of efficient direct volume rendering algorithms have been developed in recent years. These solutions range from pure software approaches [10] to the development of special purpose hardware [14, 12], and to methods which exploit the steadily evolving features of general purpose graphics hardware [2, 1, 18, 11, 16].

The high computational cost for volume rendering is on one hand caused by the huge number of spatial interpolation operations. On the other hand, for large volume data the limited memory bandwidth soon becomes the bottleneck. As a consequence, several efficient algorithms developed in recent years store the volume data in local video memory and exploit general purpose texturing hardware for fast interpolation.

Although these texture-based methods are the most promising implementations on low cost hardware, these approaches are less applicable to display multiple independent volumes, which interpenetrate each other. Displaying multiple data sets by blending a stack of texture slices back-to-front requires a complex depth sorting algorithm and also makes an efficient texture memory management extremely difficult.

As an alternative, the commercially available volume rendering library *VGL* provided by *Volume Graphics* can efficiently handle arbitrary intersecting volumes. Due to the applied pure software ray-casting approach, the image quality and the achieved frame rate represent a tradeoff compared to state-of-the-art texture based approaches. However, *VGL* supports different rendering modes (ray-casting with local illumination, isosurface display, maximum intensity projection) and is available on a variety of platforms (Win32, Linux, Solaris, IRIX) including multiprocessors.

5 Interaction and Collision Detection

To guarantee encouraging results, techniques that detect overlaps of the fragments during navigation are necessary. Thus, the key issue for repositioning of bone fragments is *collision detection*, which is a fundamental problem in 3D interactive applications.

In this context, most of the previous works focus on the interaction of polygonal objects, such as polygon meshes, polyhedra or splines (e.g. [13, 3, 6]). Unfortunately, those approaches suffer from the fact, that explicit surfaces are required, which can lead to unprecise results. A more natural approach for collision detection would be a solution for volumetric objects, since the containing voxels can describe different interaction behaviors of the included structures. A few papers for *volumetric* collision detection have been published in recent years [7, 9, 8, 5]. Those approaches try to solve the problem by introducing hierarchical structures such as *octrees* or *sphere trees*. The bounding volumes mostly include axis-aligned bounding boxes (AABB) or oriented bounding boxes (OBB) including different interaction methods of the voxels.

In this paper, some ideas for solving the volumetric collision detection problem are adopted from He *et al.* [8]. Here, the authors present a work based on octree and sphere-tree hierarchies with OBBs. Additionally, for the interaction of voxels of two volumetric objects a probability model of each object is defined using a predefined probability map. Thus, each point inside a volume is assigned a value in the range $[0, 1]$, that can be seen as the probability that a surface crossing that point exists. A possible interpretation for the probability map, men-

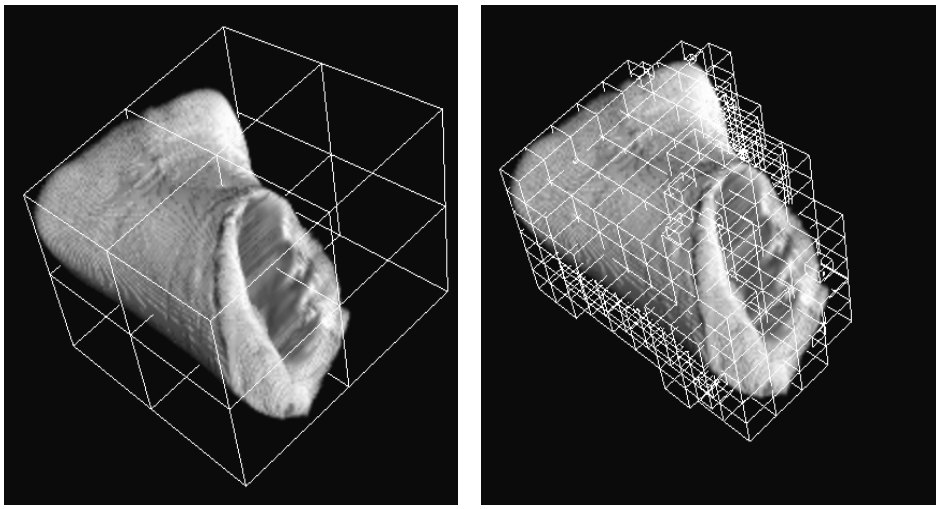


Figure 1: Different levels of the hierarchy of a cow foot bone: hierarchy depth 1 (left) and depth 3 (right).

tioned in [8], is to use opacities in the context of volume rendering. In general, the collision probability can then be calculated by the multiplication of the surface probabilities of two object at that certain point.

5.1 Building the Hierarchy

In order to create the hierarchy for accelerated collision detection, an octree of oriented bounding volumes is used, since the basic primitives in our datasets are the rectilinear voxels.

For constructing the hierarchy, a two-stage algorithm is used as described in [8]. Firstly, the algorithm starts bottom-up from the leafs and merges the eight neighbouring nodes. Since this approach also includes nodes that do not contain any relevant information (e.g. empty voxels), a second step is added to reduce the overhead of collision detection by pruning the tree top-down.

Thus, each node N^i in the hierarchy is assigned a value interval $[N^i_{\rho\ min}, N^i_{\rho\ max}]$, that represents the minimum and maximum collision probabilities of that node. Additionally, the user specifies a solid threshold τ_{max} and an empty threshold τ_{min}, that shorten the tree. In the second stage, which is recursively defined, the algorithm starts at the root node whereas breadth-first traversal is used. By taking N^i as the current node, the following decisions have to be taken [8]:

- If $N^i_{\rho\ max} < \tau_{min}$, delete both N^i and all its childs.
- If $N^i_{\rho\ min} > \tau_{max}$, delete all the childs of N^i.
- For a direct child N^i_1 of N^i, if N^i_1 has only one child N^i_2, then delete N^i_1 and make N^i_2 to be the child of N^i.

Using this second stage of the algorithm reduces the tree to those nodes, that contain only voxels with $\tau_{min} < \rho < \tau_{max}$. Changing the values of τ_{min} or τ_{max} by the user increases or decreases the complexity of the hierarchy. Figure 1 shows the results of a cow foot bone, which presents different levels of the hierarchy.

5.2 Collision Detection

In the previous section, we described how to realize a reduced hierarchy of the volumetric objects. Thus, we now have to use efficient algorithms for interference detection of the bounding boxes. In [17] the authors give a cost function for two large models and their hierarchical representation when the objects' interference has to be detected:

$$T = N_v \times C_v + N_p \times C_p \qquad (1)$$

T describes the total cost function for interference detection, N_v is the number of bounding volume pair overlap tests, C_v are the costs of testing a pair of bounding volumes for overlap, N_p is the number of primitive pairs tested for interference and C_p

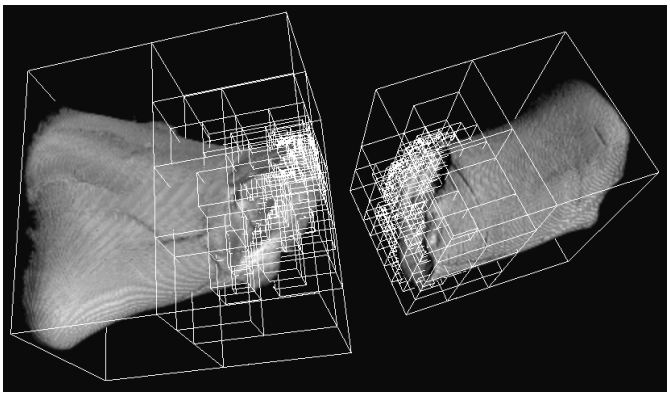


Figure 4: Two colliding bones showing the depth-first traversal. After interference, the bones are torn apart, so that the traversal is visible.

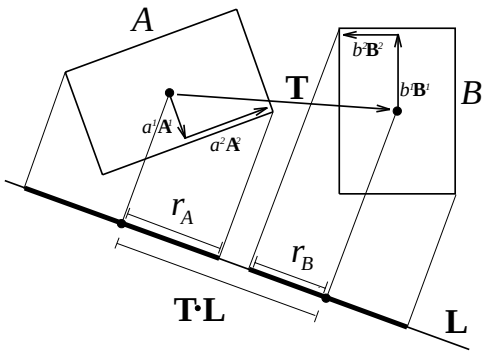


Figure 2: $\mathbf{L}$ is a separating axis of OBBs A and B because A and B become disjoint intervals under projection onto $\mathbf{L}$.

```
bool collision(N n1, N n2)
{
 if ¬testCollision(n1,n2)
    return false
 if n1.isLeaf() ∧ n2.isLeaf()
    return true
 for i=1,..,n1.getChildCount()
    if collision(n1.getChild(i),n2)
       return true
 for i=1,..,n2.getChildCount()
    if collision(n1,n2.getChild(i))
       return true
 return false
}
```

Figure 3: Pseudo code for depth-first traversal.

are the costs of testing a pair of primitives for interference. Using this formula, Gottschalk *et al.* [7] implemented an approach, called the *separating axis theorem*, for rapid interference detection, taking (1) into account. Given two OBBs, A and B, with B placed relative to A by rotation and translation (cf. Figure 2). Additionally, given r_A and r_B as the radii of A's and B's interval, the following inequation has to be fulfilled for disjoint intervals [7]:

$$|\mathbf{T} \cdot \mathbf{L}| > \sum_i |a_i \mathbf{A}^i \cdot \mathbf{L}| + \sum_i |b_i \mathbf{B}^i \cdot \mathbf{L}|. \quad (2)$$

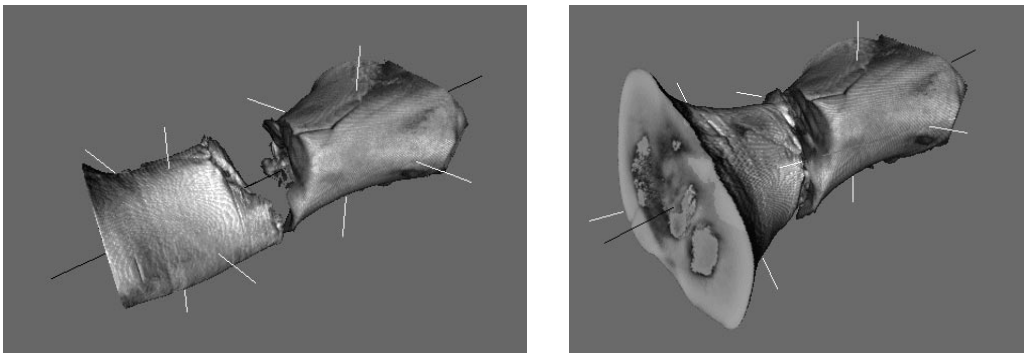
In [7], further simplifications are given for the collision test in (2).

Using the hierarchy defined in Section 5.1 and the collision test presented above, the algorithm for interference detection using depth-first traversal is described in Figure 3. As an alternative, breadth-first traversal is also possible as presented in [4].

Using depth-first traversal, the results can be seen in Figure 4. Here, two bone segments interfere and the hierarchy is traversed. Afterwards, the bone fragments are torn apart so that the traversed cells become visible.

6 Optimization Process

Based on the constellation of bone fragments obtained by manual repositioning, we can now apply an automatic fitting procedure for optimally align-



(a) (b)

Figure 5: Finding a suitable initial position for the optimization process: first the principal axes of both bone fragments are aligned (a), then the left data set is moved along and rotated around the common axis as close to the other fragment as possible (b).

ing each pair of sub-volumes representing fragments which had once been adjoint. The general idea is to fix one of the two fragments and move the other one by translations and rotations until a criterion that measures the gap between the two fragments becomes minimal.

Considering the criterion as a scalar function that depends on the translation and rotation parameters of the second fragment's movement, this approach can be understood as a 6-dimensional optimization problem. Among the many numerical methods that exist for solving such problems we chose *Powell's method* [15] which performed well as long as a good starting value could be provided. In order to find such a suitable initial position we took the following approach.

Firstly, we compute the *principal axes* of both sub volumes. The principal axis $A(\vec{n}) = \{\lambda\vec{n} : \lambda \in \mathbb{R}\}$ of a segmented volume data set with n voxels represented by their centers $r_i \in \mathbb{R}^3$, $i = 1, \ldots, n$, is the one that minimizes the average ℓ_2 distance

$$D(\vec{n}) = \frac{1}{n}\sum_{i=1}^{n} \mathrm{dist}(r_i, A)^2.$$

It is defined by the normalized eigenvector $\vec{n}$ belonging to the greatest eigenvalue of the 3×3 matrix

$$Q = \sum_{i=1}^{n} (r_i - c)(r_i - c)^t,$$

where $c = \frac{1}{n}\sum_{i=1}^{n} r_i$ is the volume's centroid. Secondly, we rotate the second fragment such that the principal axes of both volumes align. Thirdly, the second data set is translated along and rotated around this common axis until a reasonable starting position for the numerical optimization process is found (cf. Figure 5).

Now that both fragments are already close together we circumscribe the region of the bone fracture with a box called the *Volume of Interest* (VOI) (cf. Figure 6). Counting the number of voxels within the VOI that

a) belong to the first fragment,
b) belong to the second fragment,
c) belong to both fragments, i.e. both data sets overlap in these voxels,

yields the numbers n_1, n_2, and n_I. The objective

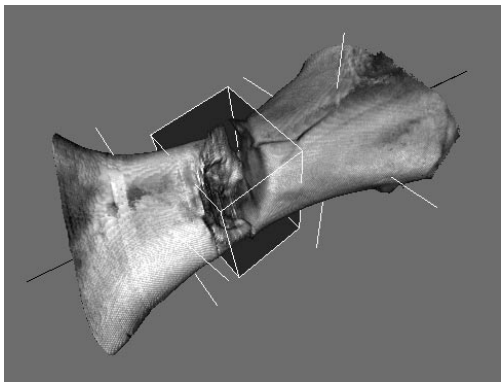


Figure 6: Volume of Interest of two bone fragments.

function measures the remaining gap between the two fragments by computing

$$\frac{2n_I - n_1 - n_2}{n_{\mathrm{VOI}}},$$

where n_{VOI} is the total number of voxels in the VOI. Note that since the first fragment is fixed n_1 is constant. Minimizing this quantity is equivalent to placing the second data set such that the VOI is maximally dense while penalizing overlapping regions.

7 Results and Discussion

The presented approach was implemented as prototype application. Throughout our experiments datasets of a tibia fracture, a broken cow foot and several hand bones are used. In order to evaluate the presented approach, we measured the time, both for the creation of the hierarchies and for the optimization process. According to the hierarchy, we evaluated three datasets, taken with CT or C-arm modalities. The datasets were segmented as presented in Section 3. Table 1 shows the time for the calculation of the hierarchical octrees for the tibia fracture, the cow foot and the hand bones.

Table 1: Calculation time of hierarchy in seconds.

dataset	dimension	time
tibia-fracture	$190 \times 232 \times 191$	132.6
	$191 \times 213 \times 80$	69.7
	$83 \times 83 \times 152$	16.8
cow-foot	$208 \times 157 \times 204$	106.5
	$159 \times 118 \times 143$	41.9
hand	$92 \times 60 \times 192$	20.1
	$75 \times 57 \times 110$	7.9
	$40 \times 48 \times 30$	1.0

Table 2: Calculation time of the repositioning of the cow foot in seconds.

number of iterations	time
220	101.2
312	127.1
330	151.7
1102	502.4

According to the collision detection, nearly interactive framerates can be achieved depending on the level of detail for the raycasting. When navigating volumes of size 200^3, the average time for interference test was 200 msecs. The memory required to build the hierarchy amounts 45 bytes per node. Therefore, we implemented a memory manager that minimized the memory page swapping.

In Table 2 several timings for the optimization process can be seen. Here, some calculations for the broken cow foot are presented using breadth-first traversal, whereas the number of iterations strongly depends on the quality of the manual adjustment. If the pre-justification is disadvantageous in very few cases the optimization process terminated before finding the optimal solution.

A drawback of the presented optimization process is, that it only calculates the mathematically optimum of the repositioning. We have to keep in mind, that this proceeding can only be an additional help for physicians, since the medical solution mostly differs from the mathematical one. The reason is, when fractures are glued together in reality, the objects are pressed on each other very strongly which insures proper union of the fracture. To allow for this the optimization presented here would have to be adapted using the probability model. Additionally, segmentation errors can influence the results.

8 Conclusion

In this paper, we presented an algorithm for interactive repositioning of bone fractures as an aid for orthopaedists. The datasets were created using standard CT or C-arm devices and were segmented semi-automatically using volume growing approaches. In order to visualize the volumes, the commercial ray-casting library *VGL* was used, which allows rendering of independent volumes. Since collision detection algorithms are necessary for the navigation of the bone fragments, an efficient method is presented using OBBs and an octree hierarchy. The interference test is accelerated by the separating axis theorem and the created hierarchy. In order to semi-automatically reposition the fractures, the volumes are pre-justified by the user. Afterwards, the automatic optimization process is started using Powell's algorithm.

References

[1] M. Brady, K. Jung, Nguyen HT, and T. Nguyen. Two-Phase Perspective Ray Casting for Interactive Volume Navigation. In *Visualization '97*, 1997.

[2] B. Cabral, N. Cam, and J. Foran. Accelerated volume rendering and tomographic reconstruction using texture mapping hardware. In *Proceedings of 1994 Symposium on Volume Visualization*, pages 91–98, 1994.

[3] J. Cohen, M. Lin, D. Manchoma, and M. Pongami. I-collide: An interactive and exact collision detection system for large-scale environments. In *1995 Symposium on Interactive 3D Graphics*, pages 198–196, 1995.

[4] C. Eckstein. Interaktive Relokalisierung von Knochenfraktur-Segmenten. Master's thesis, University of Erlangen-Nuremberg, September 2000.

[5] N. Gagvani and D. Silver. Shape-based volumetric collision detection. In *Proc. of IEEE Visualization*, 2000.

[6] N. Garica-Alonso, N. Serrano, and J. Flaquer. Solving the collision detection problem. *IEEE Computer Graphics and Applications*, 13(3):36–43, May 1994.

[7] S. Gottschalk, M. Lin, and D. Manocha. Obbtree: A hierarchical structure for rapid interference detection. In *Proc. of ACM SIGGRAPH*, 1996.

[8] T. He and A. Kaufman. Collision detection for volumetric objects. In *Proc. of IEEE Visualization*, pages 27–34, 1997.

[9] P. M. Hubbard. Interactive collision detection. In *Proc. of IEEE Symposium on Research Frontiers in Virtual Reality*, 1993.

[10] P. Lacroute and M. Levoy. Fast volume rendering using a shear-warp factorization of the viewing transformation. In *Proceedings of SIGGRAPH '94*, pages 451–458, 1994.

[11] M. Meißner, U. Hoffmann, and W. Straßer. Enabling Classification and Shading for 3D Texture Based Volume Rendering Using OpenGL and Extensions. In *Visualization '99*, 1999.

[12] M. Meißner, U. Kanus, and W. Straßer. VIZARD II: A PCI-Card for Real-Time Volume Rendering. In *Proc. SIGGRAPH/Eurographics Workshop on Graphics Hardware*, 1998.

[13] G. Papaionnou, E. Karabassi, and T. Theoharis. Virtual archaelogist: Assembling the past. *IEEE Computer Graphics and Applications*, pages 53–59, March/April 2001.

[14] H. Pfister, J. Hardenbergh, J. Knittel, H. Lauer, and L.Seiler. The VolumePro Real-time Ray-Casting System. In *Proc. SIGGRAPH*, 1999.

[15] W. H. Press, S. A. Teukolsky, W. T. Vetterling, and B. P. Flannery. *Numerical Recipes in C*. Cambridge University Press, 2nd edition, 1992.

[16] C. Rezk-Salama, K. Engel, M. Bauer, G. Greiner, and T. Ertl. Interactive volume rendering on standard PC graphics hardware using multi-textures and multi-stage rasterization. In *Proc. SIGGRAPH/Eurographics Workshop on Graphics Hardware*, 2000.

[17] H. Weghorst, G. Hooper, and D. Greenberg. Improved computational methods for ray tracing. *ACM Transactions on Graphics*, pages 52–69, 1984.

[18] R. Westermann and T. Ertl. Efficiently using graphics hardware in volume rendering applications. In *Proc. of SIGGRAPH*, Comp. Graph. Conf. Series, 1998.