Point-based Modelling and Rendering using Radial Basis Functions

Patrick Reuter[*] Ireneusz Tobor[†] Christophe Schlick[‡] Sébastien Dedieu[§]

LaBRI - Université Bordeaux I - France

Abstract

A point-based 3D surface modelling technique combined with a new point rendering technique is presented. Surfaces are modelled by specifying a set of unorganized points on the surface. An implicit representation of the surface through these points minimizing the bending energy is then calculated using radial basis functions while guaranteeing a specifiable continuity. The surface is directly rendered view-dependently in an output-sensitive multiresolution manner without the creation of a polygonal mesh representation. This is done by the local generation of 3D surface points for the rendering adapted to the output device. A new texturing technique using the material properties of the points is presented.

CR Categories: G.1.2 [Approximation]: Approximation of surfaces and contours—Spline and piecewise polynomial approximation I.3.3 [Picture/Image Generation]: Display algorithms—[I.3.5]: Computational Geometry and Object Modeling—Curve, surface, solid, and object representations

Keywords: point-based rendering, point-based modelling, radial basis functions

1 Introduction

Modelling and rendering techniques implemented in almost every 3D graphics hardware are currently based on techniques developed as far back as the middle of the 1970's: surfaces are approximated by meshes of polygons and rendered using a depth buffer. Until recently, implementing such *mesh-based modelling* and *mesh-based rendering* techniques was obviously the optimal choice on a quality versus cost criterion according to existing computer resources. But the huge memory increase of modern graphics hardware has made it possible to treat scenes with dramatically more polygons, and during the rendering of such complex scenes many projected polygons are mapped to less than one screen pixel. So the main advantage of scan-line rendering, i.e. the incremental calculation of a polygon's inner points, has mostly vanished for current scenes of high complexity. Furthermore, representing surfaces by sets of planar polygons remains only an approximation and implicitly contains and therefore requires information about connectivity.

[*]e-mail: preuter@labri.fr
[†]e-mail:tobor@labri.fr
[‡]e-mail:schlick@labri.fr
[§]e-mail:dedieu@labri.fr

In the rendering field, a recent trend called *point-based rendering* has emerged for about four years now. In the modelling field, a similar trend, that could be called *point-based modelling*, can be detected in a couple of recent papers. The main idea that links all these papers is to use discrete surface points without explicit connectivity instead of using usual meshes. As the connectivity has not to be managed, some common operations (e.g. level-of-details, geometrical deformation, topology modifications) are much easier to implement with point-based models compared to mesh-based models. On the other hand, expensive *hole-filling* algorithms are required during the rendering of point-based models, in order to obtain a visually continuous surface rather than a cloud of disconnected points.

In this paper, we present a meshless graphics environment using points as modelling and rendering primitives. Surfaces are modelled by specifying a set of unorganized seed points on the surface. A continuous implicit representation of the surface through these points minimizing the bending energy is then calculated using radial basis functions. A desired continuity of the surface to be modelled can be specified. The surface can either be globally reconstructed by using thin-plate radial basis functions or be locally reconstructed by using compactly supported radial basis functions. The latter technique becomes necessary when the number of modelling points is high. Besides the specification of discrete surface points, the modelling of surfaces can also be influenced by specifying the normals of the surface points as well as their material properties using a new texturing technique.

Rendering the resulting reconstructed surface is also done in a meshless manner. In a first step, for every specified surface point, a sphere is associated so that the associated spheres of all surface points are overlapping. Then, the projection of the spheres on the screen are calculated, and for all inner pixels of the resulting discs the 3D surface points are identified by interpolating the implicit surface representation. These points are finally used for rendering. Multiresolution rendering is achieved by splatting 3D surface points over more than one pixel. Hence, the surface is rendered view-dependently in an output-sensitive manner without reconstructing a mesh representation.

The remainder of this paper is organized as follows: in the following section, we discuss previous work on point-based modelling, point-based rendering, and surface reconstruction. In section 3, we show how radial basis functions can be used to create an implicit representation of a surface starting from discrete surface points and normals. In section 4, we show how the surface can be rendered without creating a mesh representation. The basic operations of our point-based modelling environment are presented in section 5. Finally, section 6 presents experimental results, and in section 7 we conclude and propose some directions to future work.

2 Previous Work

2.1 Overview

There are basically three main problems to solve when trying to implement a complete point-based graphics environment. First, how to generate a continuous surface from an unorganized point set?

Second, how to efficiently render such a surface? Third, how to easily edit such a surface to get free-form modelling? Fortunately, many existing work can be used to solve, at least partly, these problems. This section presents some of this work.

2.2 Point-based Surface Reconstruction

Surface reconstruction from unorganized point sets is a major topic of computational geometry that has generated dozens of articles during the last decades. An exhaustive survey of the different solutions proposed in the literature is clearly out of the scope of this paper. The reader may refer to some classical monographes on computational geometry [de Berg et al. 1997; O'Rourke 1998]. Surface reconstruction techniques can be divided into two major categories: the reconstruction is done by employing either a parametric or an implicit surface.

We first recall recent work about parametric surface reconstruction. Hoppe et al. [1992] propose a piecewise surface estimation that uses a least square method to compute the local tangent plane at every seed point. The surface is then defined as the zero-set of the distance function to the nearest point's tangent plane. Amenta et al. [2001] presented a technique in which a median axis transformation is done on the unorganized point set using a Voronoi diagram. Levin [1999] proposed to approximate surfaces by defining a local reference domain for every input point approximating the tangent planes. The reference domains are used to compute a local polynomial approximation of the surface using a projection procedure based on his Moving Least Squares method [Levin 1998]. The resulting surface is C^{∞} continuous.

Concerning implicit surface reconstruction techniques, Savchenko et al. [1995] introduced a technique where the reconstructed surface is defined as a zero-set of a global implicit function. The process can be considered as an energy minimization technique, in which an initial solution, called carrier solution, is combined with a spline volume defined by Green's functions. The resulting surface is C^1 continuous. Later, Turk and O'Brien [1998; 2002] proposed a similar idea but using thin-plate radial basis functions instead of Green's functions. This offers a C^{∞} continuity for the resulting surface. Both techniques are limited to a few thousands of seed points. Carr et al. [2001] presented another reconstruction technique based on radial basis functions using a fast evaluation technique developed by Beatson et al. [1992; 1997].

Besides all these global reconstruction techniques for implicit surfaces, Morse et al. [2001] proposed to locally reconstruct an implicit surface by using the compactly supported radial basis functions defined by Wendland [1995]. This local method allows the treatment of a much higher number of input points.

2.3 Point-based Rendering

The first use of points as rendering primitive for solid objects can be attributed to Levoy and Whitted [1985]. Szeliski and Tonnesen [1992] adapted particle systems to surface models by introducing the concept of oriented particles. But it is only recently, that point rendering has grown interest in the computer graphics community. Using discrete points as rendering primitives can result into holes in the output image, especially for close-up views. There are two major approaches to address this so-called hole-filling problem.

In the first approach, holes are filled in image space. This was initially done by Grossman and Dally [1998] who proposed a push-pull algorithm that is unfortunately prone to blocky artifacts. Pfister et al. [2000] built on this work and added texture filtering and shading features. The texture filtering technique was further improved by Zwicker et al. [2001] based on a novel screen space formulation of an elliptical weighted average (EWA) filter, and moreover, an object space formulation of the EWA filter was derived enabling the use of modern graphics hardware to accelerate the rendering [Ren et al. 2002].

In the second approach, holes are filled in object space. Rusinkiewicz and Levoy [2000] presented the QSplat algorithm that generates a hierarchy of bounding spheres starting from the vertices of a polygonal mesh. The hierarchy traversal is adjusted in order to guarantee a given frame rate and each sphere is rendered as a circular splat resulting in blobby artifacts. Alexa et al. [2001] used Levin's surface interpolation method [Levin 1999] discussed above to resample point sets in object space in a preprocess for rendering at a given resolution. Kalaiah and Varshney [2001] capture the local differential geometry at each point sample and render the entire surface hardware-accelerated as a collection of local neighborhoods.

Besides these pure point-based rendering techniques, various approaches have been proposed recently combining point-based and polygon-based rendering in one framework using some hybrid hierarchy [Chen and Nguyen 2001; Cohen et al. 2001; Dey and Hudson 2002].

2.4 Point-based Surface Modelling

Point-based modelling, i.e. using a set of unorganized points as geometrical primitives for interactive modelling, is much less common than point-based rendering in computer graphics. To our knowledge, the only published work that can be included in this field was presented by Turk and O'Brien [1998; 2002], in which a thin-plate radial basis reconstruction has been combined with the particle sampling technique developed by Witkin and Heckbert [1994] for interactive visualization. But particles provide only a coarse information about the shape of the surface. To get a precise rendering Turk and O'Brien proposed to convert the implicit surface into a polygonal mesh, using a marching cubes algorithm [Lorensen and Cline 1987], for instance, or to generate a direct rendering of the surface by ray-tracing; both techniques are too slow to provide interactive framerates after surface changes. Turk and O'Brien also describe some basic low-level operations as moving a point, adding a point, or changing a normal, they can be used to get free-form modelling. As we will see in section 5, some high-level operators can be easily added in the editing tools library of a point-based modelling system which offers a powerful environment when being combined with the adaptive point-rendering technique described in section 4.

Recently, Pointshop 3D was presented by Zwicker et al. [2002], a system for interactive shape and appearance editing of 3D point-sampled geometry. It was initially designed as a generalization of 2D photo editing using 2D image pixels to 3D photography using discrete 3D surface elements. Pointshop is not considered as a point-based modelling system as changes of the surface geometry are limited to normal displacements and to moderate changes of the surface structure.

3 Radial Basis Functions

About 20 years ago, in an extensive survey, Franke [1982] identified radial basis functions as one of the most accurate and stable methods to solve the scattered data interpolation problem, but it is only very recently that radial basis functions have received an interest from the computer graphics community [Turk and O'Brien 1999; Carr et al. 2001; Morse et al. 2001]. In order to obtain an implicit surface representation using radial basis functions, a function f satisfying the equation

$$f(x_i) = h_i = 0, \ \ i = 1, ..., n \tag{1}$$

for all n seed points x_i has to be found. The common convention about implicit functions f is, that the surface is represented by the zero-set $f(x) = 0$ and has positive values inside and negative values outside the surface, see [Bloomenthal 1997]. In order to obey to the common convention of implicit surfaces, at least one constraint for an off-surface point has to be added, say

$$f(x_{n+j}) = h_{n+j}, \quad j = 1, ..., m \tag{2}$$

for m off-surface points being inside ($h_i > 0$) or outside ($h_i < 0$) the surface.

For the reconstruction of the surface, a suitable radial basis function $\phi : [0, \infty) \to \mathbb{R}$ has to be fixed, and with $k = n + m$ the interpolation function has the following form

$$f(x) = \sum_{i=1}^{k} w_i \phi(x - x_i) + P(x), \tag{3}$$

where $P(x)$ is a polynomial of first degree accounting for the linear and constant portions of f. Based on the equations (1), (2), and (3), the following linear system is built up in order to determine the weights w_i of the radial basis functions at the given seed points:

$$\begin{bmatrix} \phi_{11} & \phi_{12} & .. & \phi_{1k} & 1 & x_1^x & x_1^y & x_1^z \\ \phi_{21} & \phi_{22} & .. & \phi_{2k} & 1 & x_2^x & x_2^y & x_2^z \\ \vdots & \vdots & & \vdots & \vdots & \vdots & \vdots & \vdots \\ \phi_{k1} & \phi_{k2} & .. & \phi_{kk} & 1 & x_k^x & x_k^y & x_k^z \\ 1 & 1 & .. & 1 & 0 & 0 & 0 & 0 \\ x_1^x & x_2^x & .. & x_k^x & 0 & 0 & 0 & 0 \\ x_1^y & x_2^y & .. & x_k^y & 0 & 0 & 0 & 0 \\ x_1^z & x_2^z & .. & x_k^z & 0 & 0 & 0 & 0 \end{bmatrix} \begin{bmatrix} w_1 \\ w_2 \\ \vdots \\ w_k \\ 1 \\ p^x \\ p^y \\ p^z \end{bmatrix} = \begin{bmatrix} h_1 \\ h_2 \\ \vdots \\ h_k \\ 0 \\ 0 \\ 0 \\ 0 \end{bmatrix} \tag{4}$$

Here, $\phi_{ij} = \phi(\|x_i - x_j\|)$ where $\|.\|$ denotes the Euclidean distance. It can be seen, that the linear system is symmetric, and it can be made positive definite with the appropriate radial basis function. Duchon [1977] identified the following thin-plate radial basis function for three dimensions resulting in a C^∞ surface:

$$\phi(x) = \|x\|^3 \tag{5}$$

The linear system (4) can either be indirectly solved by a conjugate gradient method, or directly solved by an LU or single value decomposition. In practical, the number of constraints k is limited to a few thousands as the linear system is almost completely filled and the determination of the solution costs $O(k^3)$. However, Beatson et al. [Beatson and Newsam 1992; Beatson and Light 1997; Carr et al. 2001] describe a fast evaluation technique reducing the cost from $O(k^3)$ to $O(k^2)$ and hence enabling the use of a higher number of seed points.

Another way to allow a higher number of seed points is the use of compactly supported radial basis functions, i. e. $\phi(x) = 0$ for $\|x\| \geq r$, r being the support radius. Besides the functions of Wu [1995], Wendland [1995] constructed a new class of polynomial compactly supported radial basis functions of minimal degree for a given smoothness. Some of these functions are presented in Table 1.

$(1-\|x\|)_+^2$	C^0
$(1-\|x\|)_+^4(4\|x\|+1)$	C^2
$(1-\|x\|)_+^6(35\|x\|^2+18\|x\|+3)$	C^4
$(1-\|x\|)_+^8(32\|x\|^3+25\|x\|^2+8\|x\|+1)$	C^6

Table 1: Wendland's compactly supported radial basis functions for a given smoothness.

The support radius of these function is normalized to 1 but can easily be scaled to a support radius r by taking $\phi(\frac{\|x\|}{r})$. The support radius should be chosen with care: when the radius is too small, the local neighborhood of a seed point will not be captured correctly, whereas a radius that is too large will decrease the efficiency of the process. Note also that there has to be at least one off-surface point within the support radius of a seed point. According to Turk et al. [1998] and Morse et al. [2001], we suggest to create an off-surface point using the normals of the seed points.

By using compactly supported radial basis functions, the linear system at equation (4) becomes a sparse linear system as $\phi(\|x_i - x_j\|) = 0$ for all (x_i, x_j) having a greater distance than the support radius. Sparse linear systems can be efficiently stored and solved either directly or iteratively.

A 450 point model (Figure 4(a)) of the traditional Stanford bunny is used throughout this paper as a test model instead of the original 35,947 point model in oder to show that only few points are required to obtain good reconstruction results. The surface was reconstructed within a few seconds using Wendland's compactly supported radial basis functions for C^0 (Figure 4(b)), C^2 (Figure 4(c)), and C^4 continuity (Figure 4(d)). See section 6 for timing details. It can be seen that there is nearly no noticeable difference between the reconstruction of C^2 and C^4 smooth surfaces.

4 Point-based Surface Rendering

After the reconstruction of the implicit surface representation defined by the function f in equation (3), the surface can be rendered as explained in this section. Instead of traditionally rendering the implicit surfaces using a marching cubes algorithm [Lorensen and Cline 1987], a new meshless rendering algorithm is presented rendering the implicit surface view-dependently in an output-sensitive multiresolution manner.

First, inspired by the QSplat point rendering algorithm [Rusinkiewicz and Levoy 2000], for every specified surface point, spheres are centered around each surface point; the different radii are chosen so that the spheres at all seed points are just overlapping. This is done in order to guarantee, that the resulting discs from the projection of the spheres form a closed region, as can be seen in Figure 1.

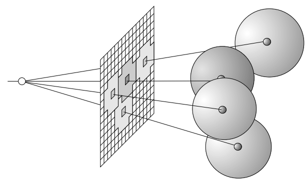

Figure 1: The projection of the four seed points forms a closed region.

Whereas the QSplat algorithm renders the surface only approximately by splatting the points making exclusive use of graphics hardware, our new algorithm renders the surface exactly by identifying additional 3D surface points in the local neighborhood of the seed points using a software implementation and by rendering all 3D surface points using graphics hardware. As a consequence, our point-based objects can be easily mixed with other geometrical primitives (polygons, splines) within the hardware rendering pipeline.

Consider the projection of the sphere to the output image in Figure 2. All inner pixels p_i of the disc are identified. The plane T^{near} (resp. T^{far}) is the nearest (resp. farthest) tangent plane of the sphere with respect to the viewpoint. For each pixel p_i the ray from the viewpoint to p_i intersects the planes T^{near} (resp. T^{far}) in the points q_i^{near} (resp. q_i^{far}).

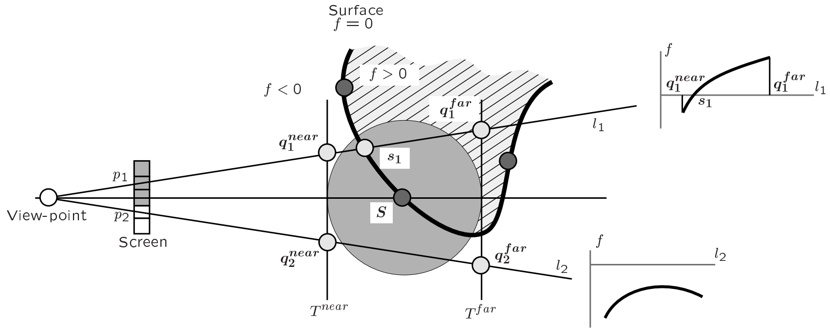

Figure 2: The projection of a sphere to the screen in 2D.

Then, the values of the implicit function f representing the surface in q_i^{near} and q_i^{far} are calculated. By supposing that the sphere is sufficiently small so that there is at most one intersection with the surface between q_i^{near} and q_i^{far}, we only have to consider the two following cases:

- If the values of the implicit function $f(q_i^{near})$ and $f(q_i^{far})$ are of different sign (see q_1^{near} and q_1^{far} in Figure 2), we calculate the surface intersection points s_i by interpolating between q_i^{near} and q_i^{far} using the regula falsi algorithm. Experiences have shown, that two iterations of the algorithm are enough to get a correct approximation. The resulting surface intersection point s_i will be projected to the pixel p_i and its normal can be analytically calculated using the gradient of the implicit function f,

$$\nabla f(s) = \left[\frac{\partial f}{\partial x}(s), \frac{\partial f}{\partial y}(s), \frac{\partial f}{\partial z}(s)\right]. \qquad (6)$$

- If $f(q_i^{near})$ and $f(q_i^{far})$ are of the same sign (see q_2^{near} and q_2^{far} in Figure 2), we consider that the ray is not intersecting the surface.

In this manner, all pixels p_i of the projected spheres are considered, and the additional surface points for rendering are generated view-dependently in an output sensitive manner. The additional surface points can even be generated in parallel as the spheres can be considered independently from each other.

Note the conceptual difference of the point-based rendering technique described here to standard acceleration techniques for ray casting: we apply forward warping in regions where the seed points are sufficiently dense and do not create holes in the output image. In the other regions, no ray needs to be cast in order to identify the sphere which is associated to the corresponding seed point as this is a prerequisite.

In order to associate textures to the reconstructed implicit surface, the interactive texture placement method by Pedersen [1995] could be easily adapted, or 3D solid textures could be used [Peachey 1985; Perlin 1985]. As our graphics environment uses points as modelling and rendering primitive, we present an alternative technique using the material properties of the seed points for texturing in the following. This technique does not require a surface parameterization.

To calculate the material properties for the additional surface points during rendering, we use an object space formulation of a spherical gaussian filter. This was motivated from a similar idea of a texture space formulation of the elliptical weighted average (EWA) filter introduced by Greene [1986] and a screen space formulation of the EWA filter by Zwicker et al. [2001]. The material property m_{new} of an additional surface point s_{new} is calculated by weighting the material properties m_i of the n seed points x_i with respect to the euclidean distance to the additional surface point as can be seen in equation (7). The euclidean distance is an approximation for the geodesic distance and is chosen for efficiency reasons. However, satisfactory results are obtained as we consider a small spacing between the seed points, the chord will approximate the geodesic appropriately.

$$m_{new} = \frac{\sum_{i=1}^{n} m_i e^{-\|s_{new}-x_i\|^2/R_i^2}}{\sum_{i=1}^{n} e^{-\|s_{new}-x_i\|^2/R_i^2}} \qquad (7)$$

The parameter R_i controls the influence region of a seed point's material property. This parameter can be adjusted according to local surface properties, anyway, we found satisfactory results using the same value for all R_i. For efficiently calculating (7), the values of the exponential function are precomputed and stored in a lookup table.

Multiresolution rendering is achieved by splatting additional generated 3D surface points over more than one pixel. Hence, for a splat covering n pixels, the number of additional surface points to calculate is reduced to a factor of $\frac{1}{n}$ resulting in a significant rendering speed-up.

Visual results of our new point-based rendering algorithm leading to high-quality shading effects and perfect silhouettes can be seen in a close-up view of our 450 point model of the Stanford bunny in Figure 4(e).

5 Surface Modelling

5.1 Overview

In this section, the wide variety of modelling possibilities of our algorithm are outlined and illustrated on the modelling of a simple sphere. Some of these modelling possibilities were already discussed by Turk and O'Brien [1998; 2002], but using our new point-based rendering algorithm the entire surface can be interactively modelled.

In our implementation, we always use the Wendland's radial basis function (Table 1) generating a C^2 continuous surface as we consider that it provides the best quality/speed trade-off. One off-surface point is created per seed point using the normal vector. Figure 3(a) presents a sphere reconstructed from a set of 42 seed points and 42 off-surface points defined by the corresponding normal vectors.

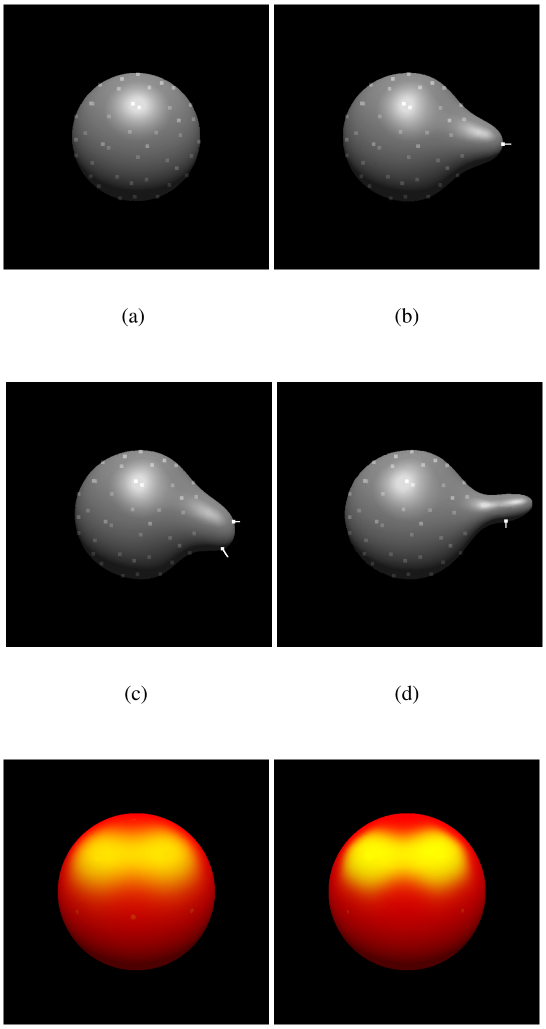

Figure 3: Modelling on a sphere. (a) Starting points, (b) moving a point, (c) adding a point, (d) changing a normal, (e)-(f) changing point materials.

5.2 Low-level Operators

There are basically four low-level operators that can be used to edit the shape of the surface, and one for editing the material.

Changing point positions: The position of a modelling point can be changed by simply moving it to its new position (Figure 3(b)). Changing the position of a point only affects one line and one column of the linear system (4), so the new surface (Figure 3(b)) can be recalculated in a very efficient manner using the previous solution of the linear system. This can be done either with a direct incremental solver such as the Sherman-Morrison technique [Press et al. 1992] or with an iterative solver like the conjugate gradient method that uses the previous solution as a starting point. Note that to avoid holes in the rendering of the new surface the radius of the corresponding bounding sphere has to be changed accordingly.

Changing point normals: The normal of any seed point can be changed as well, see Figure 3(c) for an example. This also changes only one line and one column of the linear system (4), and the new solution of the linear system is calculated efficiently as shown above.

Adding new points: New seed points can be added anywhere on the surface. The addition of new seed points on the surface adds two lines and two columns to the linear system, but the weights w_i of the precedent solution remain unchanged because the implicit function value of the surface was already 0 before the insertion of this new point. The position or normal of the new point can then be changed as shown above. The insertion and displacement of a new seed point and the resulting new surface can be seen in Figure 3(d).

Removing points: Seed points can also be removed, and the linear system can be recalculated efficiently similar to adding points.

Changing point materials: Finally, the material of a point can be changed. Either the material of a seed point is changed, or a new point with material properties is added on the surface. As the surface remains unchanged, the linear system does not have to be recalculated. A change of two point materials to a yellow colour using two different parameters for R can be seen in Figures 3(e) and 3(f).

5.3 High-level Operators

Besides the low-level operators described above that act on one single constraint, one can imagine more complex high-level operators that act simultaneously on a large number of constraints. During the last decades, several dozens of geometric modelling techniques that can be identified as *space deformation techniques* have been proposed. Their common principle is to define geometric deformation as a general transformation from $\mathbb{R}^3$ to $\mathbb{R}^3$. Among these techniques, one can cite for instance the warping operator by Barr, the free-form deformation operator by Sederberg and Parry, and others, see [Bechmann 1994] for a survey.

As our point-based primitives are totally defined by a set of 3D points, all these techniques can be easily included in our graphics environment. For instance, Figure 4(f) presents our test model on which a classical squeezing operator has been applied.

In our current implementation, all the editing operators are acting on the seed points, which gives a process similar to the edition of a spline surface by using control points. A valuable extension that we are currently investigating is to allow direct edition of the surface by adapting some techniques originally developed to provide direct manipulation of spline surfaces [Bartels and Beatty 1989; Welch and Witkin 1992; Gleicher 1994].

6 Results

All the timings presented in this section were measured on a Pentium IV at 1.7 GHz with 512 MB of memory, no code optimization effort has been done.

The main computational effort of the initial surface reconstruction using our example of Wendland's C^2 compactly supported radial basis functions is spent on the solution of the linear system (equation 4). We used an iterative solver [Dongarra et al. 1996] for the timings shown in table 6 as the use of direct solvers limits the number of seed points and is not adapted to solve sparse linear systems. For compactly supported radial basis functions, the parameter l indicates the average number of seed points to be taken into consideration per point, a lower l resulting in a sparser linear system and faster processing times.

Model	l (points)	reconstruction (sec.)	rendering (sec.)
Bunny 450	~ 36	1.8	2.1
	~ 69	4.4	4.2
	~ 112	6.8	6.9
Dragon 2832	~ 11	0.7	3.1
	~ 31	10.5	6.6
	~ 65	20.6	13.0

Table 2: Timings in sec. for reconstruction and rendering of one frame using one processor.

Table 6 also shows the timings for rendering the surface by a single processor into a 256×256 window. As mentioned above, the rendering part can be perfectly distributed on multiple processors, and thanks to the scalability of the algorithm, we expect a linear speed-up of rendering times using up to a few dozens of processors.

Figure 5 illustrates several steps of the modelling process using low-level and high-level operators.

7 Conclusion and Future Work

In this paper we described a complete meshless graphics environment using sets of unorganized points as modelling and rendering primitives. The strengths of this environment are

- the ability to generate global or local reconstruction of surfaces with specifiable continuity,
- the point-based rendering technique that provides view-dependent multiresolution and high-quality display of smooth surfaces in an output sensitive manner,
- the point-based texturing technique not requiring a surface parameterization,
- interactive point-based modelling of smooth surfaces that offers both low-level and high-level editing operators, and
- the merging of all these techniques into one point-based, meshless graphics environment.

Describing surfaces by the use of radial basis functions is very lightweight in terms of memory consumption as only the set of seed points together with the radial basis function and the calculated weights suffice to represent the surface. This is very interesting for geometry compression as well as bandwith-limited network applications.

Using radial basis functions in order to reconstruct implicit surfaces inherently assumes that the surface is everywhere locally symmetric, making it impossible to preserve sharp features along edges and corner points. One elegant solution to remove this limitation may be to use anisotropic basis functions [Dinh et al. 2001].

Finally, as the bottleneck of the rendering time is the evaluation of the radial basis functions, we are currently working on a fast evaluation method for radial basis functions based on a polynomial approximation.

References

ALEXA, M., BEHR, J., COHEN-OR, D., LEVIN, D., FLEISHMAN, S., AND SILVA, C. T. 2001. Point set surfaces. In *IEEE Visualization 2001*, IEEE, 21–28.

AMENTA, N., CHOI, S., AND KOLLURI, R. K. 2001. The power crust. In *Proceedings of the sixth ACM symposium on Solid modeling and applications*, ACM Press, 249–266.

BARTELS, R., AND BEATTY, J. 1989. A technique for the direct manipulation of spline curves. In *Proceedings of Graphics Interface '89*, Morgan Kaufmann Publishers, 33–39.

BEATSON, R. K., AND LIGHT, W. A. 1997. Fast evaluation of radial basis functions: methods for two-dimensional polyharmonic splines. *IMA Journal of Numerical Analysis 17*, 3 (July), 343–372.

BEATSON, R., AND NEWSAM, G. 1992. Fast evaluation of radial basis functions. *Computational Mathematics and Applications 24*, 12, 7–20.

BECHMANN, D. 1994. Space deformation models survey. *Computers & Graphics 18*, 4, 571–586.

BLOOMENTHAL, J., Ed. 1997. *Introduction to Implicit Surfaces*, vol. 391. Morgan-Kaufmann.

CARR, J. C., BEATSON, R. K., CHERRIE, J. B., MITCHELL, T. J., FRIGHT, W. R., MCCALLUM, B. C., AND EVANS, T. R. 2001. Reconstruction and representation of 3d objects with radial basis functions. In *Proceedings of ACM SIGGRAPH 2001*, ACM, 67–76.

CHEN, B., AND NGUYEN, M. X. 2001. Pop: a hybrid point and polygon rendering system for large data. In *IEEE Visualization 2001*, IEEE, 45–52.

COHEN, J. D., ALIAGA, D. G., , AND ZHANG, W. 2001. Hybrid simplification: Combining multi-resolution polygon and point rendering. In *IEEE Visualization 2001*, IEEE, 37–44.

DE BERG, M., VAN KREVELD, M., OVERMARS, M., AND SCHWARZKOPF, O. 1997. *Computational Geometry: Algorithms and Applications*. Springer-Verlag, Berlin.

DEY, T. K., AND HUDSON, J. 2002. PMR: Point to mesh rendering, a feature-based approach. In *IEEE Visualization 2002*, IEEE.

DINH, H. Q., SLABAUGH, G., AND TURK, G. 2001. Reconstructing surfaces using anisotropic basis functions. In *Proceedings of International Conference on Computer Vision (ICCV) 2001*, 606–613.

DONGARRA, J., LUMSDAINE, A., POZO, R., AND REMINGTON, K., 1996. Iml++ v. 1.2: Iterative methods library reference guide. Available at http://math.nist.gov/lapack++/.

DUCHON, J. 1977. Spline minimizing rotation-invariant semi-norms in sobolev spaces. In *Constructive Theory of Functions of Several Variables*, W. Schempp and K. Zeller, Eds., vol. 571, 85–100.

FRANKE, R. 1982. Scattered data interpolation: Tests of some methods. *Mathematics of Computation 38*, 157, 181–200.

GLEICHER, M. 1994. *A Differential Approach to Graphical Interaction*. PhD thesis, School of Computer Science, Carnegie Mellon University.

GREENE, N., AND HECKBERT, P. S. 1986. Creating raster omnimax images from multiple perspective views using the elliptical weighted average filter. *IEEE Computer Graphics and Applications 6*, 6, 21–27.

GROSSMAN, J. P., AND DALLY, W. J. 1998. Point sample rendering. In *Rendering Techniques '98*, Springer Verlag, Eurographics, 181–192.

HOPPE, H., DEROSE, T., DUCHAMP, T., MCDONALD, J., AND STUETZLE, W. 1992. Surface reconstruction from unorganized points. In *Computer Graphics (Proceedings of ACM SIGGRAPH 92)*, vol. 26, ACM, 71–78.

KALAIAH, A., AND VARSHNEY, A. 2001. Differential point rendering. In *Rendering Techniques 2001*, Springer Verlag, Eurographics, 139–150.

LEVIN, D. 1998. The approximation power of moving least-squares. *Mathematics of Computation 67*, 224, 1517–1531.

LEVIN, D. 1999. Mesh-independent surface interpolation. In *4th International Conference on Curves and Surfaces*, 46.

LEVOY, M., AND WHITTED, T. 1985. The use of points as display primitive. Tech. Rep. TR 85–022, University of North Carolina at Chapel Hill.

LORENSEN, W. E., AND CLINE, H. E. 1987. Marching cubes: A high resolution 3D surface construction algorithm. In *Computer Graphics (ACM SIGGRAPH 87 Proceedings)*, vol. 21, ACM, 163–169.

MORSE, B. S., YOO, T. S., RHEINGANS, P., CHEN, D. T., AND SUBRAMANIAN, K. R. 2001. Interpolating implicit surfaces from scattered surface data using compactly supported radial basis functions. *Proceedings of the International Conference on Shape Modeling and Applications*, 89–98.

O'ROURKE, J. 1998. *Computational Geometry in C*, 2nd ed. Cambridge University Press.

PEACHEY, D. R. 1985. Solid texturing of complex surfaces. In *Computer Graphics (Proceedings of ACM SIGGRAPH 85)*, vol. 19, ACM, 279–286.

PEDERSEN, H. K. 1995. Decorating implicit surfaces. In *Proceedings of ACM SIGGRAPH 95*, ACM, 291–300.

PERLIN, K. 1985. An image synthesizer. In *Computer Graphics (Proceedings of ACM SIGGRAPH 85)*, vol. 19, ACM, 287–296.

PFISTER, H., ZWICKER, M., VAN BAAR, J., AND GROSS, M. 2000. Surfels: Surface elements as rendering primitives. In *Proceedings of ACM SIGGRAPH 2000*, ACM, 335–342.

PRESS, W. H., TEUKOLSKY, S. A., VETTERLING, W. T., AND FLANNERY, B. P. 1992. *Numerical Recipes in C: The Art of Scientific Computing*, 2nd ed. Cambridge University Press.

REN, L., PFISTER, H., AND ZWICKER, M. 2002. Object space ewa surface splatting: A hardware accelerated approach to high quality point rendering. *Computer Graphics Forum (Eurographics 2002) 21*, 3, 461–470.

RUSINKIEWICZ, S., AND LEVOY, M. 2000. Qsplat: A multiresolution point rendering system for large meshes. In *Proceedings of ACM SIGGRAPH 2000*, ACM, 343–352.

SAVCHENKO, V. V., PASKO, A. A., OKUNEV, O. G., AND KUNII, T. L. 1995. Function representation of solids reconstructed from scattered surface points and contours. *Computer Graphics Forum 14*, 4 (October), 181–188.

SZELISKI, R., AND TONNESEN, D. 1992. Surface modeling with oriented particle systems. In *Computer Graphics (Proceedings of ACM SIGGRAPH 92)*, vol. 26, ACM, 185–194.

TURK, G., AND O'BRIEN, J. 1998. Variational implicit surfaces. Tech. Rep. GIT-GVU-99-15, Georgia Institute of Technology.

TURK, G., AND O'BRIEN, J. 1999. Shape transformation using variational implicit functions. In *Proceedings of ACM SIGGRAPH 99*, ACM, 335–342.

TURK, G., AND O'BRIEN, J. F. 2002. Modelling with implicit surfaces that interpolate. *ACM Transactions on Graphics 21*, 4, 855–873.

WELCH, W., AND WITKIN, A. 1992. Variational surface modeling. In *Computer Graphics (Proceedings of ACM SIGGRAPH 92)*, vol. 26, ACM, 157–166.

WENDLAND, H. 1995. Piecewise polynomial, positive definite and compactly supported radial functions of minimal degree. *Advances in Computational Mathematics 4*, 389–396.

WITKIN, A. P., AND HECKBERT, P. S. 1994. Using particles to sample and control implicit surfaces. In *Proceedings of ACM SIGGRAPH 94*, ACM, 269–278.

WU, Z. 1995. Compactly supported positive definite radial functions. *Advances in Computational Mathematics 4*, 283–292.

ZWICKER, M., PFISTER, H., VAN BAAR, J., AND GROSS, M. 2001. Surface splatting. In *Proceedings of ACM SIGGRAPH 2001*, ACM, 371–378.

ZWICKER, M., PAULY, M., KNOLL, O., AND GROSS, M. 2002. Pointshop 3d: An interactive system for point-based surface editing. *ACM Transactions on Graphics 21*, 3 (July), 322–329.

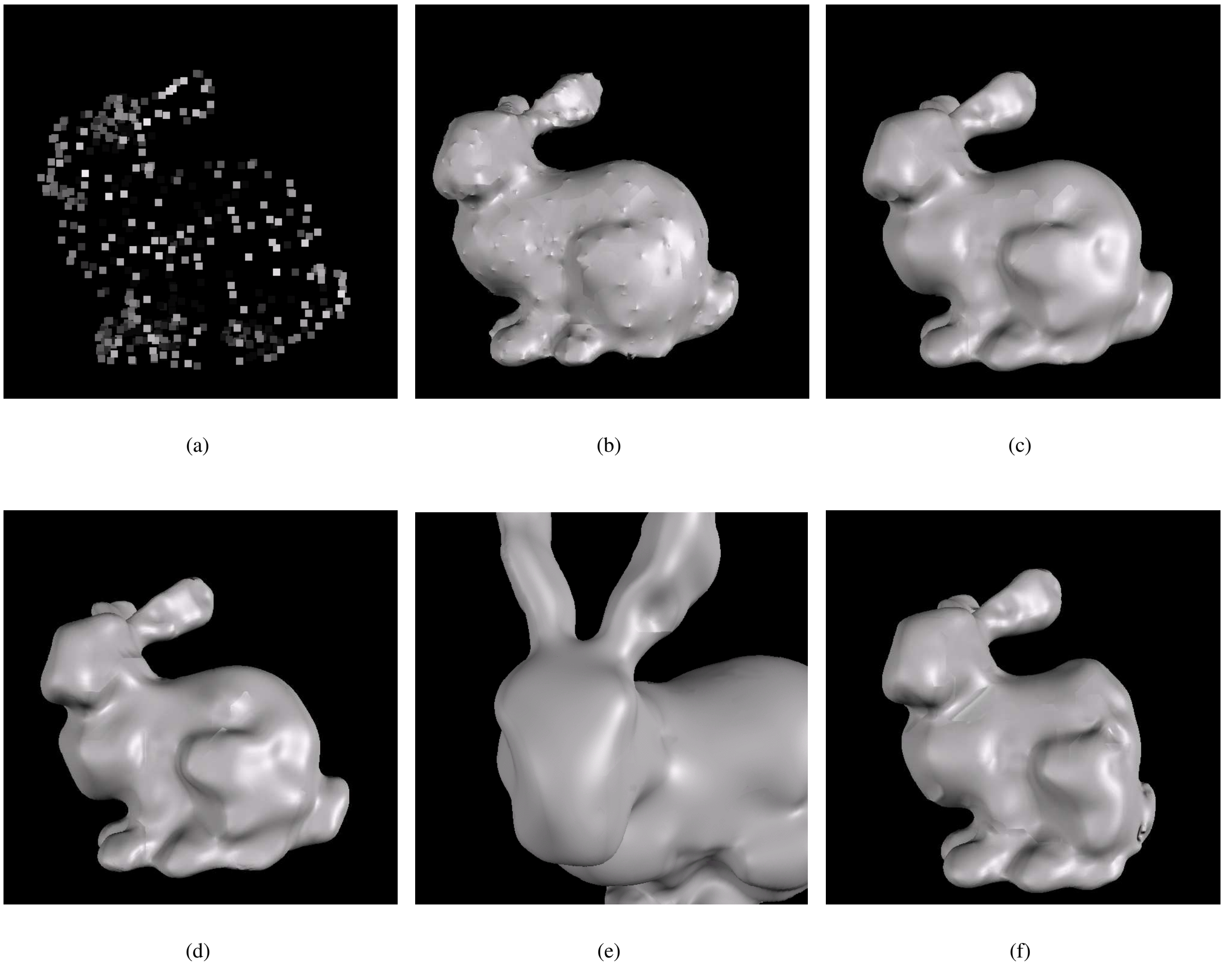

Figure 4: Rendering of the bunny. (a) A set of 450 seed points, (b) C^0 surface, (c) C^2 surface, (d) C^4 surface, (e) close-up view of a C^2 surface, (f) the surface squeezed in the x-axis.

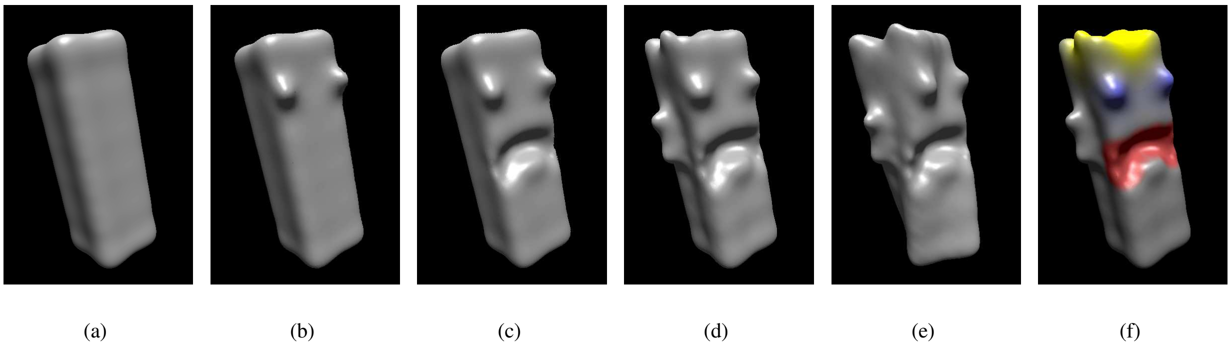

Figure 5: Modelling example. (a) Base surface reconstructed from 224 points, (b)-(c) changing point positions, (d) adding 9 new points, (e) application of a twist operator, (f) changing 9 point material properties.