Shape Transformation of Multiple Objects Using Slices

Shamima Yasmin
School of Computer Sciences
Universiti Sains Malaysia
11800 USM Penang, Malaysia
shamima@cs.usm.my

Abdullah Zawawi Talib
School of Computer Sciences
Universiti Sains Malaysia
11800 USM Penang, Malaysia
azht@cs.usm.my

ABSTRACT

3D shape transformation is usually confined to transformation between a pair of objects. The objective of this paper is to look at shape transformation from a different perspective: instead of binding this concept between two objects, the technique is extended to the concept of incorporating the characteristics of a number of objects in one body at a time. Equal number of slices are generated from all objects. Slices may be parallel to each other or each slice may have different orientation. Traversal of a data along its longitudinal direction may generate slices which are differently oriented from each other. When multiple objects are transformed to one and is used as an influence shape, it also works as incorporating multiple influence shapes at a time during transformation between two objects. The paper shows the ease of implementation of this concept in sliced data and also discusses its extendibility.

Keywords:
Shape Transformation, Boundary interpolation, Surface Reconstruction.

1. INTRODUCTION

Generally shape transformation means incorporating the characteristics of two different bodies in one output. Our aim is to interpret the term in a different way: incorporating the overall characteristics of a number of different objects (more than two) in one output or incorporating the characteristics of a number of different objects in different proportions in one output. The idea of inventing and designing new models may have applications in show piece design, in pottery industry as well as in animation industry.

In this paper a shape transformation algorithm based on slices is discussed. A number of objects are collected and equal number of slices are generated from these objects. Slices from a particular object may be parallel to each other or each slice can have different orientations. The method works with minimal user intervention without producing any significant distortion to the eye. Again a number of different transformed output can be generated from the same number of objects and user has the right to choose the best one. This method particularly shows the extendibility, versatility of the sliced data and ease of its implementation.

2. BACKGROUND

Shape transformation algorithms can be classified into the following two broad categories: a) Surface-based: consists of continuous mapping of small pieces of polygonal surfaces of corresponding objects; b) Volume-based: modifies voxel values of a volume data set.

Surface-based approach uses user-defined control fields such as point fields, line fields etc. during transforma - tion to map key features of the input objects [Hong88, Kent92, Gregory98, Lazarous94, Lee99]. Surface-based methods are important because of its ability to morph between objects of different types of genus, but these methods also require a significant amount of user input. Another troubling feature of surface-based method is the problem of self-intersection. It cannot guarantee that polygonal surfaces will not pass through themselves, creating self-intersecting intermediate result as found in

[Hong88]. All these problems will become more prominent when more than two objects are involved.
Volume-based approach alleviates some of the problems mentioned above. Among them, the simplest approach is the cross-dissolving method [Hughes92,He -94] which at first transforms volume data from spatial domain to frequency domain, interpolates volume in frequency domain and again transforms back to spatial domain. Transformation of multiple objects based on fourier or wavelet can be easy but both methods will have difficulties in specifying slightly complex geometric transformations such as object rotation.
This problem can be alleviated by applying warping before interpolation as found in [Lerois95]. Here user-defined warp is applied on input objects to resemble each other. Instead of using point and line control fields, user-specified disk field [Chen96] can be used. Equal number of disks are applied on both source and target to establish correspondence between them. Each disk has its own normal direction which helps considering distortion of the body. But as the number of input objects increases, the amount of user intervention involved also increases in both of the above mentioned cases.
'Distance Volume' [Payne92] measured by computing the shortest distance of each voxel within the volume to the surface of the object or 'Level Set Method' [Breen01] where the way in which points on the surface moves is used to establish connection among all input objects may not scale well as the number of objects increases.
In shape transformation using implicit function [Turk99], implicit functions of each pair of 2-D slices are determined using a set of constraints i.e. location, weight, scalar values etc. When the number of objects increases, this method may become a bit complex. Pasko et al. uses CSG for blending a number of objects but in a very limited way [Pasko05].
From the above discussion, it is obvious that most of the shape transformation methods do not scale well as the number of objects increases. Our aim is to develop a shape transformation algorithm which optimizes user input, considers rotation/orientation of rigid body during transformation as well as scales better when the number of input objects increases.

3. PROPOSED ALGORITHM

The algorithm mainly consists of the following major steps as shown in Figure 1:

- Data Traversal and the Slicing of the Data;
- Boundary Extraction;
- Boundary Projection and Boundary Interpolation;
- Orientation and Translation of Boundaries;
- Surface Reconstruction.

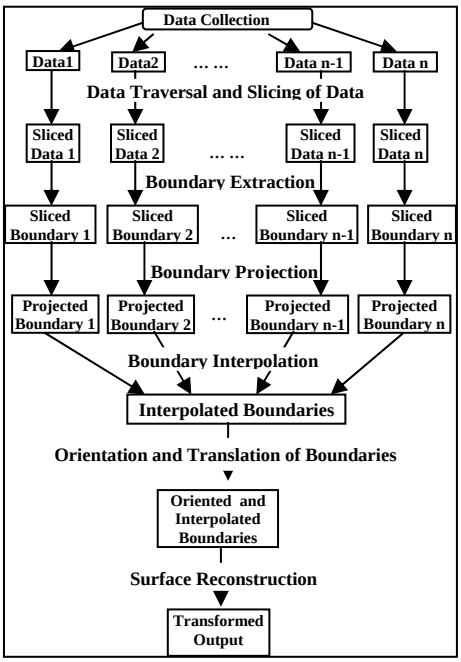

Figure 1: Flow Chart of the Proposed Algorithm.

Various steps of the algorithm are discussed in detail in the following sub sections.

3.1 Data Traversal and Slicing of the Data

A number of data is collected. The initial orientations along which the data are subdivided in the first step of the binary subdivision is defined along any of the directions of the Oriented Bounding Box (OBB) [Lin96]. An Oriented Bounding Box (OBB) is a bounding box that does not necessarily align itself along the coordinate axes. OBB is constructed from the mean and covariance matrix of the cells and their vertices that define the dataset. The eigen vectors of the covariance matrix are extracted, giving a set of three orthogonal vectors that define the alignment of the dataset. Figure 2 shows the difference between a normal bounding box and an oriented bounding box. No doubt, an oriented bounding box more closely fits the data than a normal bounding box. The purpose of choosing the oriented bounding box is to allow checking of the longitudinal direction of dataset from its oriented bounding box rather than from normal bounding box.
Eigen vectors describe the maximum, medium and mini –mum variance of concentration of point clouds. The

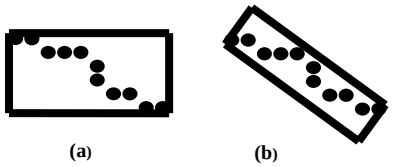

Figure 2: (a) Normal Bounding Box and (b) Oriented Bounding Box.

'maximum' direction shows the maximum amount of concentration of the cells of the data along that direction, whereas the 'medium' direction exhibits less amount of concentration than maximum direction and the 'minimum' direction shows the least amount of concentration or the least alignment of the cells along that direction. The first subdivision takes place along a plane centered at the center of the Oriented Bounding Box (OBB) of the object with normal along the initial alignment. This step, called 'step 0', divides the data into two end parts.

In the next step i.e. 'step 1', each of the two end pieces found from 'step 0' is wrapped with OBB and tested whether the longitudinal direction of the alignment of the sliced end is still within the maximum or medium direction of the OBB. If the alignment is still within maximum/ medium direction, a line joining the center of the previous cut plane and the center of the OBB is used as the direction of the cut plane normal for the ends in that step (Figure 3(a)). Otherwise ends are sliced along the cut plane normal found in the previous step which is used for any further subdivision of the ends and in the subsequent steps no further checking on the alignment is done. At the end of 'step 1', the data is divided into four parts i.e. two end parts and two middle parts.

Before further subdivision of the ends, if necessary, checking is done for the alignments of the two sliced ends. The procedure described in 'step 1' is followed for further subdivision of the two ends. For the middle parts, data is sliced along the plane with center as the center of the OBB and normal directed along the resultant normals of the two ends of the middle data (Figure 3(b)). Slicing is continued along the longitudinal direction until the desired number of steps is reached. In each subsequent step, the number of slices is doubled. The default longitudinal direction is the 'maximum' direction of the eigen vectors and the default number of steps for binary subdivision of the data is 'four'. To provide more flexibility, the initial longitudinal direction as well as the number of steps can be defined by the user. We indicate the maximum alignment as '0', the medium alignment as '1' and the minimum as '2'. Therefore, the users are allowed to vary the morphed output based on the initial alignment. Subdivision can also be forced to happen along any particular direction or along any of the axes i.e. x, y or z to generate parallel slices.

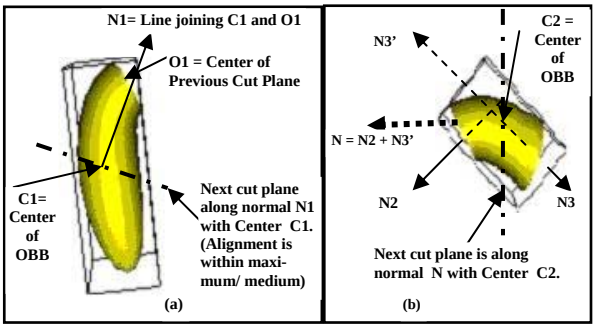

Figure 3: Division of (a) the End Data and (b) the Middle Data.

After reaching the desired number of steps, two ends are traversed along the tip. Usually no further checking for the alignments of the two ends are needed now as the current alignments of the sliced ends are usually not along the maximum or the medium direction of the OBB of the two sliced ends. Hence the normal is usually along the direction which was found at the step before the last checking step and traversing towards the tip is continued along that direction until it is close enough to the tip. At this stage two ends are again divi -ded into two parts.

3.2 Boundary Extraction

Only boundaries of the slices are extracted (Figure 4). As discussed above, in 'step 0', data is divided into two parts. In each of the subsequent steps, the number of slices are doubled. Hence in 'step 4', there are $2^{(4+1)}$ i.e. 32 slices. From 32 slices, 31 boundaries can be extracted. Then boundaries at the ends which are determined after the specified number of steps is reached are also added. Two such boundaries at the two ends result in a total of 33 boundaries each. Hence in 'step 4', the number of extracted boundaries (for each data) is computed as follows:

$$2^{(step+1)} + 1 = 2^{(4+1)} + 1 = 2^5 + 1 = 33.$$

3.3 Boundary Projection and Boundary Interpolation

All data boundaries are projected onto the XZ plane and centered at the origin. Each of the boundaries are traversed along the direction of their minimum X (X_{min}) to maximum X (X_{max}) with a traversal plane defined as (1,0,0). For each boundary, traversal spacing is deter-

 ISBN 978-80-86943-94-7

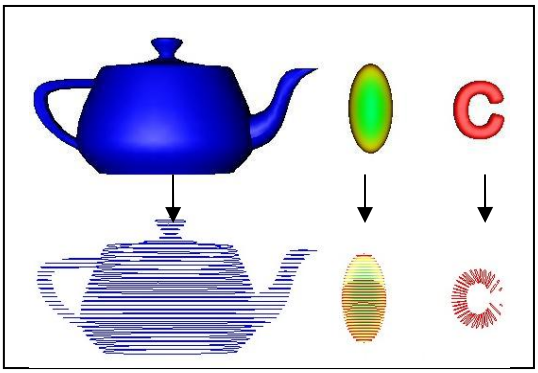

Figure 4: Extraction of Boundaries from a Number of Data.

mined separately. Equal number of traversals is perfor - med for all data. Traversal spacing is determined as follows:

$$Spacing = (X_{max} - X_{min}) / Number\ of\ Traversals$$

Only boundary points are extracted from the traversals. If the number of extracted points in any cut plane happens to be odd, it is made to be even. Next interpo - lation is performed onto the XZ plane. For simplicity, linear interpolation is used in our implementation. Here it should be noted that only one normal is extracted per boundary regardless of whether any particular boundary consists of multiple holes or empty spaces. Also each boundary has one center irrespective of the irregular geometric configuration of that particular boundary. Usually corresponding boundary point clouds are just interpolated. Sometimes enhancement of the interpola - tion process is carried out. When all boundaries have equal number of regions and there are more than one region in all of them, then each of the corresponding regions are interpolated so that interpolated point clouds also have equal number of regions (Figure 5). Region is an area where the number of points extracted by the cut plane is the same while traversing along the X axis.

Again when there are equal number of empty spaces in all boundaries, we have equal number of corresponding regions for the corresponding boundaries. Thus corres - ponding regions can be interpolated. When some of the boundaries contain empty space while other does not have any, number of empty spaces is counted from the boundary which contains the least number of empty spaces and equal number of empty spaces are inserted into the non-empty boundaries. Now all the boundaries contain empty spaces. When there are unequal number of empty spaces/ regions, rightward and leftward traver -sals are carried out from both ends until region in one of the boundaries is exhausted. Corresponding regions

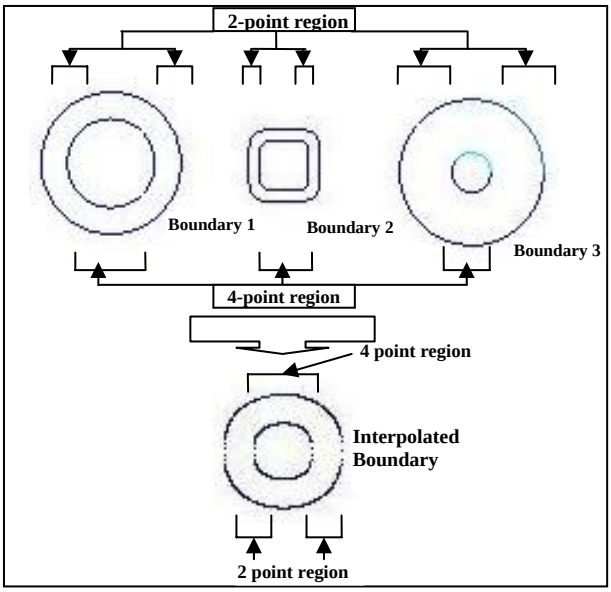

Figure 5: Interpolation of Boundaries after Region Separation.

during the traversal are just mapped and interpolated while the remaining regions can just be mapped if the exhausted side ends with an empty space. Otherwise number of empty spaces is counted from the non-exhausted side which contains the least number of empty spaces and a similar process is applied by inserting into the region of the exhausted side equal number of empty spaces.

3.4 Orientation and Translation of Boundaries

Each of the interpolated boundary already projected onto the XZ plane is oriented along the resultant normal of each of the source and target boundaries and trans lated- to the average center of each of the source and target boundaries (Figure 6). When all the interpolated boundaries are oriented as well as translated, we get the outline of the morphed output (Figure 7).

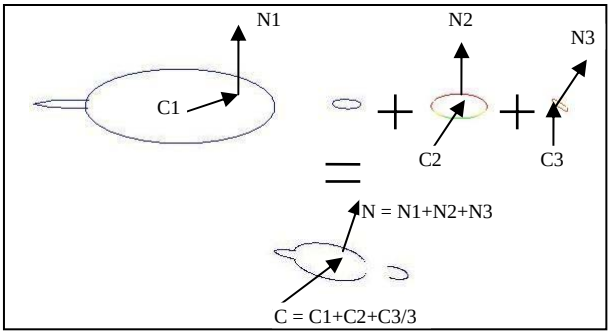

Figure 6: Orientation and Translation of a Single Inter - polated Boundary.

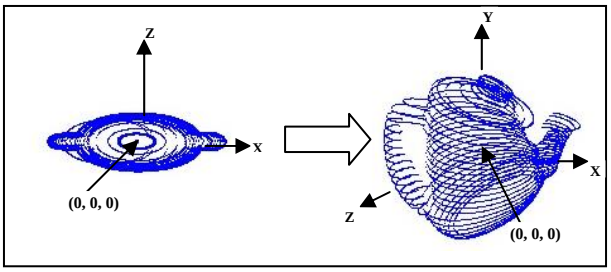

Figure 7: Orientation and Translation of All Interpolated Boundaries.

3.5 Surface Reconstruction

From the stack of oriented and translated boundaries, surface of the morphed object is constructed. Each of the boundaries merges with the next boundary by dividing the in-between space of the two consecutive boundaries into a number of cells. Surface reconstruction is performed by only considering each of the two consecutive boundaries. This simplifies the overall surface construction process as where data is highly irregular, necessary modification among cell coordinates is limited to only two consecutive boundaries. Surface reconstruction in detail is discussed next.

3.5.1 Disconnected Region Separation

Each consecutive boundary may have regions which are disconnected from one another (Figure 8(a)). Nearest neighbor searching is carried out to find this kind of regions. The disconnected regions are laterally mapped (for better effect) and the other regions are to be mapped across the boundary. This is called vertical mapping (Figure 8(b)). The details of the vertical mapping are discussed in the next subsection.

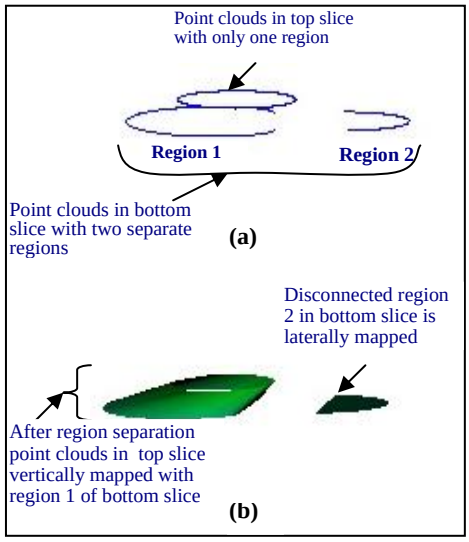

Figure 8: Separating Disconnected Regions between Two Consecutive Boundaries.

3.5.2 Basic Cell Construction

After region separation, two consecutive point/ cell arrays (representing two consecutive slices) are obtained and vertically mapped. The two arrays which contain the number of interpolated points at each index need to be compressed so that the process of mapping can be carried out in an easier and straightforward manner. Figure 9 shows the process of compressing two consecutive arrays. In the compression, the arrays are transformed into two new arrays each: region number array (where region numbers are stored) and number of occurrences array (where the number of occurrences of each region number are stored).

Firstly, the size of both arrays should be made equal using a heuristic approach. Sometimes some index values are dissolved and some are omitted in order to make the size of both arrays equal. Corresponding values of the number of occurrences arrays should also be made equal so that they are ready to be vertically mapped. In the case of unequal values, the larger of the two values is made equal to the smaller number by removing excess number of that particular number of occurrences value. Corresponding number in the two region number arrays should also be equal for the purpose of vertical mapping. If they are not equal, a further processing needs to be done. The process starts with finding the nearest matched index values of the region number arrays by traversing to the left and the right. The nearest matched values will ensure better continuity between different-numbered regions. Next the corresponding region numbers are split into two portions where the values of the region number of the first portion is derived from the continuous mapping of the nearest matched index values to the corresponding region number values and the values of the region number of the second portion are the remaining region numbers resulting from the split. In the example (Figure 9), the first discrepancy occurs at index number ‘2’ and the nearest matched values are at index number ‘1’ with a value of ‘4’ and ‘4’. The current values (i.e. 8 and 6) need to be split into two portions. The first portions are made equal to ‘4’ and the second portions are assigned the remaining values (8-4 = 4 and 6-4 = 2).

At the end of the entire processing, two sets of region number arrays are obtained. The top set (Figure 9(a)) now consists of equal region number and can therefore be vertically mapped whereas each of the bottom set (Figure 9(b)) is to be laterally mapped separately.

Enhancement is carried out in surface reconstruction when empty space is met or at the transition point between two different-numbered regions.

 ISBN 978-80-86943-94-7

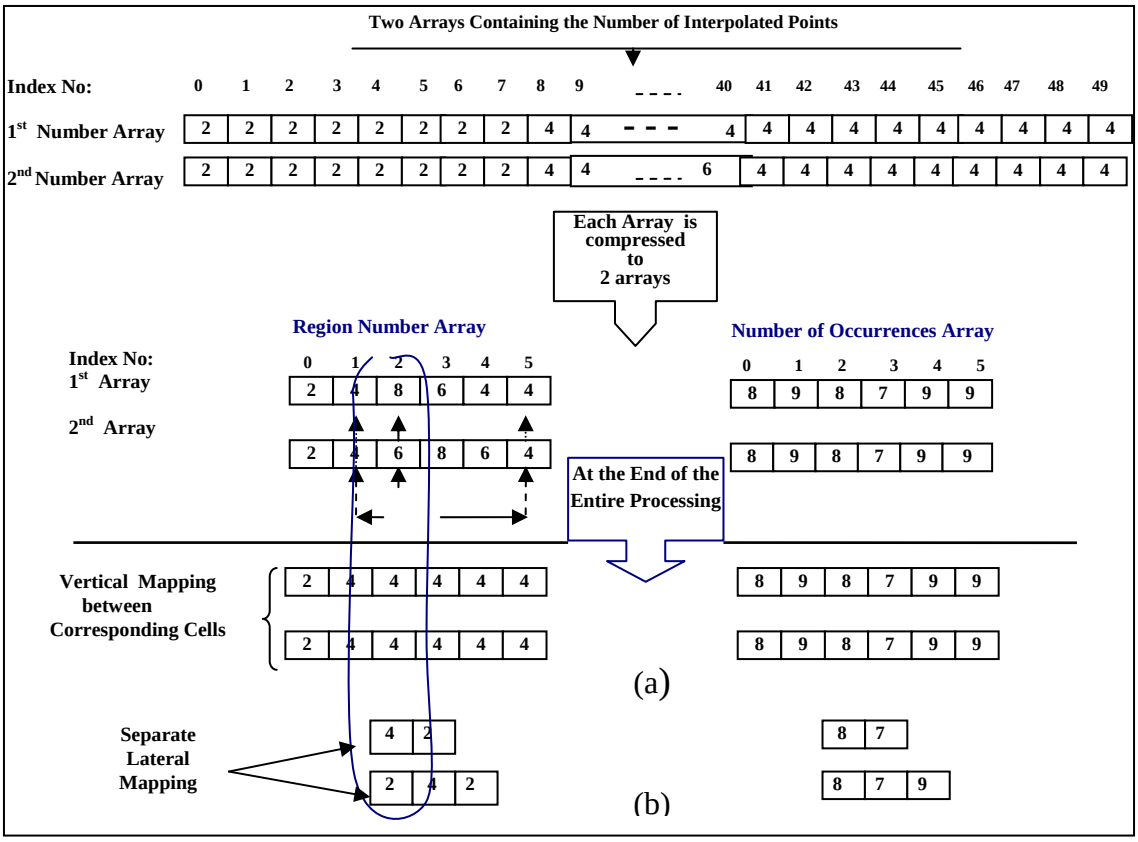

Figure 9: Basic Cell Construction between Two Consecutive Interpolated Boundaries.

4. IMPLEMENTATION AND RESULTS

The algorithm has been implemented using C++ with Visualization Tool Kit (VTK) as graphics platform. In Figure 10(a), transformed shape is constructed when the number of objects is three using '4' as the number of steps with the initial direction of traversal for the first data along the principal axis Y and the rest along the maximum direction of the eigen vector i.e. 'alignment = 0'. As the number of step increases, the number of slices is doubled which also increases the overall run time. Figure 10(b) shows the transformed output when the initial direction of traversal for all data is along the maximum direction of the eigen vector i.e. 'alignment = 0'. In Figure 10(c) transformed output is generated when the initial direction of traversal for all data is along the medium direction of the eigen vector i.e. 'alignment = 1' whereas in Figure 10(d), transformed output is generated with initial direction of traversal for all data along minimum direction of the eigen vector i.e. 'alignment = 2'.

Figure 11 shows the shape transformation when there are four input objects. In Figure 11(a), shape transfor – mation is performed with the initial direction of traver -sal for all data along the principal axis Y and in Figure11(b) transformed output is generated with initial direction of traversal for all data along minimum direc -tion of the eigen vector i.e. 'alignment = 2'.

Figure 12 compares the shape transformation as the ini -tial direction of traversal for the same input object cha -nges. In Figure 12(a), the initial direction of traversal for all data is along the maximum direction of the ei -gen vector i.e. 'alignment = 0' and in Figure 12(b), the initial direction of traversal for the first data along prin -cipal axis Y and the rest of the data is along the maxi -mum direction of the eigen vector i.e.'alignment = 0'.

Instead of incorporating the overall characteristics of all the objects into one, different parts of different objects can be incorporated in different proportions during shape transformation or some portions of some objects can be kept unchanged. This may result in a totally different objects which is a blend of a number of objects. Figure 13 shows shape transformation by blend -ing two objects in top and bottom portion while the middle portion is a blend of three objects. Figure 14 shows the gradual transformation between two objects

 ISBN 978-80-86943-94-7

where the initial directions of traversal for both objects are along the minimum direction of the eigen vector i.e 'alignment = 2'.

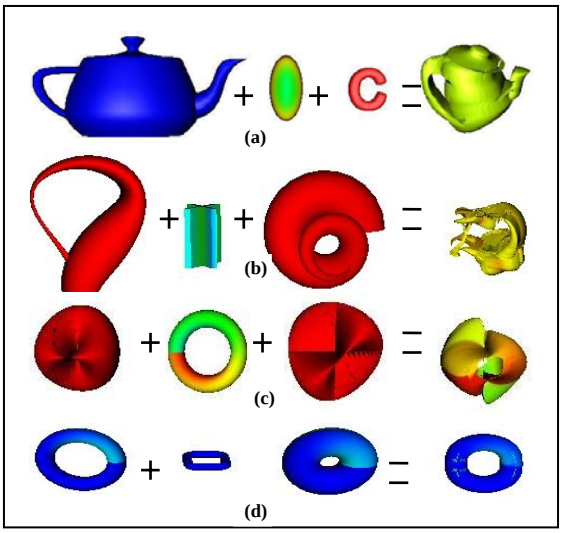

Figure 10: Shape Transformation when the Number of Input Objects is Equal to Three.

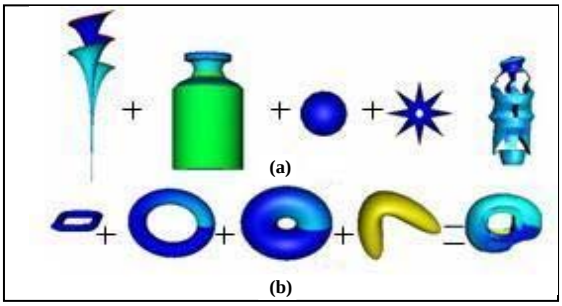

Figure 11: Shape Transformation when the Number of Input Objects is equal to Four.

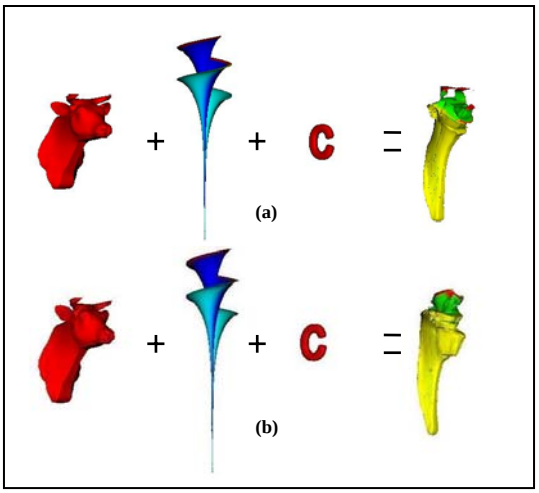

Figure 12: Shape Transformation for the same Input Objects with Different Initial Direction of Traversal.

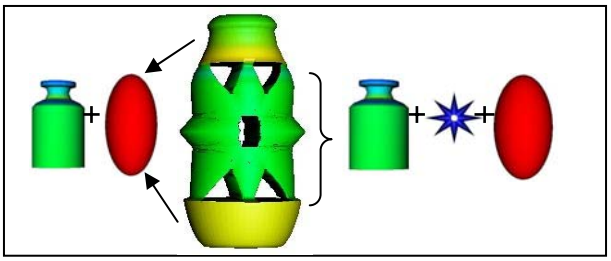

Figure 13: Shape Transformation by Blending Different Objects in Different Proportions.

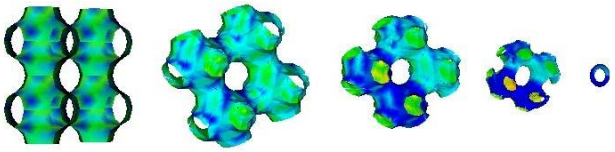

Figure 14: Gradual Shape Transformation between Two Objects (Leftmost and Rightmost).

The usual way of incorporating multiple influence shapes in shape transformation is to use different influence shapes at different stages of transformation in between two objects. Using multiple influence shape in this way does not incorporate the overall characteristics of all the influence shapes among all the intermediate objects. This shortcoming can be alleviated by using our technique. All the influence shapes are transformed first. Then the transformed object can be used as a single influence shape to produce a uniform blend of all the influence shapes during transformation (Figure 15).

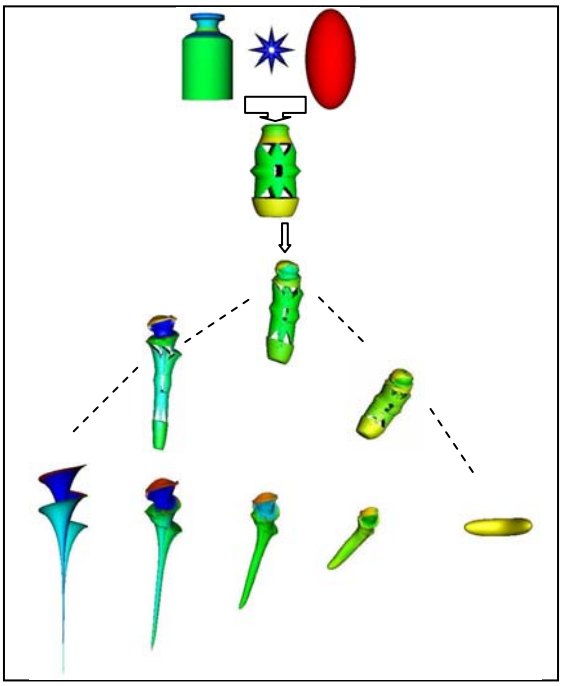

Figure 15: Transformation of Multiple Objects to One Single Object to Influence the Original Path of Morphing.

 ISBN 978-80-86943-94-7

5. DISCUSSION

This section compares the proposed algorithm with some other existing shape transformation algorithms. Most surface-based methods consider the distortion/ rotation of the rigid body, but division of both source/ target into a number of corresponding patches or meshes is needed at the expense of a large number of user input and longer pre-processing stage [Kent92, Lazarous94, Gregory98, Breen01]. Hence surface-based methods do not scale well when the number of input objects increases. The proposed algorithm works without any user input with default initial settings ('number of steps = 4' and 'alignment = max'). If variations in the number of steps and alignments are desired, the user just needs to specify these two variables. The number of slices can also be reduced by varying the number of steps. This automated method of reducing the number of slices as well as run time is absent in most other algorithms. In most other existing algorithms, specific number of user-defined disk fields [Chen96] or point/line fields [Kent92, Lazarous94, Gregory98, Breen01] are used. Varying these fields involves a considerable amount of user intervention and longer pre-processing time: minor variation in these fields can generate a major variation in the output. The proposed algorithm automatically traverses the data along the user-defined initial alignment and is free from any inaccurate user intervention and at the same time if needed, allows user to specify the initial direction of traversal. This generates different transformed output for the same set of data. Scalability is another major characteristics of the algorithm. This algorithm scales well when the number of input objects increases. Transformed shape maintains the geometric features of the input objects. Rotation/ distortion of the rigid body is also considered which is absent in some volume-based methods that use discrete mathematical function in shape transformation [Hughes92, He94, Turk99]. Some volume-based algorithms alleviate this problems [Payne92, Breen01]. However they are highly sensitive to the user-specified initial overlapping of the objects. Use of CSG in blending of a number of shapes is a bit rigid and its application is also limited [Pasko05].

6. CONCLUSION AND FUTURE WORK

Simplicity and flexibility are two major characteristics of the algorithm which have made it more dynamic and extendible over other existing shape transformation algorithms. Future work includes exploitation of the me thod in parallel / distributed computing environment as simple data structure of sliced body and binary subdivision are suitable for both data and functional partitioning.

7. REFERENCES

[Breen01] D. E. Breen and R. T. Whitaker. A Level-Set Approach for the Metamorphosis of Solid Models. IEEE Transactions on Visualization and Computer Graphics, 7(2), 2001, pp. 173-192.

[Chen96] M. Chen, M. W. Jones and P. Townsend. Volume Distortion and Morphing Using Disk Fields. Computers and Graphics, 24(2), 1996, pp. 567-575.

[Gregory98] A. Gregory, A. State, M. Lin, D. Manocha and M. Livingston. Feature-based Surface Decomposition for Correspondence and Morphing between Polyhedra. Proceedings of Computer Animation, 1998, pp. 64-71.

[He94] T. He, S. Wang and A. Kauffman. Wavelet-based Volume Morphing. IEEE Visualization Proceedings, 1994, pp. 85-92.

[Hong88] T. Hong, N. Magnenat-Thalmann and D. Thalmann. A General Algorithm for 3D Shape Interpolation in a Facet-based Representation. Proceedings of Graphics Interface, 1988, pp. 229-235.

[Hughes92] J. F. Hughes. Scheduled Fourier Volume Morphing. ACM SIGGRAPH Computer Graphics, 26(2), 1992, pp. 43-46.

[Kaul92] A. Kaul and J. Rossignac. Solid-interpolating Deformations: Constructionand Animation of PIPs. Computers and Graphics, 16(1), 1992, pp. 107-115.

[Kent92] J. R. Kent, W. E. Carlson and R. E. Parent. Shape Transformation for Polyhedral Objects. Computer Graphics, 26(2), 1992, pp. 47-54.

[Lazarous94] F. Lazarous and A. Verroust. Feature-based Shape Transformation for Polyhedral Objects. Fifth Eurographics Workshop on Animation and Simulation, 1994, pp. 241-254.

[Lee99] A. W. F. Lee, D. Dobkin, W. Sweldens and P. Schroder. Multiresolution Mesh Morphing. Proceedings of ACM SIGGRAPH , 1999, pp. 343-350.

[Lerois95] A. Lerois, C. D. Garfinkle and M. Levoy. Feature-based Volume Metamorphosis. Computer Graphics, 29, "Annual Conference Series", 1995, pp. 449-456.

[Lin96] S. Gottschalk, M. C. Lin and D. Manocha. OBBTree: A Hierarchical Structure for Rapid Interference Detection. Computers and Graphics (30), "Annual Conference Series", 1996, pp. 171-180.

[Pasko05] G. Pasko, A. Pasko and T. L. Kunii. Bounded Blending for Function-based Shape Modeling. IEEE Computer Graphics and Applications, 25(2), 2005, pp. 36-45.

[Payne92] B. Payne and A. Toga. Distance Field Manipu - lation of Surface Models. IEEE Computer Graphics and Applications, 12(1), 1992, pp. 65-71.

[Turk99] G. Turk and J. F. O'Brien. Shape Transformation Using Variational Implicit Functions. Proceedings of ACM SIGGRAPH , 1999, pp. 335-342.