Interactive Deformation with Triangles

James Dean Palmer and Ergun Akleman[*]
Visualization Sciences Program
Texas A&M University

Jianer Chen
Department of Computer Science
Texas A&M University

Abstract

In this paper, we present a new deformation technique based on triangular deformers. One of the advantage of triangles over lines is that triangles uniquely define three linearly independent vectors in 3D. These three vectors can be used as a local coordinate system. Deformation is described by a set of source and destination triangles. For each source and destination triangle and for any point 3D space, we can compute a unique transformation vector for any given source and destination triangles.

We combine these transformation vectors described by each deformer pair, by using a weigted average that is a function of the distance to the source traingle. In the paper, we provide a set of blending functions to effectively interpolate these transformation vectors from each triangle to compute a combined transformation. To obtain a deformation of a given polygonal mesh, we simply translate each vertex of the polygonal mesh with a combined transformation vector that is computed by using the position of the vertex.

Our technique can be used for both 2D and 3D deformation. We have implemented systems for interactive 2D and 3D deformations. In our 2D implementation, we use line and point deformers in addition to triangles.

1. Introduction and Motivation

This paper presents a new deformation technique that is based on triangle deformers. The first deformation method in computer graphics was introduced by Alan Barr in 1984 [2]. In 1986, Sederberg and Parry introduced *Free-Form Deformations* (FFD), which involves a mapping from one coordinate space to another through a trivariate tensor product Bernstein polynomial [12]. Over the years a number of variations on FFD have been introduced. These include Coquillart's *Extended Free-Form Deformation* (EFFD) [6] and Lazaris, Coquillart and Jansene's *Axial Deformations* [8]. Jin, Li and Peng used generalized metaballs for general constrained deformations [10]. Borrel and Rappoport introduced *Constrained Deformations* [4], Singh and Fiume proposed *Wires* [13] and Crespin introduced *Implicit Free-Form Deformations* (IFFD). IFFD provides a frame work in which most of the Free-Form Deformation techniques can be used but is based on deformers defined by a local tool and a blending function [7]. Within the IFFD model, a set of deformers act on a model. Each deformer is characterized by a local coordinate space, an invertible mapping function, an invertible transformation function and a potential function. These deformers are then applied to geometry using a process similar to FFD [7].

Crespin's work inspired our initial investigations with using implicit functions and simplicial complexes to deform geometry. We also observed a number of similarities between our technique and the *Feature-Based Image Metamorphosis* (FBIM), which is introduced in 1992 by Beier and Neely [3] and has been used as a visual effect in a number of motion pictures as well as the Michael Jackson's video *Black and White*. FBIM uses line deformers to deform the location of pixel with inverse mapping [3]. Although, FBIM is not really a deformation approach, the underlying transformation algorithm shares many similarities to deformation algorithms.

Mathematically, there are two differences between our technique and FBIM technique:

- Our deformations are based on trangles instead of lines.
- Our technique is a forward mapping instead of inverse.

Line deformers are appropriate for only 2D deformations; For 3D deformations, triangle deformers are needed. The main advantage of triangles over lines in 3D is that triangles uniquely define three linearly independent vectors. These three vectors can be used as a local coordinate system to compute a unique transformation vector for any given source and destination triangles. Another advantage of using triangle deformers is that triangles can effectively ex-

[*]Corresponding Author. Address: 216 Langford Center, College Station, Texas 77843-3137. email: ergun@viz.tamu.edu. phone: +(979) 845-6599. fax: +(979) 845-4491. Supported in part by the Texas A&M, Interdisciplinary Program.

press non-uniform scaling and shear from the source to the destination.

By using a weigted average, we combine the transformation vectors described by each deformer pair. The weights are functions of the distance to the source traingle. They are similar to implicit blending functions. For predictable results, they must be always positive and monotone decreasing. In the paper, we provide a set of blending functions. To obtain a deformation of a given polygonal mesh, we simply translate each vertex of the polygonal mesh with a combined transformation vector that is computed by using the position of the vertex.

We have implemented systems for interactive 2D and 3D deformations. In our 2D implementation, we use line and point deformers in addition to triangles. In 2D, instead of using FBIM's inverse mapping technique, we use forward mapping functions to manipulate the geometry of the grid. The deformation of the grid also deforms the texture that is mapped to the grid. This approach takes the advantage of 3D acceleration hardware that supports texture mapping and provides interactive deformations in 2D. Our 3D implementation uses exactly the same deformation concept as the 2D implementation but in 3D we only support triangle deformers to uniquely define a local coordinate system.

2. Methodology

Our approach is based on functions that are constructed by the operations that are used in implicit surface construction. The overall deformation is described by a set of deformer pairs (point, line and triangle pairs). Each deformer pair consists of one source and one destination. For each source and destination shape a local coordinate is computed to describe the transformation described by the change of position and shape between the source and destination. In addition, for each shape a weight function that is described by the distance to the shape is given. Then, the transformations are combined by using these weight function.

In our framework, to describe a deformation, the users defines a set of deformer pairs. Each deformer pair consists of two shapes: the source shape and the destination shape. Each one of these pairs uniquely describe a transformation vector for any given point in the source image. In this section, we discuss the effect of different types of deformer pairs.

The simplest of these deformers are point pairs. A point deformer can uniquely define transformation in any dimension. As can be observed from Figure 1.A, point deformer can only implement translations, but not allow scaling, rotating, or shearing.

Beier and Neely uses line deformer pairs [3]. An advantage of using lines over a simple point deformer is that one can describe scaling and rotation up to 360 degree. Figure 1.B shows a source image is translated, rotated, and scaled from the source line to the destination line by changing the coordinate system to be relative to the source and destination lines. As can be observed from the image, the line primitive doesn't allow us to shear the image and we can only scale the image uniformly. We cannot, for example, scale only the width or scale only the height. Another problem with line deformers is that in 3D deformation not uniquely defined.

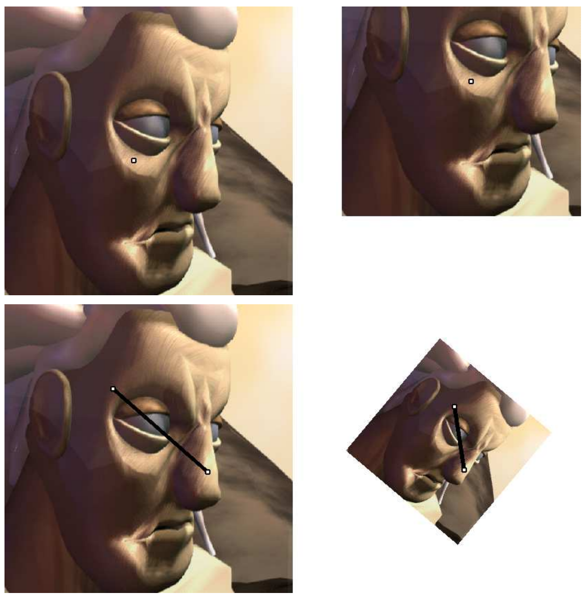

Figure 1. Examples of point and line deformers.

On the other hand, triangular deformers allow us to effectively translate, rotate, scale, and shear an image and uniquely describe a transformation. Moreover, unlike line deformers, deformations uniquely defined by triangle deformers in 3D. Figure 2.A shows the effect of triangle deformers. The source image is translated, rotated, scaled and sheared from the source triangle to the destination triangle by changing the coordinate system to be relative to the source and destination triangles. Figure 2.B shows an example of 3D deformation of a teapot by a single triangle deformer pair.

If there exists more than one deformer, the problem is to appropriately combine the transformations described by each deformer pair. In order to compute the combined transformation, we simply calculate a weighted average of transformations given by each simplex pair. Weights are computed based on the distance of a deformer to a given point.

Figure 3.A illustrates how two triangles can be used together to deform an image. One application of image deformation is caricature. Akleman, Palmer and Logan have

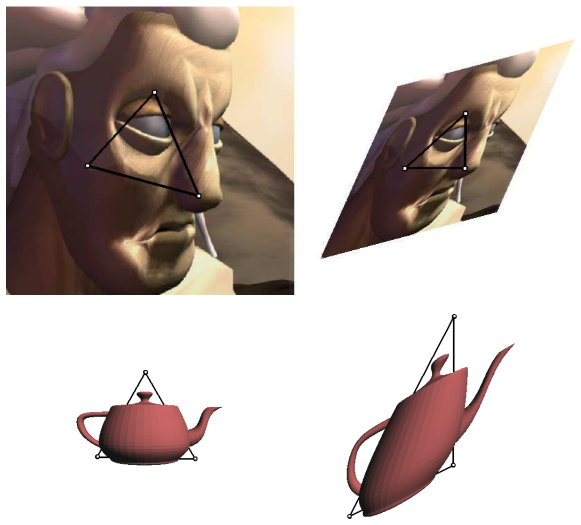

Figure 2. 2D and 3D examples of triangle deformers.

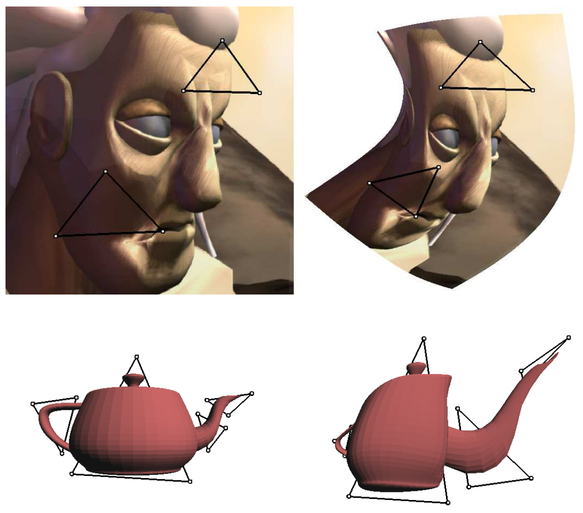

Figure 3. 2D and 3D examples of blended triangle deformers.

recently used the 2D deformation system for generating extreme caricatures [1]. Figure 3.B shows 3D deformations with multiple deformers. In this example, we have added a small deformer to the teapot handle while using another deformer to actually make the teapot body longer. We then used two deformers on the spout to make it large at the base of the spout but small and narrow at the tip of the spout.

In 2D, if we use only line primitives, FBIM's weight function and inverse maps instead of forward ones, our deformation technique will be exactly the same as the FBIM technique. In other words, our technique can be considered as a generalization of the FBIM technique. Our generalization comes from (1) using triangle and point primitives in addition to line primitives, (2) using new weight functions and (3) using forward maps instead of inverse ones. By using triangles in addition to lines, we provide users with shear transformations in addition to translation, scaling and rotation. And by using forward maps, we are able to develop an interactive system by taking advantage of texture mapping hardware.

Our 3D approach can be considered a straight generalization of FBIM to 3D since lines are natural simplest deformers for 2D and triangles are natural deformers for 3D.

3. Implementation

Several important implementation choices had to be made in developing software to test and analyze the algorithms we have developed. These issues include what language to write the application in, what user interface toolkit to use, and what graphic toolkit to use.

Our implementation is written in C++ which is an object oriented language. By using an object oriented approach in writing this software we were able to abstract the deformers into pluggable objects. In 2D, a triangle deformer can work seemlessly with a point or line deformer. Furthermore, new deformers can be added to this system without having to change much existing code and the new objects can work seemlessly with the deformers we have already developed.

We chose to use OpenGL as the graphic toolkit since OpenGL supports hardware texture and 3D acceleration. By using forward mapping functions to manipulate texture mapped grids, we can harness the hardware accelerated texture mapping support to interactively deform 2D images. While hardware accelerated texture mapping and 2D/3D transformations are used extensively in games and 3D applications, most 2D applications don't utilize the functionality that is becoming standard in the newest generation of consumer display adaptors. The application we have developed is a good example of using hardware acceleration generally intended for use in 3D within a 2D application. And of course, OpenGL was an excellent choice when we extended our 2D deformation engine to 3D as well.

Fltk was chosen as the GUI (graphic user interface) toolkit, because it is very portable and has excellent OpenGL support. The applications we have developed have been compiled under Irix and Linux. Porting them to other platforms, such as MS Windows, should not be difficult.

2D user interface that we have developed supports point, line, and triangle primitives. It also supports several different blending functions and it is extremely easy to add other blending functions. We can also vary the blending constants in these equations to gauge their effect. 3D user interface supports loading OBJ format 3D models instead of 2D images. The interface also provides convenient tools to navigate about the object or scene being deformed. It also supports multiple "frames" so one can do simple key framed animation. The 3D implementation also supports all of the same blending features that our 2D implementation does.

4. Results and Discussion

Figure 4.A shows that drastic deformations can be obtained with only a few deformers. In this example, we changed a neutral character into a grumpyone by turning down the nose and manipulating the mouth into a frown. Figure 4.B demonstrates a head that has been deformed into a happy character. We have used the deformers to change the mouth into a smile and we have moved the forehead and brow higher. Figures 4.A and 4.B also show that we don't necessarily need to be extremely accurate in bounding features with triangle deformers. Figure 5.A shows a nose feature that has been closely bounded by a triangle deformer. The deformed feature stays relatively close to the destination triangle. Figure 5.B shows a nose feature that has been loosely bounded by a triangle deformer. Since the deformer is effectively deforming all of the space that it bounds, it will not pull the nose feature out as far as the closely bounded example did.

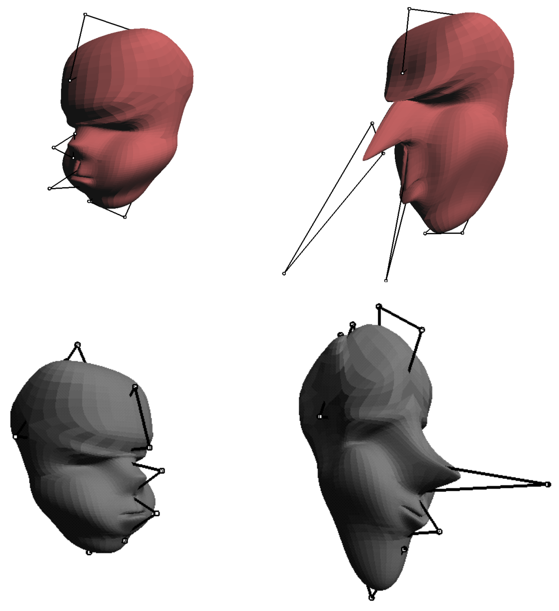

Figure 4. Deformation of a human head model to create grumpy and happy characters.

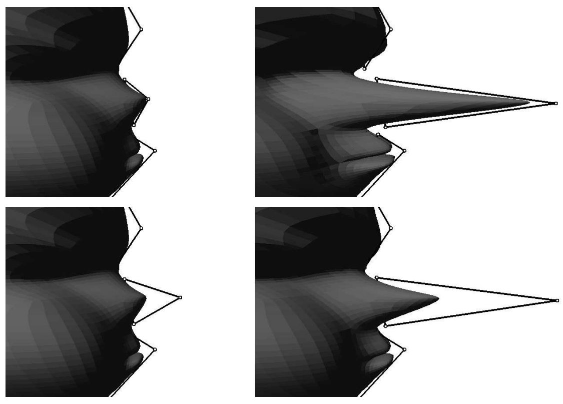

Figure 5. Deformation of a nose to create different noses.

5. Future Work and Conclusion

We developed a new interactive deformation technique. This technique improves on existing techniques and provides a powerful framework for future work. The simplicial complex based deformation algorithms that we are working with should work well across n dimensions and have a plethora of applications. Applications for this work include 2D morphing and warping, as well as 3D warping, modeling, and animation.

One limitation of the local coordinate based approaches such as this technique or FBIM is that the coordinate transformation can not express a rotation of more than 360 degrees. If we had defined a coordinate mapping transformation in terms of scale rotation and translation we could have expressed rotations more than 360 degrees. We have already developed and have had some initial successes in using this alternate mapping framework, but future work is necessary to extend it to 3D and to compare and contrast it to the local coordinate based approach.

By using C++'s object oriented inheritance we can easily define new deformers that can be plugged into our application and will blend seemlessly with existing deformers. We intend to extend our current work to higher dimensions and then we intend to define distance functions for more complex simplicial complexes. Simplicial complexes and operations on simplicial complexes are well defined in higher dimensions. By building our algorithms' framework on these structures our algorithm should easily extend from 2D to 3D and potentially to even higher dimensions.

More generally our algorithm should be applicable to simplicial complexes. A simplex is a set of d+1 points whose convex hull has dimension d. The points of the simplex may exist in a space whose dimension is larger than d. In 2D, simplexes includes points, lines and triangles. A simplicial complex is composed of a number of simplices.

The intrinsic dimension of the complex is the dimension of each simplex in the complex. The embedded dimension of the simplicial complex is the dimension of the space of the points in the simplicial complex.

Simplicial complexes represent a straightforward and well defined data structure that allow one to take advantage of linear programming methods for the solution of geometric problems, boundary evaluation, affine transformations, subdivision, and constructive solid geometry operators. Simplicial complexes also provide a very simple and general method for expressing geometry in n-dimensional space. [11]

Using this technique with simplicial complexes instead of the more specific cases we have used in this research would allow a more general framework of deformation tools that scale to N dimensions and can represent more complex deformations.

References

[1] E. Akleman, J. D. Palmer, R. Logan. Making Extreme Caricatures with a New Interactive 2D Deformation Technique with Simplicial Complexes. *Visual 2000 Proceedings*, 100-105, Sept. 2000.

[2] A. H. Barr. Global and local deformations of solid primitives. *Computer Graphics SIGGRAPH '84 Proceedings*, 18(3):21-30, 1984.

[3] Beier and Neely. Feature-Based Image Metamorphosis. *Computer Graphics*, 26(2):35-42, 1992.

[4] P. Borrel and A. Rappoport. Simple constrained deformations for geometric modeling and interactive design. *ACM Transactions on Graphics*, 13(2):137-155, 1994.

[5] E. Brisson. Representing geometric structures in d dimensions: topology and order *ACM Symposium on Computational Geometry*,218-227, 1989.

[6] S. Coquillart. Extended Free-Form Deformation : A Sculpturing Tool for 3D Geometric Modeling. *Computer Graphics SIGGRAPH '90 Proceedings*, 24(4):187-196, 1990.

[7] B. Crespin. Implicit Free-Form Deformations. *Implicit Surfaces*, 1999.

[8] F. Lazarus, S. Coquillart, and P. Jancene. Axial deformations: an intuitive deformation technique. *Computer Aided Design*, 26(8):607-613, 1994.

[9] R. Logan. *Automated Interactive Facial Caricature Generation.* Masters Thesis: Texas A&M University. December 2000.

[10] X. Jin, Y. Li and Q. Peng. General Constrained Deformations Based on Generalized Metaballs. *Computers & Graphics*, 24:219-231, 2000.

[11] A. Paoluzzi, F. Bernardini, C. Cattani and V. Ferrucci. Dimension-independent modeling with simplicial complexes. *ACM Transactions on Graphics*,12(1):56-102, 1993.

[12] T. W. Sederberg, S. R. Parry. Free-form Deformations of solid geometric models. *Computer Graphics SIGGRAPH '86 Proceedings*, 20(4):151-160, 1986.

[13] K. Singh and E. Fiume "Wires: a geometric deformation technique" *Computer Graphics SIGGRAPH '98 Proceedings*, 405-414, 1998.

[14] B. Wyvill and G. Wyvill. Field Functions for Implicit Surfaces. *Visual Computer*, 5:75-82, 1989.