Cellular Automata for 3D Morphing of Volume Data

SK Semwal
Department of Computer Science
University of Colorado
Colorado Springs, CO. 80933, USA
semwal@cs.uccs.edu

K Chandrashekhar
Department of Computer Science
University of Colorado
Colorado Springs, CO. 80933, USA
kchandra@uccs.edu

ABSTRACT

Morphing involves the smooth transformation of one model, called the source to another, called the target. Several methods have been employed in this field both for two and three dimensional morphing. This paper performs morphing through the usage of cellular automata. The goal was to develop morphing algorithms that would minimize the need for both the user input and correspondence specification between source and the target. Two algorithms, *the bacterial growth model* and the *core increment model* have been designed and implemented in C++. Both algorithms utilize simple automata rules and are stable over dissimilar data. Results are presented that display the efficiency of the approach.

Keywords

Animating Volume Data, Local Interaction creating global phenomena.

1. INTRODUCTION

Morphing is a technique in graphics that results in the transformation of an object, called the source model, into another, called the target model, in a gradual and smooth fashion. Apart from many movies, morphing now finds usage in 3D games that are in the market such as Alter Echo [outrage2003] and Perimeter [K-DLab2004]. The concept of morphing extends to other applications as well. Some example applications are: Visualization during cranio-facial surgery [Fang1996]; evolution by morphing the skulls of primates and modern humans [Rodier1997]; environmental changes on sea levels and forest cover [geoplace2004]; continental drift [Bourke2001] or erosion; and understanding biological processes such as plant growth and fetal development [pbs2004].

Both 2D and 3D morphing methods have been developed. Several good papers can be found on 2D Morphing 1992 [Beier1992,Sederberg1992] and recently in [Abraham2004]. The biggest benefit of 3D morphing over 2D is that it is independent of lighting and other environmental effects which are inherent in the images. In addition, the view of the camera can be changed in real-time in order to provide a much clearer understanding of the morphing process. We focus on the 3D variety.

Many 3D morphing algorithms require a correspondence that maps features of the source model to that of the target model. We wanted to investigate approaches that are free of this restriction. Most morphing algorithms also rely on user-defined control points that guide the way the source model morphs to the target model. While this is useful in guiding morphing in the manner that the user desires, we felt that there is room for exploring techniques using minimal user input because this correspondence process can be tedious and tiring for the user. While this perhaps takes away from the artistic impressions that users are allowed when the correspondence is defined manually, minimal manual specification has its own benefits. Our approach uses the concept of cellular automata in order to perform morphing. Cellular automata are dynamic systems where an N dimensional space is created with each cell containing a value which changes according to pre-determined rules depending on the neighborhood. From this simple local concept, complex global patterns and behavior emerge as the morphing animation considers the collective response of the cells within the lattice. We developed two new

Conference proceedings ISBN 80-903100-7-9
WSCG'2005, January 31-February 4, 2005
Plzen, Czech Republic.

algorithms morphing algorithms using the CA the *core-increment* and the *bacterial-growth* model**.** The rules that have to be written for each cell in the automata to perform this morphing are simple and hence easy to update or replace. There is no correspondence required between models except that identical 3 dimensional volume sizes are required of the source and destination volumes. No control points are needed to drive the morph. Our approaches work really well in situations where there is no pre-defined transformation path required between the source and target.

2. PAST RESEARCH

Cohen et al [Cohen-Or1998] explain the problem of morphing or metamorphosis as follows: 'Given a source model S and a target model T, morphing constructs a series of transformations $\{W_t \mid 1\ t \in [0,1]\}$ such that $W_0 = S$ and $W_1 = T$' [Lerios1995]**.** A sample morph from one of our algotihms is displayed below (Figure 1).

Figure 1: Gradual transition from source to target.

Mesh Morphing and Volume Data Morphing based upon the data which they use. Polygonal morphing is the process of metamorphosis where the source and target are polygonal or polyhedral meshes. A mesh can be defined as a linear surface that consists of a set of polygons that can be described either as a set of vertex/face/edge/graph structures or as a set of vertices [Kanai2000]. One of the important characteristics of a mesh model is that it describes the topology of the model. It is the basis of intense research with the majority of papers in the morphing field concerning this area. Several methods have already developed for entertainment industry. The input mesh-models are easy to create and many packages support model [Maya2004] creation in this format and light effects are computationally faster to process and render as they are well supported. However, complex topologies *are* difficult to morph, especially if they require user control. In addition, many applications such as medical and geological world produce large amount of volume data and may have to be converted into polygonal mesh. Both correspondence and interpolation problems are documented in Kent et al [Kent1992]. Kanai et al use the concept of merging of two meshes in a common domain [Kanai2000] and use *harmonic mapping* method [Kanai2000]. Lee et al [Lee1999] describe a process of correspondence that uses both the source and target meshes at several resolutions and coarse *base domains* or simplified meshes.

3. VOLUMETRIC MORPHING

Volume morphing uses three dimensional volumes as input for the morphing process. Models are described in terms of voxels (short for volume pixels, the smallest box shaped unit of volume). Chen et al [Chen1995] define a volume as a collection of scattered voxels, with each voxel being associated with a set of values of size S, i.e, the volume V is given by:

$$V = \{(x_i, v_i) \mid x_i \in R^3, v_i \in R^S, i = 1 .. n\}$$

The most popular format for representing volume data is in the form of a 3 dimensional grid. Each (i,j,k) position represents a voxel which has a value associated with it.

The volumetric approach is not as popular as polygonal formats in the entertainment industry. It is computationally intensive to process and render. However, volumetric approach is free of restrictions of topology and geometries. Volume morphing can easily be applied to meshes by converting them to volumetric data while the reverse can result in topologies which are difficult to morph. A large amount of data in the medical, geological and energy fields is generated in the volumetric format and needs to be morphed directly. Most volumetric algorithm do not need a bijective mapping between vertices of the source and target formats like mesh morphing techniques. The simple format allows implementation ease.

4. CELLULAR AUTOMATA

Cellular Automata (CA) were originally proposed as formal models of self-reproducing organisms [Sarkar2000]. CAs are dynamical systems that occupy a uniform, regular lattice, work in discrete steps of time and are characterized by local interactions [Wolfram1984]. They utilize a discrete lattice of sites Discrete time steps drive the simulation. Each site can only take a finite set of values. Each site evolves according to the same deterministic rules. The evolution of a site only depends on the neighborhood. The main advantage of CAs is that complex patterns and behaviors can be achieved using simple local interactions. CAs have been used in many applications [Sarkar2000,] , Bezzi2000, Sloot2002,Hamagami2003, Forsyth2002, Droun2003]. Sosič and Johnson [Sosič1995] use the

concept of CA to describe a growing automaton. Sloot et al [Sloot2002] describe a non-uniform model used to simulate drug treatment for HIV infection. Bezzi [Bezzi2000] describes the simulation of several biological processes using CAs. Claudia et al [Claudia2001] discuss the use of the CAs for simulating the effects of a landslide. [Droun2003] uses a cellular automaton to deforming 3 dimensional objects, not 3D morphing.

Most 3D morphing techniques utilize the idea of correspondence, which is mapping where each point in the source model will end up in the target model [Kanai2000, Lee1999, Kent1992, Chen1995, Cohen-Or1998, Lerios1995]. This becomes inconvenient if there are complex topologies that would require many control-points to describe the morphing accurately.

We looked at volumetric morphing as a collection of voxels comprising the source model trying to achieve similarity with the voxels in the target model. The cellular automata which was used in our design has the following characteristics: It is a 3 dimensional lattice. The dimension of the lattice is that of the volume. Each cell can either be empty or contain one value. All cells are equal, in the sense that a change of value within a position does not change the behavior of the entities occupying the automata. The cellular automaton is non-circular.

Using cellular automata as a base, we have developed 2 algorithms that perform morphing. In both cases, the volume is treated as a cellular automaton. Each non-empty voxel is treated as an independent agent.

5. CORE-INCREMENT ALGORITHM

The core-increment algorithm works using the intersecting parts of the morphs as a base. The intersection part of the source and the target is used to create a core. The core is then grown or incremented in a step-wise fashion so that it fills the space of both the source and target models. More specifically, for each voxel position present in the core, the source and target models are checked to see if any voxels within them surround the position. If so, the voxels are added to the core at the same position that they were found in the source or target. At each step, the voxel positions that are needed to occupy the space of the source and target are recorded in the delete-array and add-array respectively. The arrays contain the points added to them as separate sets during each iteration. The core-increment process completes when there are no more positions either in the source or target that the core can grow into.

Next, the source model is loaded into a new volume. The add and delete-arrays are scanned, one forward and the other in reverse. At each step, the voxel positions mentioned in the delete-array for that iteration is removed from the source model and the corresponding voxels are added from the add-array. In this way, gradually, voxels are removed from the source model where they do not intersect with the target model and voxels added where the target model is supposed to be. The forward and reverse iterations give the morphing a smooth, organic quality. The pseudo-code is as follows:

```
proc core-increment
  // Loading of volumes and tests
  Load source volume as srcVol;
  Load target volume as tgtVol;
  If  ( dimension(srcVol) != dimension(tgtVol)  )
          print error  and exit endif
  Create core with dimensions of srcVol
  // Initialization of core
 for each voxel position (i,j,k)
     if  ( both tgtVol and srcVol's  has object present)
          add (i,j,k) to the core at (i,j,k)
     endii
 end for
  // Iteration to create add and delete arrays
 do
    for each non-empty voxel position (i,j,k) in core
        if voxel found surrounding (i,j,k) in srcVol
                    add to core at (i,j,k)
                    add (i,j,k) to del-array
        end if
        if voxel found surrounding (i,j,k) in dstVol
                    add to core at (i,j,k)
                    add (i,j,k) to add-array
        end if
    end for
until core cannot increment further
Load source volume as morphVol
// Morph iterations.
 for i = 0 to sizeof(add-array)
        // iterating through the add-array
```

```
        get position at add-array[i] as (i,j,k)
        add voxel at (i,j,k) to morphVol
        // iterating through del-array in reverse
        get position at del-array[size(add-array)-i] as
            (i,j,k)
        add voxel at (i,j,k) to morphVol
        // Rendering the deformed volume
        Render morphVol
    endfor
endProc
```

The rules that define the behavior of the cellular automata that makes up the core are simple, hence easy to upgrade or replace. The algorithm uses 3 dimensional arrays containing the position data from volumetric models. This means that most popular formats of representation of volumetric data can be used directly. No complex data-types or intensive pre-processing is required. There is no correspondence required between the source and the target models. In the above example, it is clear that there is no correspondence information present.

The morph can be controlled because of the add and delete arrays containing the information of each iteration as separate sets. In cases where many points are added in the add array as compared to the del array or vice versa, by controlling the sets released per iteration from the arrays, the morph can be made to be a gradual process. This is important in cases where the source model is very small in comparison to the target model or vice versa. In the normal case, if the source were small, the non-intersecting parts of the source would either disappear quickly while the target would grow slowly, or if the target were small, the target would grow to completion while the source was still disintegrating. By coordinating the release of points this problem can be avoided.

By using random probabilities in the points being selected for each iteration, the morphing gains an organic quality (Figures 4 and 5). The growth of the morph can be made to start with an uneven texture to the surface that clears up during the end of the morph to give the texture of the target model. The method requires the creation of four volumes, two for the source and target models to be loaded, one for the core and one for the source model during the morph iterations. Since the implementation of the method results in the first instances of the source and target models being destroyed, the source cannot be reused during the morph iteration. This makes the implementation memory-heavy if very large models are used. The algorithm requires the volumes containing the source and target models to be of the same dimensions. However, there is no restriction on the size of the models themselves.

6. BACTERIAL GROWTH MODEL

Several papers during my research into cellular automata have mentioned its use for simulating the behavior of bacteria given certain environmental conditions. Each bacterium is modeled as an entity within a lattice and rules govern its reaction to the environment and other bacteria. Researchers have succeeded in simulating complex behavior for bacteria using the simple rules required for CAs. This gave rise to the idea of using the bacterial growth model as a method of morphing (Figures 1 and 2).

The following rules govern the behavior of bacteria:

(a) Bacteria are non-motilee. (b)All bacteria are of the same type and governed by the same rules. (c) A bacterium has 2 needs, the need for food and the need to reproduce, the latter being dependent on the former. (d) A bacterium will reproduce if it finds food and has space to place its offspring by making a copy of itself. (e) A bacterium creates only one offspring per iteration. (f) A bacterium with food at its current location will live and reproduce infinitely given enough space. (g) A bacterium will die if food is not present in its current position. (h) Bacteria cooperate to keep each other alive. If a bacterium is completely surrounded by other bacteria, it does not die even if its current position contains no food. (i) Each non-empty voxel within the target volume is considered as a source of food. Each source contains an infinite supply of food.

Each ‘bacterium’ within the source volume checks to see if it has food in its current position. If not, and if it is not completely surrounded in 26 directions by other bacteria, it dies with a certain probability. If it finds food, it looks for a empty place in the neighborhood to reproduce and place its progeny, with a certain probability. In this way, bacteria in parts of the source volume that do not intersect with the target volume begin to die out, thus removing the feature. Bacteria that intersect the target volume begin to breed, placing their progeny in places where the target volume is supposed to be. This results in features of the target growing to form the final shape of the target volume. The pseudo-code is as follows:

```
Proc core-increment
    // Loading of volumes and tests
    Load source volume as srcVol;
    Load target volume as tgtVol;
    if ( dimension(srcVol) != dimension(tgtVol) )
```

```
        print error; exit;
    endif
  for each non-empty voxel position (i,j,k) in srcVol
   do
     if  (voxel at position (i,j,k) is non-empty )
        // food at current position
          if  ( voxels surrounding (i,j,k) have a
          non-empty position (i1,j1,k1) )
              reproduce by placing copy of voxel at
              (i,j,k) in (i1,j1,k1) with   probability p1.
           else if  (not completely surrounded by
                voxels at position (i,j,k) )
                 die by removing voxel at (i,j,k)
                  with probability p2
                endif
          endif
       render srcVol;
     enddo
  end core-increment
```

7. RESULTS AND ANALYSIS OF THE PROPOSED APPROACHES

The above two algorithms that were developed were implemented in C++ using the Visualization Tool Kit [vtk2004] library as the rendering engine. The Visualization ToolKitVTK [VTK2004] was free and provided open-source C++ library that supports several graphics related activities including image processing and 3D visualization. It has inherent support for volume data and runs on all popular machine-platforms. The tests of algorithms were done with the following datasets with certain characteristics on a PC with dual Pentium III processors running at 1 GHz with 1.5 Gb memory. In both Table 1 and 2, these cases are identifies as (a)-(d) as follows: Case (a) is the morph sequence where source is *Input.bin* and target is Fuel.bin. Both these data sets are are 64 x 64 x 64 datasets with about 17,000 non-empty voxels (Figure 1). The source model intersects to a large extent with the target. Case (b) is the morph between *Cube256x256x256.bin* to *aneurism.bin*. These are 256 x 256 x 256 sized datasets. There are about 1.1 million voxels in total. The target (*aneurism.bin*) model is dissipated throughout the volume, being branch-like. There is no central core volume as in other models, hence there is very small amount of overlap between the source (cube) and target (aneurism) model. This leads to a large amount of voxel additions and deletions. Case (c) morphs *Cube256x256x256.bin* to *MRI-head.bin*. Once again the data sets are of size 256 x 256 x 256. They have around 7.1 million non-empty voxels between them. The source is a cube that is centered across the volume. The target volume is a MRI of a head that envelops the cube; hence most of the voxel manipulations are addition operations. Finally, Case (d) is the morph between *MRI-head.bin to bonsai.bin***.** These are again 256 x 256 x 256 sized datasets. The total of the non-empty voxels of both source and target is 7.3 million. The source model overlaps the target to a large extent and hence, most of the operations in this morph involve the deletion of voxels. Figures 2-5 show the results of our two algorithms.

The best and worst case complexity of this algorithm is n3 where n is the size of one dimension of the source or target volume. As shown in Tables 1 and 2, normally we find that the amount of time taken for each iteration as well as the number of iterations depend on the size of the volume and the number of non-empty voxels within it..

In case of test Case (b), the small amount of overlap leads to a large amount of additions and deletions. In this case, the number of iterations became large for the bacterial growth algorithm, with the iterations during the end of the morph yielding very small numbers of voxels. These do not contribute significantly to the quality of the morph. The performance of this algorithm is better than the core increment method described earlier. This should not be assumed to be a reflection of the efficiency of the algorithm. The main reason for this is that the bacterial growth algorithm incorporates an iso-surfacing algorithm. This means that only the voxels on the surface of the intermediate volumes are processed. The core increment algorithm does not easily support such a scheme and hence the current implementation processes all the voxels present in its core. Bacteria growth algorithm seems to do well in cases where there is a large percentage of overlap between volumes. The quality of the morphs is in general worse than the core-increment algorithm and this can be assessed by looking at Figures 2-5.

Table 1 and 2 also show the time taken to create morphing sequences, and the average time taken to complete a sequence during morph.

Our implantation results indicated that the core increment algorithm is more stable and provide better visual results with a varied type of source and target models than the bacterial growth models. In comparison to other existing 3D algorithms, the

solution provides a morph which does not require any human-intervention, and the morphing sequences has better visual appearance as in both cases morphing sequences are expected to grow gradually in spatial domain, avoiding frequency interpolation based aliasing completely.

8. CONCLUSIONS AND FUTURE RESEARCH

We have presented two algorithms which successfully demonstrate the 3D Morphing. The methods presented in this paper can handle branching structures and topological mismatches, which have been a problem for the past algorithms, without any human-intervention. Our current design requires that the volumes intersect. An important improvement would be to handle is that non-intersecting volumes. Our method can be extended by merging the current design with the distance field metamorphosis technique [Cohen-Or1998] when the volumes do not intersect. Parallelization has been performed on cellular automata based models before [Telford1999] and our method can benefit from that as well. We will like to also consider non-homogenized mixture of bone, tissue etc) in future as well. In addition, we also plan to develop methods to handle color (rgb) volume data sets.

9. REFERENCES

[Abraham2004] Abraham AW. Image View-Shift: Three dimensional representation from a photo. MS Thesis, University of Colorado, Colorado Springs, pp. 1-274, 2004.

[Beier1992] Thaddeus Beier and Shawn Neely. Feature-based image metamorphosis, International Conference on Computer Graphics and Interactive Techniques, pp. 35 – 42,1992

[Bezzi2000] Bezzi M. Modeling Evolution and Immune System by Cellular Automata. http://citeseer.nj.nec.com/429312.html

[Breen2001] D. E. Breen and R. T. Whitaker. A level-set approach for the metamorphosis of solid models. IEEE Transactions on Visualization and Computer Graphics Volume 7, Number 2, pp. 173, 2001

[Bourke2001] Paul Bourke. Terrain morphing. http://astronomy.swin.edu.au/~pbourke/terrain/tmorp

[Calionna2001] Claudia R Calionna ,Claudia Di Napoli, Maurizio Giordano,Mario Mango Furnari and Salvatore Di Gregorio. A network of cellular automata for a landslide simulation. International Conference on Supercomputing Proceedings of the 15th international conference on Supercomputing, pp. 419 – 426, 2001

[Chitttarao2001] Chitttaro L. Information Visualization and its Application to Medicine. Artificial Intelligence in Medicine, vol. 22, no. 2, pp. 81-88, 2001.

[Chen1995] M. Chen and M. Jones and P. Townsend. Methods for Volume Metamorphosis. In Image Processing for Broadcast and Video Production, Y. Paker and S.Wilbur (Eds.), Springer-Verlag, London, 1995.

[Cohen-Or1998] Daniel Cohen-Or, Amira Solomovic, David Levin. Three-Dimensional Distance Field Metamorphosis. ACM Transactions on Graphics (TOG), Volume 17, 2, pp. 116 – 141, 1998.

[Droun2003] Druon S, Crosnier A, Brigandat L. Efficient Cellular Automata for 2D / 3D Free-Form Modeling. Journal of WSCG (Winter School of Computer Graphics), 11, 1, pp. 102-108 : 2003

[Fang96] S. Fang and R. Raghavan and J. Richtsmeier. Volume Morphing Methods for Landmark Based 3D Image Deformation. SPIE International Symposium on Medical Imaging, 1996

[Forsyth2002] Cellular Automata for Physical Modeling. Game Programming Gems 3, 2002.

[Gagvani2001] Nikhil Gagvani and Deborah Silver. Animating volumetric models. Volume modeling Volume 63, Issue 6, November 2001, pp. 443–458, 2001.

[Hamagami2003] Tomoki Hamagami and Hironori Hirata. Method of crowd simulation by using multiagent on cellular automata. EEE/WIC International Conference on Intelligent Agent Technology, pp. 46 – 53, 2003

[He1994] T. He and S. Wang and A. Kaufman "Wavelet-Based Volume Morphing" Proceedings of Visualization 94, Pages: 85-92,: 1994

[Java3D2003] Java 3D API ® Sun Microsystems http://java.sun.com/products/java-media/3D/

[Kanai2000] Takashi Kanai Hiromasa Suzuki Fumihiko Kimura. Metamorphosis of Arbitrary Triangular Meshes with User-Specified Correspondence. IEEE Computer Graphics & Applications, 20, 2, pp. 62-75, 2000.

[Kent1992] James R Kent ,Wayne E Carlson and Richard E Parent. Shape transformation for polyhedral objects. International Conference on Computer Graphics and Interactive Techniques, pp. 47 – 54, 1992

[Kazakov2003] Maxim Kazakov,Alexander Pasko and Valery Adzhiev. Interactive metamorphosis and carving in a multi-volume scene. Proceedings of the 1st international conference on Computer graphics and interactive techniques in Austalasia and South East Asia, pp. 103, 2003.

[K-DLab2004] Perimeterv Developer: KD-LAB / 1C Company www.codemasters.com/perimeter

[Lerios1995] Apostolos Lerios, Chase D. Garfinkle, Marc Levoy. Feature-Based Volume Metamorphosis. International Conference on Computer Graphics and Interactive Techniques Proceedings of the 22nd annual conference on Computer graphics and interactive techniques, pp. 449 – 456, 1995

[Lee1999] Aaron Lee and David Dobkin and Wim Sweldens and Peter Schroder. Multiresolution Mesh Morphing. Siggraph 1999, Computer Graphics, pp. 343-350, 1999.

[Lee1988] F. Sweldns, W. Schorder, P. Cowsar, and L., Dobkin. Multiresolution Adaptive Parameterization of Surfaces. Computer Graphics, pp. 95–104, 1998.

[Maya2004] Aliaswavefront Maya ® http://www.alias.com/eng/products-services/maya/index.shtml

[Outrage2003] Alter Echo Developer: Outrage Games Publisher :THQ www.outrage.com

[Sarkar2000] Palash Sarkar. A brief history of cellular automata. ACM Computing Surveys (CSUR) Volume 32 , 1, pp. 80 – 107, 2000.

[Sederberg1992] Thomas W Sederberg and Eugene Greenwood. A physically based approach to 2–D shape blending. International Conference on Computer Graphics and Interactive Techniques pp. 25 – 34, 1992

[Sloot2002] Peter Sloot, Fan Chen, and Charles Boucher. Cellular Automata Model of Drug Therapy for HIV Infection. Lecture Notes In Computer Science Proceedings of the 5th International Conference on Cellular Automata for Research and Industry, pp. 282 – 293, 2002.

[Sosič1995] R. Sosic and Robert R. Johnson. Computational properties of self-reproducing growing automata. BioSystems, Volume: 36, pp. 7-17, 1995.

[Sussman2004] Alan Sussman, Michael Beynon,Mary Wheeler, Steven Bryant, Malgorzata Peszynska, Ryan Martino,Joel Saltz, Umit Catalyurek, Tahsin Kurc,Don Stredney, Dennis Sessanna. Exploration and Visualization of Oil Reservoir Simulation Data, 2004.

[Treece2001] Graham Treece and Richard Prager and Andrew Gev. Volume-based three-dimensional metamorphosis using sphere-guided region correspondence. The Visual Computer,volume 17, 7, pp. 397-414, 2001.

[Ulgen1997] F Ulgen. A step towards universal facial animation via volume morphing. Robot and Human Communication, 1997. RO-MAN '97 6th IEEE International Workshop, pp. 358 – 363, 1997

[Wolfram201984] Wolfram ,Stephen. Universality and Complexity in Cellular Automata. Physica D 10, pp. 1-35, 1984.

[vtk2004] The Visualization ToolKit (VTK) http://www.vtk.org/

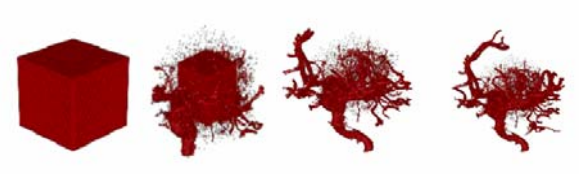

Figure 2: using bacterial growth model (Cube to Aneurism)

Figure 3: using bacterial growth model (Head to Bonsai)

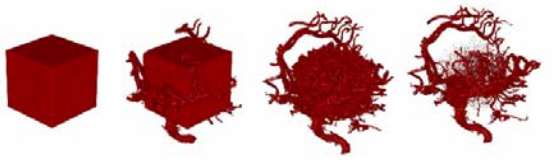

Figure 4: Using core increment model (Cube to Anuerism)

Figure 5: Using core increment model (Head to Bonsai)

	Volume size	**Source non-empty Voxel Count**	**Target non-empty Voxel Count**	**No of iterations**	**Avg Morphing Time per iteration (secs)**	**Total Time taken for morphing (secs)**	**Avg Rendering Time (secs)**
a	64 X 64 X 64	4096	13731	32	0.02	0.64	0.76
b	256X256X256	1000000	168948	387	1.16	448.92	3.16
c	256X256X256	1000000	6198649	132	3.38	446.16	2.14
d	256X256X256	6176412	1298598	195	1.33	259.35	2.71

Table 1: Core element results

	Volume size	**Source non-empty Voxel Count**	**Target non-empty Voxel Count**	**No of iterations**	**Avg Morphing Time per iteration (secs)**	**Total Time taken for morphing (secs)**	**Avg Rendering Time (secs)**
a.	64X 64 X 64	4096	13731	32	.0334	1.07	1.27
b.	256X256X256	1000000	168948	248	2.5027	620.67	4.46
c.	256X256X256	1000000	6198649	120	8.7642	1051.71	3.32
d.	256X256X256	6176412	1298598	175	11.24	1966.24	3.33

Table 2: Core element results