Surface Reconstruction Based on Compactly Supported Radial Basis Functions

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

Vladimir Savchenko
Faculty of Computer and Information Sciences, Hosei University, 3-7-2 Kajino-cho Koganei-shi, Tokyo 184-8584, Japan.

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

In this chapter the use of compactly-supported radial basis functions for surface reconstruction is described. To solve the problem of reconstruction or volume data generation specially designed software is employed. Time performance of the algorithm is investigated. Thanks to the efficient octree algorithm used in this study, the resulting matrix is a band diagonal matrix that reduces computational costs.

1 Introduction

Traditionally, constructive solid geometry (CSG) modeling uses simple geometric objects for a base model that can be further manipulated by implementing a certain collection of operations such as set-theoretic operations, blending, or offsetting. The operations mentioned above and many others have found quite general descriptions or solutions for geometric solids represented as points (x,y,z) in space satisfying $f(x,y,z) \geq 0$ for a continuous function f. Such a representation is usually called an *implicit model* or a *function representation*. Set-theoretic solids have been successfully included in this type of representation with the application of R-functions and their modifications (see [1], and [2]). In [3], an approach to volume modeling is proposed. It combines the voxel representation and

function representation. It is supposed that the two main representations (voxel/function) are given rather autonomously and a rich set of operations can be used for modeling volumes. Whatever complex operations have been applied to a geometric object that can be given by a voxel raster, by elevation data, or by a function representation, equivalent volume data, can be generated. Many practical surface reconstruction techniques such as restoration design or reverse engineering tasks based on measured data points require the solution of optimization problems in fitting surface data. We use the term *volume model* to refer to the wide class of surfaces that like other implicit surfaces descriptions can be used for CSG.

A vast volume of literature is devoted to the subject of scattered data reconstruction and interpolation. In most applications, a Delaunay triangulation is used for 3D reconstruction. Unfortunately, this method has some serious drawbacks, even with the elimination of large triangles; the reconstructed shape remains convex-looking, as noted in [4]. Another approach to surface reconstruction is skeletal. An implicit surface generated by point skeletons may be fit to a set of surface points [5], but this method is rather time consuming. One of other approaches is to use methods of scattered data interpolation, based on the minimum-energy properties [6], [7], [8]. These methods are widely discussed in mathematical literature (see [9], [10]). The benefits of modeling 3D surfaces with the help of radial basis functions (RBF) have been recognized in [11] for Phobos reconstruction. They were adapted for computer animation [12], [13] and medical applications [14], [15] and were first applied to implicit surfaces by Savchenko et al. [16]. However, the required computational work is proportional to the number of grid nodes and the number of scattered data points. Special methods to reduce the processing time were developed for thin plate splines and discussed in [17], [18].

Actually, the methods exploiting the RBFs can be divided into three groups. First group is “naive” methods that are restricted to small problems, but they work quite well in applications, dealing with shape transformation (see, for example [19]). Second group is fast methods for fitting and evaluating RBFs that allow large data sets to be modeled [20], [17], [21]. The third and last group is compactly supported radial basis functions (CSRBFs) introduced by Wendland in [22]. The benefits of modeling 3D implicitly defined surfaces with the help of CSRBFS have been recognized in [23] where au-

thors have used a *k*-d tree [24] to build the resulting sparse matrix. To find the solution to the system of equations a direct LU sparse matrix solver [25] was used.

In practice, the problem of reconstruction consists of the following steps: sorting the data, constructing the system of linear algebraic equations (SLAE), solving the SLAE, and evaluating the functions. In fact, while the solution of the system is the limiting step, constructing the matrix and evaluating the functions to extract the isosurface may also be computationally expensive. In this contribution, we made an attempt to solve the problem according to the abovementioned steps. Thus, the main goal of the ongoing project was to develop an effective library of C++ classes that can be successfully applied to computationally intensive problems of surface reconstruction using RBFs splines.

2 Method of shape reconstruction with RBF splines

For a three-dimensional arbitrary area Ω, the thin–plate interpolation is the variational solution that defines a linear operator T when the following minimum condition is used:

$$\int_{\Omega} \sum_{|\alpha|=m} m!/\alpha!(D^{\alpha}f)^2 d\Omega \to \min, \tag{1}$$

where m is a parameter of the variational functional and α is a multi-index. It is equivalent to using the radial basis functions $\phi(r) = r^1$ or r^3 for m = 2 and 3, respectively, where r is the Euclidean distance between two points. Since the function $\phi(r)$ is not compactly supported, the corresponding system of linear algebraic equations (SLAE) is not sparse or bounded. Storing the lower triangle matrix requires $O(N^2)$ real numbers, and the computational complexity of a matrix factorization is $O(N^3)$. Thus, the amount of computation becomes significant, even for a moderate number of points N.

Wendland in [22] constructed a new class of positive definite and compactly supported radial functions for 1D, 3D, and 5D spaces of the form

$$\phi(r) = \begin{cases} \phi(r), 0 \le r < 1 \\ 0, r > 1 \end{cases} \tag{2}$$

whose radius of support is equal to 1. The function $\phi(r) = (1 - r)^2$, which is an interpolated function that supports only C^0 continuity, is used in our software. This function provides positive defined and nonsingular systems of equations. However, it is also possible to apply functions that support a higher continuity. An investigation [26] of the smoothness of this family of polynomial basis functions shows that each member $\phi(r)$ possesses an even number of continuous derivatives.

The volume spline $f(P)$ having values h_i at N points P_i is the function

$$f(P) = \sum_{j=1}^{N} \lambda_j \phi (|P - P_j|) + p(P), \quad (3)$$

where $p = \nu_0 + \nu_1 x + \nu_2 y + \nu_3 z$ is a degree one polynomial. To solve for the weights λ_j we have to satisfy the constraints h_i by substituting the right part of equation (3), which gives

$$h_i = \sum_{j=1}^{N} \lambda_j \phi (|P_i - P_j|) + p(P_i) \quad (4)$$

Solving for the weights λ_j and ν_0, ν_1, ν_2, ν_3 it follows that in the most common case there is a doubly bordered matrix $\boldsymbol{T}$ that consist of three blocks, square submatrices $\boldsymbol{A}$ and $\boldsymbol{D}$ of size $N \times N$ and $k \times k$, respectively, and $\boldsymbol{B}$, that is not necessarily square and has the size $N \times k$, where $k = 4$ for 3D space.

3 Algorithm for reconstruction

3.1 Sorting scattered data

Space recursive subdivision is an elegant and popular way of sorting scattered 3D data. We propose an efficient approach based on the use of variable-depth octal trees for space subdivision, that allows us to obtain the resulting submatrix $\boldsymbol{A}$ as a band diagonal matrix that reduces the computational complexity. Because of the structure of octal trees [27], for each node of the tree, we need to store the following:

- a pointer to the parent node,
- 8 pointers to child nodes,

- a pointer to the list of points (empty if this node is not a leaf).

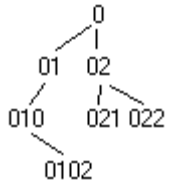
All memory needed to store such a tree is (memory for each node) × (number of nodes) + (memory for storing point) × N. In our software implementation, we are employing standard approach for creating the tree from an initial point data set (Algorithm I - "Octal tree creation") with an additional required parametric value K, which denotes the maximum number of points in the leaf. Afterwards we can use this tree to search for neighbors of any given point from the given N points. The neighbors are points belonging to a sphere of radius r with center at the given point. We call this sphere an r-sphere.

Algorithm II "Searching using the tree," is a "searching" function based on the fact, that for each node in the tree, we can state that this node is either:

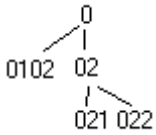
- entirely inside the given r-sphere,
- entirely outside the given r-sphere,
- partly inside the given r-sphere, partly outside it.

To accelerate the search function octal tree is simplified after creation. If node A has only one sub-node B, we can remove node A and replace it with B. This procedure is reasonable only if K is small. For example, consider the following octal tree (numerals represent node numbers):

Initial octal tree



Octal tree after simplification



For instance, for data from the "Head" example (see, Table 1 (a)) if K is set equal to 1 this procedure removes 113 nodes of the octal tree from the initially created 2427 nodes.

The maximum complexity of the two algorithms that have been described strongly depends on the initial data and the parameter K. The depth of the tree depends on the length of the cube edge corresponding to the leaf. This length is equal to $(1/2)^M$, where M is the depth of the tree that depends on the original data. If the initial points are distributed more or less uniformly, then the tree will have sufficiently uniform filling and will be symmetric. If K is set equal to 1, at that

time the tree will be close to a full octal tree with N leaves. The maximum complexity of searching the tree will be proportional to the depth of this tree, which is $\log_8 N$.

A more detailed account of these algorithms, including a C++ library description can be found in [28]. The procedure of searching for the neighbors of a point in a given r-sphere is applied several times in the application. For example, it is used for calculating the function (3) to sum up only the points that are neighbors of the specified point with coordinates (x,y,z) and thus accelerate the function calculation and surface extraction. But the first application is the construction of the band diagonal submatrix $\phi(|P - P_i|)$ that accounts for a significant portion of the computational cost. In our application, to store a band-diagonal matrix, we use the so-called profile form or a slightly modified version of the Jennings envelope scheme [29]. To store the submatrix $\boldsymbol{A}$, an array can be used for diagonal elements; values of non-diagonal elements and correspondent indices of the first non-zero elements in the submatrix lines are placed in two additional arrays.

To make our submatrix band diagonal, we need to re-enumerate the initial points in a special way. We propose the following Algorithm III "Sorting data using an octal tree":

- Take a point from the initial data and put it in the list.
- Go through the list, and for each point in the list search for the neighbors from the initial data. When new points are found add them to the list.
- Remove from the initial array points that have been placed in the list.
- If there are no more points in the list (all points were appended in the second step), then take the first point remaining in the initial array and repeat the above steps.

Algorithm III can be represented as pseudo-code:

```
input_list         - input list of points
output_list        - output list of points
neighbors_id_list - temporary list of ids

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 is done with octree
       neighbors_id_list=
   FindNeighbors(output_list[i],input_list);
       for j:=0 to neighbors_id_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_id_list[j]);
       end
   end
end
```

As a result of this algorithm a band with maximum size α_1 of the neighbors of a point is obtained. The maximum complexity of this algorithm is the complexity of searching for neighbors through the octal tree for each point, that is, $N \times$ (the maximum complexity of the algorithm II). We can reduce our computational outlays by calculating the matrix and the order of the points simultaneously.

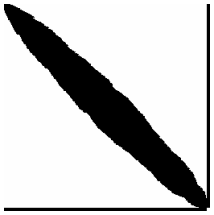


Fig. 1. Typical matrix with band-diagonal part created by algorithm III.

After sorting we get a band-diagonal submatrix as shown in Fig. 1. Note that the half-width for selected r cannot be decreased. Considering the following unlikely event would clarify this concept. If we connect all neighboring points we will obtain a graph, and if this graph has a cycle, then the maximum size we will get is less than or equal to the cycle length. Thus, if the radius of support is quite large, then the cycle will include nearly all the points from the input data. In this case, the maximum size of the band will also be large, and we will have an expansion of the band at some point.

3.2 SLAE solution

Note that solving any sparse system has the goals of saving time and space. The advantage of Gaussian **LU** [30] decomposition has been well recognized, and many software routines have been developed. For a symmetric and positive definite matrix, a special factorization, called Cholesky decomposition, is about twice faster compared to alternative methods for solving linear equations. From the discussion in section 3.1 it follows that in the most common case there is a doubly bordered band diagonal system $\boldsymbol{T}$ that consist of three blocks, square sub-matrices $\boldsymbol{A}$ and $\boldsymbol{D}$ of size $N \times N$ and $k \times k$ respectively, and $\boldsymbol{B}$ that is not necessarily square and has the size $N \times k$. A combination of block Gauss solution and Cholesky decomposition was proposed by George and Liu [31] and in our software tools we follow their proposal.

3.3 Surface evaluation and extraction

The surface fitted to a set of surface data point forms a volume model of a geometric object. This surface can be visualized directly by using an implicit ray-tracer, and can be voxelized, or polygonized

to extract a mesh of polygons. For the visualization of reconstructed volumes an implicit function modeler tool is used (see [32]).

Table 1. Processing time. Test configuration: *AMD Athlon 1000 Mhz, 128 MB RAM, Microsoft Windows 2000.*

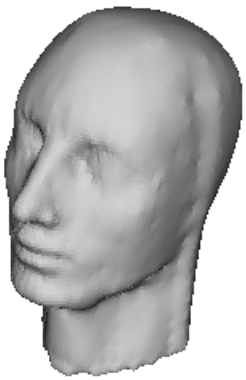
	"Head", Fig. 2 (a), number of points N = 1487, the selected radius of support r = 0.2	"Seashell", Fig. 2 (b), number of points N = 915, selected radius of support r = 0.2	"Venus", Fig. 2(c), number of points N = 6719, selected radius of support r = 0,13
Tree creation	0.001 sec.	0.001 sec.	0.01 sec.
Sorting time	0.03 sec.	0.03 sec.	0.26 sec.
Matrix calculation time	0.05 sec.	0.02 sec.	0.58 sec
Memory requirement to store the band diagonal submatrix of the matrix A	1,675,800 bytes (1 MiB) (if stored traditionally it would be 8856576 bytes (8 MiB))	669,068 bytes (0 MiB) (if stored traditionally it would be 3348900 bytes (3 MiB))	21171936 bytes (20 MiB) (if stored traditionally it would be 180579844 bytes (172 MiB))
Solution time with Cholesky decomposition	0.911 sec.	0.1 sec.	39.01 sec.
Polygonal surface extraction time	0.49 sec.	0.41 sec.	2.73 sec.

Note that the approach taken in this study does not guarantee a restoration of highly topologically complex volume objects. For 3D reconstruction using cross-sectional data, in [33] it is proposed that, for m different contours in one slice, m different function descriptions of separate contours must be used, and that union of the m car-

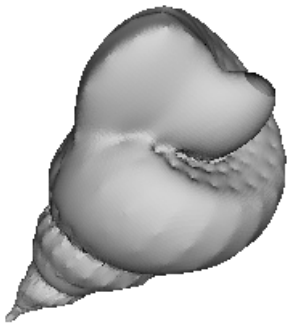
rier functions calculates the description of the reconstructed 2D object. For the 3D case such an approach looks exceedingly complicated. Moreover, RBFs demonstrate excessive blending features that lead to undesirable smoothing effects. The approach taken by Turk and O`Brien in [34] of using points specified on both sides of the surface will provide successful restoration of a surface, but it involves drawbacks that leads one to suppose that it would be inefficient for applying RBFs for volume reconstruction. Since it is out of the scope of this chapter to discuss all these matters, we would like to mention the following: this approach does not produce CSG-like solids as required in CSG (see [1]); the "both sides" approach has a problem with the surface extraction (a surface extractor can jump outside the band of non-zero points). There is also a problem of constructing or specifying off-surface points along a surface normal that also leads to a doubling of the given number of surface points.

4 Results

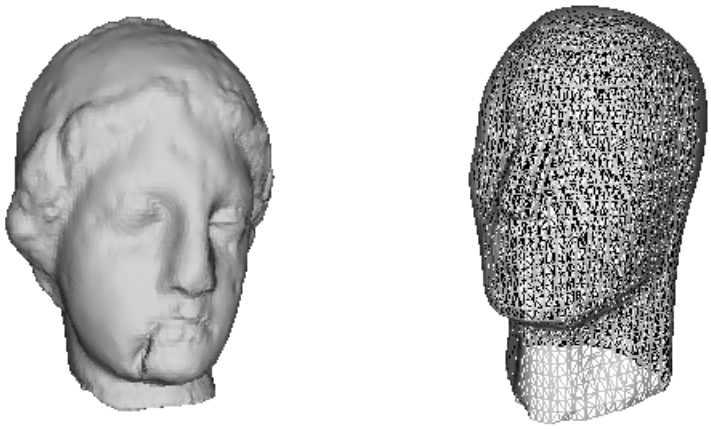
Visual inspection (images in Fig. 2) allows us to judge the interpolation features of the algorithm that has been discussed. Benchmark results can be seen in Table 1.



(a)



(b)



(c) (d)

Fig. 2. (a) “Head” reconstruction; (b) “Seashell reconstruction; (c) “Venus” reconstruction; (d) Wire-frame image of the extracted model “Head” (the number of polygons to be extracted is controlled by user-defined parameter in our software system).

The toolkit can be used in conjunction with other algorithms to create animated applications. For example in Fig. 3 the 3D application of combined mappings is illustrated. It is used to produce a visually smooth transformation between the initial union of two blended “torus” (genus 1) into the final “head” shape (genus 0) in the presence of obstacles. The series of frames shows the process of temporal metamorphosis and spatial transformations defined by the space mapping.

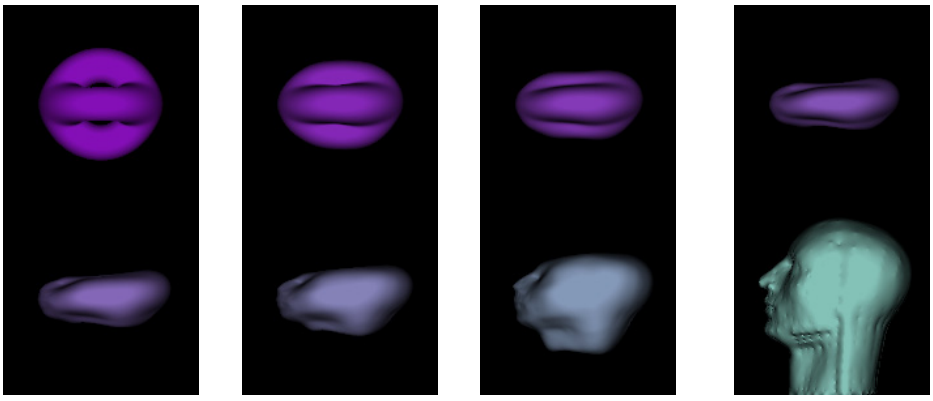


Fig. 3. Metamorphosis with constraints: spatial and temporal transformations.

5 Summary

In this contribution, the problem of using RBFs spline reconstruction for volume modeling was thoroughly investigated. The main goal of the ongoing project was to understand the processing characteristics and capabilities of the RBF reconstruction approach and its visualization aspects.

Thanks to the efficient octree algorithm, the resulting matrix of size $N+k*N+k$, where k=4 for 3D case, has a band diagonal submatrix (not a sparse one) of size $N*N$ that reduces greatly the computational complexity.

An online reconstruction server has been established on our web page [28] that makes it possible to get a visualization of a VRML object using a web-browser. This web page also provides our reconstruction software to download and several examples.

References

1. V. Shapiro, Real functions for representation of rigid solids, *Computer Aided Geometric Design*, 11(2), 153-175, 1994.
2. A. Pasko, V. Adzhiev, A. Sourin, V. Savchenko, Function representation in geometric modeling: concepts, implementation and applications, *The Visual Computer*, 11(6), 429-446. 1995.
3. V.V. Savchenko, A.A. Pasko, A.I. Sourin, T.L. Kunii, Volume Modeling: Representations and advanced operations, *Computer Graphics Int. Conf., F. Wolter and N.M. Patrikalakis (eds.)*, Hannover, Germany, IEEE Computer Society, 4-13, June 22-26 1998.
4. S. Djurcilov, A. Pang, Visualizing Sparse Gridded Data Sets, *IEEE Computer Graphics and Applications*, 52-57, Sept/Oct 2000.
5. S. Muraki, Volumetric Shape Description of Range Data Using "Blobby Model", *Computer Graphics* vol.25 (*Proceedings of SIGGRAPH*), no. 4, 227-235, 1991.
6. J.H. Ahlberg, E.N. Nilson, J. L. Walsh, The Theory of Splines and Their Applications, *Academic Press*, New York, 1967.
7. J. Dushon, *Splines Minimizing Rotation Invariants Semi-Norms in Sobolev Spaces*, *Constructive Theory of Functions of Several Variables,* W. Schempp and K. Zeller (eds.), Springer-Verlag, 85-100, 1976.

8. V.A. Vasilenko, *Spline-functions: Theory, Algorithms, Programs*, Novosibirsk, Nauka Publishers, 1983.
9. 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.
10. 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.
11. V. Savchenko, V. Vishnjakov, The Use of the "Serialization" Approach in the Design of Parallel Programs Exemplified by Problems in Applied Celestial Mechanics, *Performance Evaluation of Parallel Systems, Proceedings PEPS*, University of Warwick, Coventry, UK, 29-30 Nov., 126-133, 1993.
12. P. Litwinovicz, L. Williams, Animating Images with Drawing, Computer Graphics *(Proc. SIGGRAPH),* 409-412, 1994.
13. V. Savchenko, A. Pasko, T.L. Kunii, A.V. Savchenko, Feature Based Sculpting of Functionally Defined 3D Geometric Objects, *Proceedings Multimedia Modeling Conference*, Singapore, 14-17 Nov., T.T. Chua et al. (eds.), World Scientific Pub., 341-34, 1995.
14. 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.
15. 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.
16. V. Savchenko, A. Pasko, O. Okunev and T. Kunii, Function representation of solids reconstructed from scattered surface points and contours, *Computer Graphics Forum*, 14(4), 181-188, 1995.
17. R.K. Beatson and W.A. Light, Fast Evaluation of Radial Basis Functions: Methods for 2-D Polyharmonic Splines, Tech. Rep. 119, Mathematics Department, Univ. of Canterbury, Christchurch, New Zealand, Dec. 1994.
18. 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.
19. V. Savchenko, L. Schmitt, Reconstructing Occlusal Surfaces of Teeth Using a Genetic Algorithm with Simulated Annealing Type Selection, *6th ACM Symposium on Solid Modeling and Applications*, Sheraton Inn, Ann Arbor, Michigan, June 4-8, 2001.
20. L. Greengard and V. Rokhlin, A Fast Algorithm for Particle Simulation, *J. Comput. Phys.*, 73, 325-348, 1987.
21. 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, August 2001.
22. H. Wendland, Piecewise polynomial, positive defined and compactly supported radial functions of minimal degree, *AICM*, 4, 389-396, 1995.

23. 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, 89-98, May 2001.
24. J.L. Bently, Multidimensional binary search trees used for associative searching, CACM, 18(9), 509-517, 1975.
25. J.J. Dongarra, J.D. Croz, S. S. Hammarling, and I. Duff. a set of level 3 Basic Linear Algebra Subprograms, ACM Transactions on Mathematical Software, 16(1), 1-17, Mar. 1990.
26. H. Wendland, On the Smoothness of Positive Definite and Radial Functions *(Preprint submitted to Elseivier Preprint)*, 14 September 1998.
27. H. Samet, *The Design and Analysis of Spatial Data Structures*, Addison-Wesley Pub Co., 1986.
28. http://www.karlson.ru/csrbf/
29. A. Jennings, A Compact Storage Scheme for the Solution of Symmetric Linear Simultaneous Equations, *Comput. Journal* 9, 281-285, 1966.
30. W.H. Press, S.A. Teukolsky, T. Vetterling, B. P. Flannery, *Numerical Recipes in C*, Cambridge University Press, 1997.
31. A. George, J.W.H. Liu, *Computer Solution of Large Sparse Positive Definite Systems*, Prentice-Hall: Englewood Cliffs, NJ, 1981.
32. *The Visualization Toolkit Textbook and open source C++ Library, with Tcl, Python, and Java bindings*. http://public.kitware.com/VTK/, published by Kitware, 2001.
33. V. Savchenko, A. Pasko, Reconstruction from Contour Data and Sculpting 3D objects, *Journal of Computer Aided Surgery*, 1 Supl., 56-57, 1995.
34. G. Turk and J.F. O`Brien, Shape Transformation Using Variational Implicit Functions, *SIGGRAPH*, 335-342, 1999.