EUROGRAPHICS 2002 / I. Navazo Alvaro and Ph. Slusallek
(Guest Editors)

Real-time 3D Deformations by Means of Compactly Supported Radial Basis Functions

Nikita Kojekine, Vladimir Savchenko[†], Mikhail Senin[‡], Ichiro Hagiwara

Faculty of Engineering, Tokyo Institute of Technology, 2-12-1, O-okayama, Meguro-ku, Tokyo 152-8552, Japan.

Abstract

We present an approach to real-time animation of deformable objects. Optimization of algorithms using compactly supported radial basis functions (CSRBF) allows us to generate deformations performed fast enough for such real-time applications as computer games. The algorithm described in detail in this paper uses space mapping technique. Smooth local deformations of animation objects can be defined by only a moderate number of control vectors and locality of deformations can be defined by radius of support. We also present examples of animations and speed benchmarks.

Categories and Subject Descriptors (according to ACM CCS): I.3.7 [Computer Graphics - Three-Dimensional Graphics and Realism]: Animation

1. Introduction

Many recent works have focused on using shape transformation as a basic operation in computer graphics. Here, we consider the problem of transformation a given geometric shape into another in a continuous manner. In fact, if we want to use the method of fitting a surface into a set of control points as a design tool, it is important to assure that our fitting method does what our intuition would expect. On the other hand, we want to perform such deformations in real-time. Important examples of surface deformations have been investigated during the past few years. Various strategies were proposed to minimize user interaction; however, the problem of fitting a surface to a set of control points in real-time still remains a largely unsolved issue of great practical importance.

Our main goal was to obtain an algorithm performing fast, plausible and smooth deformations. For this purpose we optimize the algorithms to reduce their computation time. For example, these algorithms can be applied, in computer games. This optimization not only allows to perform complex deformations in real-time, but also saves memory required for animation. For example, in the most popular 3D game engines (Quake, Unreal, Half-Life) for skeleton animation of models all coordinates of all points of an object are stored for every frame of animation, what leads to huge amount of data stored for each model. When we use the described technique, only a small amount of additional information is needed.

In this paper we propose a CSRBF-based [1] mechanism for calculating the interpolated points of a deformed surface of the animation object. This paper extends the work of Kojekine et al. [2], where software tools based on CSRBFs were designed and applied for surface reconstruction of 3D geometric objects.

The rest of the paper is organized as follows. The next section gives an overview of shape transformation techniques and of works related to animation and surface reconstruction and deformation problems. In Section 3 we discuss the mathematical background of our algorithm, and in Section 4 we present the algorithm. Sections 5 and 6 show examples of animation and speed benchmarks. Section 7 contains conclusions and discusses the future work.

[†] Faculty of Computer and Information Sciences, Hosei University, 3-7-2 Kajino-cho Koganei-shi, Tokyo 184-8584, Japan.
[‡] Moscow Institute of Physics and Technology, Kerchenskaya str., house 1"A", building 1, Moscow 113-303, Russia

2. Related work

Most existing shape transformation techniques fall into one of the three following categories:

- mapping the space onto itself;
- metamorphosis;
- modification of defining functions.

It is out of scope of this paper to make a detailed review of all transformation techniques, we only briefly mention the most popular ones. An interested reader can find more references in the paper [3].

A mapping can be controlled by numerical parameters of predefined functions, by set of control points and by differential equations. Free-Form Deformations (FFD) are well-known examples of space mappings, that were pioneered by Sederberg and Parry [4] and extended in [5], [6], [7]. User-defined point lattices control FFD. Borrel and Bechmann [8] proposed general deformation techniques for space mappings. These techniques provide forward and inverse mappings that suite better to implicit surfaces.

Most of deformation methods are too global to provide series of small bumps defined by arbitrary points. Although the method of Borrel and Rappoport [9] has been designed for localized space mappings, it can lead to non-intuitive results when bounding spheres of several control points intersect.

The survey [10] discusses common mathematical foundations of the space deformation techniques. Vast literature is devoted to the subject of scattered data interpolation, which can be used for a space mapping, and if applied to some point set in the space, it changes this set into a different one. One of the approaches is to use methods of scattered data interpolation, based on the minimum-energy properties [11], [12], [13]. These methods are widely discussed in literature (see [14], [15]). As far as we know, the first publication on use of discrete 2D landmark points is that of Bookstein [16], [17]. In paper [18] Bookstein discussed a method where destination points are selected to form the configuration of minimum bending energy.

The benefits of using radial basis functions (RBF) have been recognized in many works and RBFs were adapted for 2D and 3D computer animation [19], [20], medical application [21], [16], and for reconstruction from 3D scattered data. To reduce the processing time special methods were developed for thin plate splines and discussed in [22], [23], see also recent publications [24], [25].

Actually, the methods that use the RBFs can be divided into three groups. The first group consists of "naive" methods, that are restricted to small problems, but they work quite well in applications that deal with shape transformation (see, for example [26], [20]). This methods are computationally expensive even for small data sets and cannot be used for real-time animation. The second group consists of fast methods that allow modeling of large data sets [25], [22].

The third and the last group are the CSRBFs [1]. Recently they were applied to reconstruction of scattered data sets [27], [2]. The very good overview of related works, problems and limitations could be found in the paper [28], which also describes the problems of scattered data interpolation and introduces very fast algorithm for constructing C^2-continous interpolation functions.

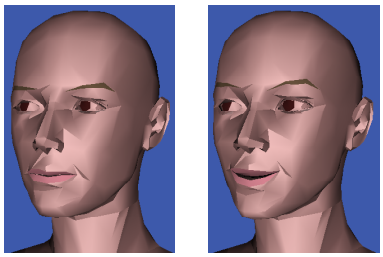

Figure 1: *An example of facial animation.*

The ability to transform the shape of a surface is useful in animation, especially for face simulation. The problems in this research area still remain among the most difficult. Researches have devoted significant efforts to this problem [32], [33]. For more references, see [34]. In recent years, modeling virtual actors has attracted great interest and a lot of attention has been paid to the synthesis of the face expression of the speaker. Two approaches have been dominant: the first one works with 3D models to control the face movements, and the second one uses 2D images. Those who are interested can visit the web site [35] to look over examples of the generated actors. Our approach can also be applied for real-time facial animation, see for example Figure 1 (benchmark details are in Figure 10 at the end of this article).

Shape transformation is also a useful tool for forensic identification. The book [36] presents a good overview of existing methods for shape reconstruction and modification. The attempts to simulate visco-elastically deformable materials have been concentrated on mesh descriptions of cloth and other soft objects such as the muscles and skin [37], [38], [39], [40], [41], [42], [43], and mostly used the physics-based simulation. In addition to the shape modification, there is a challenging problem of modeling three-dimensional faces. We refer interested readers to the paper [44]. 3D geometric modeling systems based on shape deformations have been proposed by many researchers who use the simple idea that tangible geometry of deformations can be defined by the user-defined starting and destination points. Probably this approach was firstly implemented for 2D morphing in the papers of Wolberg [45], Beier and Neely [46].

It is clear that the problem of producing fast, plausible (especially for such complex cases as facial simulations) and smooth deformations is an important and largely unsolved problem.

3. Shape deformations using CSRBF

One of the approaches to the animation problem is to warp the space where an object is embedded. The object can contain a skeleton, which simulates its characteristic behavior (see [29], [30]), and an outer surface layer. In this case overall shape transformation can be represented as the sum of global skeleton-based displacements and local displacements are defined using scattered data interpolation technique. A unified approach can also be applied, see [31].

To describe a local deformation of the outer surface layer, we can construct a mapping of the space that is determined by the mapping of a small number of *control points* — points whose images are predefined (see Figure 2.a). The space mapping is considered to be an elastic deformation, following various papers, see for instance [18].

Interpolations of scattered data using CSRBFs demonstrate good blending features and an example shows that they produce what our intuition would expect. Figures 2.b and 2.c illustrate the fact that the resulting mapping depends on a set of control vectors and the radius of support that defines the locality of the deformation.

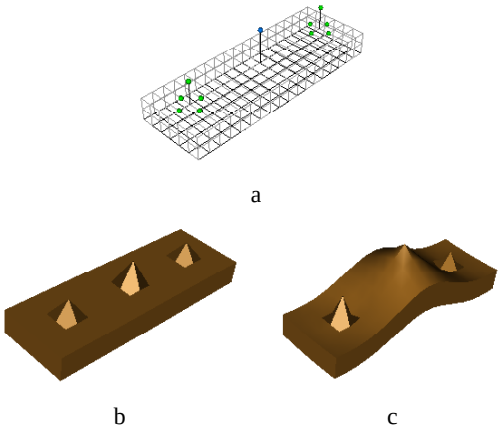

Figure 2: ***a*** *— An example of a transformation of a brick. Vectors show "mapping" directions and values;* ***b, c*** *— Different possible configurations of the initial form depending on the radius of support.*

To construct an $\mathbb{R}^3 \to \mathbb{R}^3$ mapping we can use the CSRBFs multivariate interpolation. Suppose a set of pairwise distinct control points $X = \{\vec{x}_1, \ldots, \vec{x}_N\} \subseteq \mathbb{R}^3$ is given. Suppose further, we know the values $g_1, \ldots, g_N$ at the control points and we search for a continuous function that interpolates these values at the control points. Then the RBF interpolant has the form

$$s_{g,X}(\vec{x}) = \sum_{i=1}^{N} \alpha_i \phi(\|\vec{x} - \vec{x}_i\|) + p(\vec{x}) \qquad (1)$$

where $\|\cdot\|$ denotes the usual Euclidean norm in $\mathbb{R}^3$, and p is a low degree polynomial. The coefficients α_j and the polynomial p are determined by the interpolation conditions

$$s_{g,X}(\vec{x}_j) = g_i, \quad 1 \leqslant i \leqslant N, \qquad (2)$$

and the additional requirements

$$\sum_{i=1}^{N} \alpha_i q(\vec{x}_j) = 0 \qquad (3)$$

for all polynomials q of degree $\deg(q) \leqslant \deg(p)$. This method of interpolation exactly reproduces polynomials of degree $m \leqslant \deg(p)$, whenever interpolation is unique.

In our application we follow Wendland [1], who constructed for $\mathbb{R}^3$ a new class of positive definite and compactly supported radial functions, that have the form

$$\phi(r) = \begin{cases} \psi(r), & 0 \leqslant r \leqslant 1 \\ 0, & r > 1 \end{cases}, \qquad (4)$$

where $\psi(r)$ is a univariate polynomial, and its radius of support is equal to 1. Scaling of the function $\psi(r/\alpha))$ allows any desired radius of support α.

In our animation application we have selected a simple function $\psi_{2,0}(r) = (1-r)_+^2$, which supports C^0 continuity. This function is simple to calculate and, according to our experiments, it produces plausible deformations. However, other functions, that support higher continuity, can be used (see Figure 9). For more references see [47].

The interpolation is unique because ϕ is a positive definite function in our case.

Given two data sets in $\mathbb{R}^3$, $\vec{s}_i = \{x_s^i, y_s^i, z_s^i\}, 1 \leqslant i \leqslant N$ for a non-deformed object, and $\vec{d}_i = \{x_d^i, y_d^i, z_d^i\}, 1 \leqslant i \leqslant N$ for the deformed object, we can construct a space mapping $T : \mathbb{R}^3 \to \mathbb{R}^3$ which is a CSRBF interpolation of form (1) in each of its components. We use a linear polynomial $p(\vec{x}) = \beta_0 + \beta_1 x + \beta_2 y + \beta_3 z$. This guarantees that T will be an affine transformation whenever the interpolation data admit such transformation. The conditions (2) and (3) present a system of liner algebraic equations (SLAE) on the coefficients $\alpha_i, \beta_0, \beta_1, \beta_2, \beta_3$.

Since function ϕ is compactly supported, the resulting SLAE has a sparse matrix. We have developed an algorithm (to be discussed in Section 4.2), which allows to sort initial data points in a special way, so that the resulting matrix T, which has the size $(N+4) \times (N+4)$, is not only sparse, but also has a band-diagonal submatrix $A = \{a_{ij}\}$, $a_{ij} = \phi(\|\vec{x}_i - \vec{x}_j\|)$, $1 \leqslant i, j \leqslant N$ of size $N \times N$. After constructing this band-diagonal submatrix it is possible to store it in memory in a very compact way and also to solve the SLAE by using a fast and simple direct solver. These speeds up computations significantly and makes it possible to use our approach in real-time animation.

4. Algorithm

The general scheme of our algorithm consists of the following steps:

- sorting scattered data,
- constructing a SLAE,
- solving the SLAE,
- evaluating the transformation.

While the solution of the system is the limiting step, constructing the matrix and evaluating the functions may also be computationally expensive.

4.1. Sorting scattered data

User defines the set of control vectors to describe the desired shape deformation. This set of control vectors can be defined as two lists of points. The first list contains initial positions of control points, possibly lying on the surface of the object. The second list contains destination positions of control points. First list is used to create a submatrix A; this is the left-hand side of the corresponding SLAE. The second list is used to create the right-hand sides for displacements along x, y and z directions.

We use space recursive subdivision for sorting the first list of control points, because it is an elegant and popular way of sorting scattered 3D data. An efficient approach based on the use of variable-depth octree for space subdivision allows us to obtain the resulting matrix as a band-diagonal matrix and to reduce the computational complexity.

Our first goal is to build an octree (see for example [48]) data structure from the original point data. We follow the procedure described in [2]. This octree is used to search for neighbors of any given point from the given N points. The neighbors are points of the sphere of radius R, whose center is located at the given point, where R is the user-defined radius of support for CSRBF.

4.2. Constructing a SLAE

Figure 3 presents a pseudo-code for our algorithm that is used to obtain a new order (`output_list`) of initial control points (`input_list`). In this way the submatrix A is constructed as band-diagonal. The maximum size of the band equals the maximum possible number of neighbors inside the selected radius of support R of a point from initial control points list. A typical example of the constructed band-diagonal submatrix is illustrated in Figure 4.

Obtaining submatrix A as band-diagonal has two advantages. One is that it is possible to store this matrix in a compact way, another one that it can be solved by using a simple, reliable and fast direct solver. In our application, we use the so-called profile form or the slightly modified Jennings envelope scheme [49] to store a band-diagonal matrix. To store the submatrix A, an array can be used for diagonal elements;

```
input_list — unsorted list of points
output_list — sorted list of points
neighbors_ids_list — temporary list of integers

i := 0
while (input_list.length ⩾ 0) do
begin
   // add first element of input_list to output_list
   // remove first element from input_list
   output_list.add(input_list[0]);
   input_list.remove(0);
   while (i < output_list.length) do
   begin
      // find in input_list all neighbors of output_list[i] and
      // put their indices into neighbors_ids_list
      // this can be done with the help of octree
      neighbors_ids_list=
           FindNeighbors(output_list[i],input_list);
      for j := 0 to neighbors_ids_list.length do
      begin
         // add neighbor element of input_list to output_list
         // remove this element from input_list
         output_list.add(input_list[neighbors_list[j]]);
         input_list.remove(neighbors_ids_list[j]);
      end
   end
end
```

Figure 3: *Algorithm of sorting initial data for obtaining a band-diagonal matrix.*

values of non-diagonal elements and correspondent indices of the first non-zero elements in the matrix lines are placed in two additional arrays. Naturally, the size of the band depends on the selected radius of support R. For the example shown in Figure 1 the storing of the full matrix of floats will result in 4096 bytes required, while with $R = 0.1$ it will be only 276 bytes using our scheme. With $R = 0.2$ it will be only 332 bytes (the result shown in the picture), and 1148 bytes for $R = 0.5$. In the example shown in Figure 8 our storage requires only 6320 bytes, not 33856 bytes as the full storage requires. This issue is more important for large deformations. Note that user-selected radius of support R defines both the locality and the speed and memory requirements for the deformation.

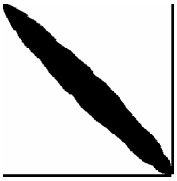

Figure 4: *An example of a typical band-diagonal matrix constructed by the proposed algorithm.*

4.3. SLAE solution

Note that solving any sparse system has goals of saving time and space. The attractiveness of using implicit methods such as conjugate gradient methods for large sparse systems has been well recognized in different applications. If

A is positive definite and symmetric, the algorithm cannot break down, but only in theory [50]. Conjugate gradient methods work well for matrices that are well-conditioned. In practical applications, this restriction can limit the accuracy with which a solution can be obtained, and thus we prefer to use explicit SLAE solution methods for matrices stored in profile form. The advantage of Gaussian LU decomposition [51] has been well recognized and many software routines were developed. For a symmetric and positive definite matrix, a special factorization, called Cholesky decomposition, is about two times faster than alternative methods for solving linear equations. Unfortunately, we could not find a "substantial" collection of routines for sparse matrix calculation in [52] which forced us to develop in C++ our own tool for SLAE solution. A combination of block Gauss solution and Cholesky decomposition was proposed by George and Liu in [50], and we follow their proposal in our software tool. After breaking up one linear set into a triangular set of equations, these equations can be solved by forward and back substitution three times, for three right-hand sides. When these steps of computations are completed, the unambiguous information needed to create an animation for any number of frames consists of the selected radius of support, initial control points ($3N$ floats), spline coefficients for x, y and z ($3N$ floats), and the points order after sorting (N integers). We can also reorder points in model description and do not store the points order.

4.4. Evaluating the transformation

Computing the transformations is the most critical part in the sense of time optimizations for real-time applications. The algorithm shown in Figure 5 demonstrates the calculation of full transformation by CSRBFs. For every point, which is inside the radius of support, the distance is calculated once and after that space transformations are calculated according to a phase parameter (value varies from 0 to 1) that defines the total deformation. The deformation process is regarded as taking place step by step so that the transition from a known state to a new one takes place with small increments. That is, intermediate transformations for every step of animation are generated according to the phase parameter.

```
input_points_list — initial model points
output_points_list — points after deformation
spline_coefficientes_list — coefficients of spline
neighbors_ids_list — temporary list of integers

// for each point compute displacement
for i:=0 to input_points_list.length-1 do
begin
   // initial displacement
   dx := 0;
   dy := 0;
   dz := 0;
   // find all neighbours of current point
   // and put their indices into neighbors_ids_list
   // this can be done with help of octree
   neighbors_ids_list =
        FindNeighbors(input_points_list[i]);
   for j:=0 to neighbors_ids_list.length-1 do
   begin
      // calculate φ(||x⃗ − x⃗_i||)
      phi := Phi(input_points_list[neighbors_ids_list[j]],
           input_points_list[i]);
      // correct displacement in accordance with
      // influence of current neighbor;
      // spline coefficients could be calculated earlier
      dx := dx + SplineCoefficient[j].x*phi;
      dy := dy + SplineCoefficient[j].y*phi;
      dz := dz + SplineCoefficient[j].z*phi;
   end
   output_points_list[i].x := input_points_list[i].x + dx;
   output_points_list[i].y := input_points_list[i].y + dy;
   output_points_list[i].z := input_points_list[i].z + dz;
end
```

Figure 5: *Algorithm for computing shape transformation.*

5. Software framework

We developed the animation algorithm within a system consisting of a collection of C++ libraries that provide support for 3D modeling and interaction using open source library *"The Visualization Toolkit"* (VTK) [53]. The animation algorithm operates on the region of animation object selected by the user. Nevertheless, the opportunity to extend or localize the active (deformable) area allows us to decrease quite tedious process of selecting starting and destination points. Our C++ class for shape transformation can be used in the pipeline execution method (lazy mode) of VTK, that actually means that it can be combined with other shape transformation classes, including the other instances of the same class (for example, several CSRBFs shape transformations can be applied with different radii to the same object to calculate the object deformation according to its local coordinate system). The complete scene is represented as a collection of animation objects. In addition, we support movements of the object to provide:

- translation of the local coordinate system of an object along some trajectory (movement);
- change of orientation of the local coordinate system of the object (rotation); It is assumed that while applying transformation the center of gravity of the object is not moving.

The animation object can be constructed from several parts where one can apply:

- different independent transformations for various parts of an object, and
- different colors and textures for various parts of an object.

The created software system consists of two applications:

- *"Picker"* provides creation and translation of transformations. This program enables the user to input interactively control points and vectors for CSRBF spline calculation. The screenshot of the *"Picker"* level interface is shown in Figure 6.
- *"Animation composer"*. This interface enables the user with possibility to define trajectory interactively and to

provide orientation and timing marks (i.e., the schedule of the movement of an object along its trajectory). Kochanek-Bartels interpolating spline [54] is used for animating the movement and quaternion calculus [55] is used for animating the rotation. The screenshot of this interface is shown in Figure 7.

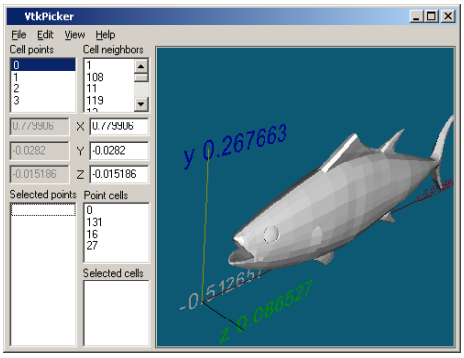

Figure 6: *The "Picker" interface.*

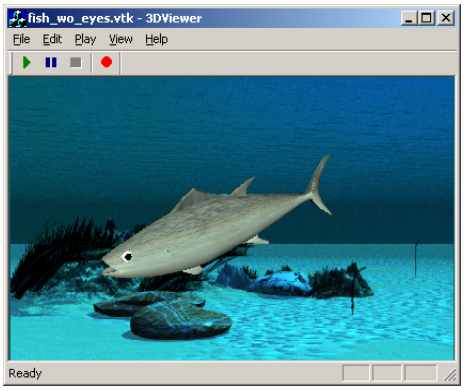

Figure 7: *The "Animation Composer" interface.*

The "Animation composer" program also supports an "Animation" mode. In this mode the program displays animated objects in accordance with their defined trajectories, rotations and transformations. This program also enables user to save animation results as a movie in personal computer movie file formats (full frames avi). Our interface level software was designed and tested for use on a PC under Windows (9x, ME, NT, 2000, XP). Command line interface version of the "Animation" program was also developed and tested on both Windows and Linux platforms.

6. Testing and benchmarking the animation software system

We used a polygonal model of the fish, defined by 975 vertices and 1119 polygons, to test our animation software system (desrcibed software and examples are available to download from our web page [57]). 92 control vectors were used to define the deformation of the mouth, tail and fin. Results of applying shape transformation for fish animation are shown in Figure 8. Two frames shown are selected from the generated animated images. The average speed of visualization and computation of transformations was about 632 frames/sec.

Another example of real-time transformations is shown in Figure 9. The sphere is bouncing inside a polygonal box. The box is deformed according to transformation vectors defined for each collision direction using C^2-continous $\psi_{3,1}(r) = (1-r)_+^4(4r+1)$ (see [47]) function. Such real-time deformations can be used to model elastic environment (i.e. walls, ceiling) in computer games. In spite of the fact that in this example we have to perform full computations for every frame, the real-time rendering speed (72 frames/sec.) was achieved.

Benchmark details for this examples are bellow in Figure 10. The screen resolution was 1024×768, 32 bpp.

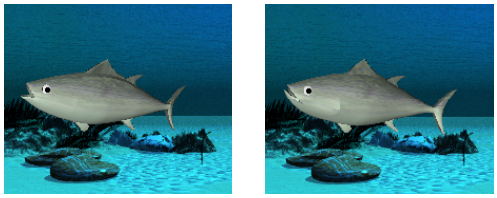

Figure 8: *Fish animation.*

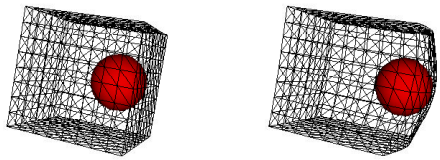

Figure 9: *Bouncing sphere.*

7. Conclusions

In this contribution we have presented an approach for surface modification based on CSRBFs. Experimental results demonstrate that our approach allows to generate plausible deformations, that is, they exhibit good blending features, and they produce what our intuition would expect. The algorithm was developed using C++ classes and is fully portable to the most of the modern computer systems. We have found that CSRBFs produces good results in visual appearance, processing time and memory requirements.

As our benchmarks (see Figure 10) show, the average number of animated frames can be up very high on modern personal computers. This means that the algorithm for

The model size, the number of vectors defining the deformation and the radius of support r	Rendering speed in frames per second (fps.)
Figure 1: The animation model is defined by 6158 vertices and 7024 polygons; the deformation is defined by 32 control vectors; the radius of support $r = 0.2$.	107 fps.
Figure 8: The animation model is defined by 975 vertices and 1119 polygons; the deformation is defined by 92 control vectors; the radius of support $r = 0.3$.	632 fps.
Figure 9: The box is defined by 386 vertices and 768 polygons, the number of control vectors and the radius of support are variable.	72 fps.

Figure 10: *Animation benchmarks on our test configuration: Athlon 1Ghz, 650MB RAM, ATI Radeon 250Mhz 8500LE 64MB video board VIA KT133 Chipset, Windows 2000 SP2, VTK 4.0.*

calculating the transformation is performed fast enough to be used in such real time applications as computer games. Also, this direction looks very attractive for various applications, especially ones which deals with the face animation (see Figure 1).

Selecting the set of control vectors to define the transformation is quite a difficult and time-consuming task. The opportunity to extend or localize the active (deformable) area by assigning various radii of support allows us to lighten the work of end user.

The future work will proceed in two directions. We are continuing to design an end-to-end user interface. We are planning to add other types of transformations such as key frame and inverse kinematics for animating a character to create an effective complete animation system. We also consider using the haptic visualization [56] to produce the desired transformations as a subject of future work.

References

1. H. Wendland. Piecewise polynomial, positive defined and compactly supported radial functions of minimal degree. *AICM*, **4**:389–396, 1995. 1, 2, 3
2. N. Kojekine, V. Savchenko, D. Berzin, I. Hagiwara. Software Tools for Compactly Supported Radial Basis Functions. *Computer Graphics and Imaging, Proc. IASTED, Hawaii, USA, August 13-16*, 234–239, 2001. 1, 2, 4
3. V. Savchenko and A. Pasko. Transformation of Functionally Defined Shapes by Extended Space Mappings. *The Visual Computer*, **14**:257–270, 1998. 2
4. T.W. Sederberg, S.R. Parry. Free-form deformation of solid geometric models. *Computer Graphics, 20*, **20**(4):151–160, 1986. 2
5. S. Coquillart. Extended free-form deformation: a sculpting tool for 3D geometric modeling. *Computer Graphics*, **24**(4):187–196, 1990. 2
6. S. Coquillart, P. Jancene. Animated free-form deformation: an interactive animation technique. *Computer Graphics*, **25**(4):23–26. 2
7. W.M. Hsu, G.F. Hughes and H. Kaufman. Direct manipulation of free-form deformations, *Computer Graphics*, **26**(2):177–184, 1992. 2
8. P. Borrel and D. Bechmann. Deformation of N-dimensional objects, *International Journal of Computational Geometry and Applications*, **1**(4):427–453, 1991. 2
9. P. Borrel and A. Rappoport. Simple constrained deformations for geometric modeling and interactive design, *ACM Transactions on Graphics*, **13**(2):137–155, 1994. 2
10. D. Bechmann. Space deformation models survey, *Computers & Graphics*, **18**(4):571–586, 1994. 2
11. J.H. Ahlberg, E.N. Nilson, J.L. Walsh. The Theory of Splines and Their Applications, *Academic Press, New York*, 1967. 2
12. J. Dushon. Splines Minimizing Rotation Invariants Semi-Norms in Sobolev Spaces, *Constructive Theory of Functions of Several Variables, W. Schempp, K. Zeller (Eds.), Springer-Verlag*, 85–100, 1976. 2
13. V.A. Vasilenko. Spline-functions: Theory, Algorithms, Programs, *Novosibirsk, Nauka Publishers*, 1983. 2
14. R.M. Bolle, B.C. Vemuri. On Three-Dimensional Surface Reconstruction Methods, *IEEE Transactions on Pattern Analysis and Machine Intelligence*, **13**(1):1–13, 1991. 2
15. G. Greiner. Surface Construction Based on Variational Principles, *Wavelets, Images and Surface Fitting, P. J. Laurent et al. (Eds), AL Peters Ltd.*, 277–286, 1994. 2
16. F.L. Bookstein. Principal Warps: Thin Plate Splines and the Decomposition of Deformations, *IEEE Transactions on Pattern Analysis and Machine Intelligence*, **11**(6), 567–585, 1989. 2
17. F.L. Bookstein, Morphometric Tools for Landmark Data, *Cambridge University Press*, 1991. 2
18. F.L. Bookstein. Two Shape Metrics for Biomedical

Outline Data: Bending Energy, Procrustes Distance, and The Biometrical Modeling of Shape Phenomena, *Proc. Shape Modeling Conference (SMIA'97), March 3–6, Aizu-Wakamatsu, Japan*, 110–120, 1997. 2, 3

19. P. Litwinovicz, L. Williams. Animating Images with Drawing, *Computer Graphics, Proc. SIGGRAPH*, 409–412, 1994. 2

20. V.V. Savchenko, A.A. Pasko, T.L. Kunii, and A.V. Savchenko. Feature based sculpting of functionally defined 3D geometric objects, *T.S. Chua et al. (Eds), Multimedia Modeling, Towards Information Superhighway, Proc. MMM, Nov.*, 341–348, 1995. 2

21. J.C. Carr, W.R. Fright and R.K. Beatson. Surface Interpolation with Radial Basis Functions for Medical Imaging, *IEEE Transaction on Medical Imaging*, **16**(1):96–107, 1997. 2

22. R.K. Beatson and W.A. Light. Fast Evaluation of Radial Basis Functions: Methods for 2D Polyharmonic Splines, *Tech. Rep. 119, Mathematics Department, Univ. of Canterbury, Christchurch, New Zealand*, Dec. 1994. 2

23. W. Light. Using Radial Functions on Compact Domains, *Wavelets, Images and Surface Fitting, P.J. Laurent et al. (Eds.), AL Peters Ltd.*, 351–370, 1994. 2

24. J.C. Carr, T.J. Mitchell, R.K. Beatson, J.B. Cherrie, W.R. Fright, B.C. McCallumm and T.R. Evans. Reconstruction and representation of 3D Objects with Radial Basis Functions, *Computer Graphics, Proc. SIGGRAPH*, 67–76, 2001 2

25. L. Greengard and V. Rokhlin. A Fast Algorithm for Particle Simulation, *J. Comput. Phys*, **73**:325–348, 1997. 2

26. V. Savchenko and L. Schmitt. Reconstructing Occlusal Surfaces of Teeth Using a Genetic Algorithm with Simulated Annealing Type Selection, *Proc. 6th ACM Symposium on Solid Modeling and Application, Sheraton Inn, Ann Arbor, Michigan, June 4–8*, 39–46, 2001. 2

27. B. Morse, T.S. Yoo, P. Rheingans, D.T. Chen, and K.R. Subramanian. Interpolating implicit surfaces from scattered surface data using compactly supported radial basis functions, *Shape Modeling conference, Proc. SMI, Genova, Italy, May*, 89–98, 2001. 2

28. S. Lee, G. Wolberg, and S.Y. Shin. Scattered Data Interpolation with Multilevel B-Splines, *IEEE Transaction on Visualization and Computer Graphics*, **3**(3):228–244, 1997. 2

29. J. Bloomenthal and C. Lim. Skeletal Methods of Shape Manipulation, *Proc. of International Conference on Shape Modeling and Applications, March 1-4, Aizu-Wakamatsu, Japan*, 44–47, 1997. 3

30. A. Verroust and F. Lazarus. Extracting Skeletal Curves from 3D Scattered Data, *Proc. of International Conference on Shape Modeling and Applications, March 1-4, Aizu-Wakamatsu, Japan*, 194-202, 1999. 3

31. J.P. Lewis, Matt Cordner, Nickson Fong, Pose Space Deformation: A Unified Approach to Shape Interpolation and Skeleton-Driven Deformation, *Proc. of SIGGRAPH*, 165-172, 2000. 3

32. D. Thalmann, J. Shen, and E. Chauvineau. Fast Realistic Human Body Deformations for Animation and VR Applications, *Computer Graphics International, Pohang, Korea*, 166–174, 1999. 2

33. P. Fua, R. Plankers, and D. Thalmann. From Synthesis to Analysis: Fitting Human Animation Models to Image Data, *Computer Graphics International, Canmore, Canada, June 7–11*, 4–11, 1999. 2

34. Y. Lee, D. Terzopoulos, and K. Waters. Realistic Modeling for Facial Animation, *Proc. SIGGRAPH, Los Angeles, CA, August*, 191–198, 1995. 2

35. http://www.biovirtual.com/ 2

36. M. Chen, A.E. Kaufman and R. Yagel (Eds.). Volume Graphics, *Springer*, 2000. 2

37. J.C. Platt and A.H. Barr. Constraint Methods for Flexible Models, *Computer Graphics*, **22**(4):279–278, 1988. 2

38. D. Terzopoulos, J.C. Platt, A.H. Barr and K. Fleisher. Elastically Deformable Models, *Computer Graphics*, **21**(4):205–214, 1987. 2

39. M. Aono. A Wrinkle Propogation Model for cloth, *Computer Graphics International*, 95–94, 1990. 2

40. T.L. Kunii and H. Gotoba. Singularity Theoretical Modeling and Animation of Garment Wrinkle Formation Processes, *The Visual Computer*, **6**:326–336, 1990. 2

41. B. Lafleur, M.N. Thalmann and D. Thalmann. Cloth Animation with Self-collision Detection, *Proc. IFIP Conference Modeling in Computer Graphics*, 179–187, 1991. 2

42. M. Hotton and S. Alexander, Soft Cellular Modeling: A Technique for the Simulation of Non-rigid Materials, *Computer Graphics: Development in Virtual Environments, R.A. Earnshav and J.A. Vince (eds), Academic Press*, 449–460, 1995. 2

43. L. Ling, M. Damodran and R.K.L. Gay. Physical Modeling for Animating Cloth Motion, *Computer Graphics: Development in Virtual Environments, R.A. Earnshav and J.A. Vince (eds), Academic Press*, 461–474, 1995. 2

44. S. Skaria, E. Akleman, F.I. Parke. Modeling Subdivision Control Meshes for creating Cartoon Faces,

Proc. Shape Modeling Conference (SMIA'01), May 7-11,Genova, Italy, 216–225, 2001. 2

45. G. Walberg. Skeleton based image warping, *Visual Computer*, **5**(1/2):95–108, 1989. 2
46. T. Beier, S. Neely. Feature-based image metamorphosis, *Computer Graphics*, **26**(2):35–42, 1992. 2
47. H. Wendland. On the smoothness of positive definite and radial functions, *Preprint submitted to Elsevier Preprint*, 1998. 3, 6
48. H. Samet. The Design and Analysis of Spatial Data Structures, *Addison-Wesley Pub Co.*, 1986. 4
49. A. Jennings. A Compact Storage Scheme for the Solution of Symmetric Linear Simultaneous Equations, *Comput. Journal*, **9**:281–285, 1966. 4
50. A. George and J.W.H. Liu. Computer Solution of Large Sparse Positive Definite Systems, *Prentice-Hall: Englewood Cliffs, NJ*, 1981. 5
51. W.H. Press, S.A. Teukolsky, T. Vetterling, B.P. Flannery. Numerical Recipes in C, *Cambridge University Press*, 1997. 5
52. Intel Math Kernel Library, *Reference Manual, Copyright 1994-2000, Intel Corporation.* 5
53. The Visualization Toolkit Textbook and open source C++ Library, with Tcl, Python, and Java bindings. *http://www.kitware.com/vtk.html, published by Kitware*, 2001. 5
54. D. Kochanek, R. Bartels. Interpolating Splines with Local Tension, Continuity, and Bias Control, *Computer Graphics*, **18**(3):33–41, 1984. 6
55. K. Shoemake. Animating rotation with quaternion calculus, *ACM SIGGRAPH course notes 10, Computer animation: 3D Motion, Specification and Control*, 1987. 6
56. SensAble Technologies. *GHOST Software Developer's Toolkit (SDK), Programmer's Guide, Version 1.21*, 1997. 7
57. http://www.karlson.ru/csrbf/ 6