Three Dimensional Freeform Sculpting Via Zero Sets of Scalar Trivariate Functions *

Alon Raviv and Gershon Elber
Department of Computer Science
Technion, Israel Institute of Technology
Haifa 32000, Israel
E-mail: {alonr | gershon}@cs.technion.ac.il

Abstract

This paper presents a three dimensional interactive sculpting paradigm that employs a forest of scalar uniform trivariate Bspline functions.

The sculpted object is evaluated as the zero set of the forest of trivariate functions defined over a three dimensional working space, resulting in multi-resolution control capabilities. The continuity of the sculpted object is governed by the continuity of the trivariates. The manipulation of the objects is conducted by modifying the scalar control coefficients of the meshes of the participating trivariates. Real time visualization is achieved by incrementally computing a polygonal approximation via the *Marching Cubes* algorithm. The exploitation of trivariates in this context benefits from the different properties of the Bspline's representation such as subdivision, refinement and convex hull containment.

A system developed using the presented approach has been used in various modeling applications including reverse engineering.

1 Introduction

Direct interactive manipulation of three dimensional geometry has been introduced in numerous applications. For example, the approach of Coquillart [2] and Jeng and Xiang

*This work was partially supported by the Fund for Promotion of Research at the Technion, Haifa, Israel.

[7] to this problem included interactive deformation by moving lattice points that hold a deformed object or patches that construct an object. Sculpting through physically based modeling is another example. In this approach, an intuitive way of applying physically based constraints, like forces, is used for modeling. For example, such an approach has been employed by Pentland et. al. [11]. The operations that are typically employed in modeling are quite different from those used by sculptors, and hence less intuitive as well as less appealing to the art community.

The approach of sculpting by carving or adding material has been dealt with, in several studies. Galyean and Hughes [5] and Wang and Kaufman [13] developed environments in which the user can freely sketch a three dimensional model, just like in paint packages of two dimensional images, an approach that is closer to traditional sculpting. The actual modeling is conducted using a three dimensional input device in [5] or a two dimensional input device in [13], which controls the position of the three dimensional sculpting tool. This tool can either remove material from or add material to the object. The object is described via a discrete *characteristic function*, $\mathcal{F}(x, y, z)$, that is essentially an implicit form, whose values are positive at any point in which the material exists and negative elsewhere. The tool is also described via a similar characteristic function, $\mathcal{T}(x, y, z)$, which modifies the value of $\mathcal{F}$ in the volume it occupies. Both the object and the tool are defined over *voxmaps*. A *voxmap* is a three dimensional grid of uniform voxels. The discrete values of the function are assigned to the vertices of the voxels. An example of a two dimensional *voxmap* can be seen in Figure 1.

The *characteristic functions* in [5, 13] are C^0 continuous as the representation is piecewise linear. Hence, when the tool updates the object, the object is modified in a discontinuous manner instead of a smooth one. In order to alleviate the effects of aliasing, both *characteristic functions* of the object and of the tool undergo pre-filtering in [5, 13] and are assigned continuous values between zero and one at the vertices of the voxels. In both [5] and [13], the resolution of the three dimensional grid is fixed, making sculpting at different levels of details difficult.

Galyean and Hughes [5] visualize the object using a polygonal model created with the aid of the local *Marching Cubes* algorithm [8]. This means that only the voxels modified

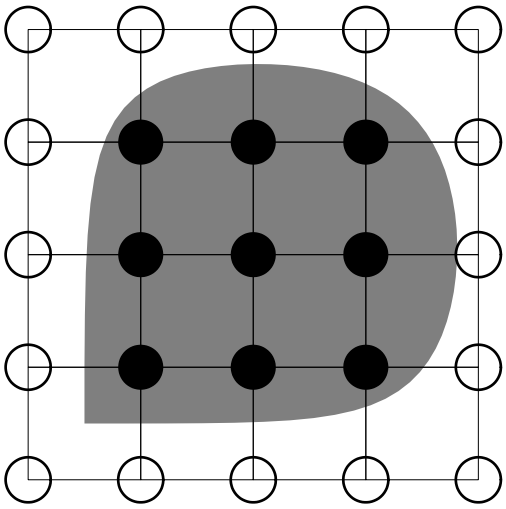


Figure 1: A two dimensional voxmap. This voxmap defines a binary function, where the black solid circles denote locations in which material exists and the white empty circles denote locations in which there is no material.

by the tool are re-evaluated each time the tool operates. Rendering is also performed locally. Since the voxels that are modified by the tool are known, only the regions in the screen that are covered by the projections of the modified voxels are re-rendered using a dedicated data structure. Wang and Kaufman [13] employ a local ray casting algorithm of their own, for the display. In both cases, the rendering of the object is view point dependent, and significant computations must be carried out when the view point is changing.

Sculpting through material carving or adding has also been considered by Mizuno, Okada and Toriwaki [10]. They propose an interactive modeling technique to form solid objects that emulate realistic wooden sculptures. The user can carve the surface of the object without penetrating it, just like a sculptor using a chisel. To hold the solid objects and carving tools, [10] uses a CSG representation. The sculpting is conducted with the aid of rendering that employs ray casting for every pixel in the screen against the object and the tool. As a consequence, sculpting is done in a fixed uniform resolution (the resolution of the screen) and is view point dependent. All the lists must be recomputed when the view point is modified.

One of the first attempts at modeling in various resolutions has been considered by Forsey and Bartels in [3]. In [3], hierarchical *overlays* are used to perform local refinement using vector Bspline surfaces. An *overlay* is a patch of control points at any desired resolution which is used to refine a local region on the surface. The *overlay's* resolution is determined by it's hierarchical level. The higher the level, the finer the resolution. Since the user is not limited by the number of levels, he/she can always refine the surface at any given resolution.

For realistic modeling, changes to the *overlays* in lower levels must be reflected and displayed as global changes to the *overlays* that exist in the levels above. Since vector surfaces are used in [3], it is not trivial to compute the effects on *overlays* in higher levels which are the result of changes in the levels below. Thus, special purpose computations and data structures are required.

This paper presents a three dimensional multi-resolution sculpting scheme using zero sets of scalar uniform trivariate Bspline functions. The use of trivariate Bsplines allows arbitrary continuity and robust evaluation, and permits the exploitation of the unique properties of the representations such as subdivision and convex hull containment. Furthermore and in contrast with the discrete approaches, the existence of a closed form and continuous formulation of the form of $\mathcal{F}(x, y, z) = 0$, allows one to completely eliminate the ambiguities that are introduced by the *Marching Cubes* algorithm, an algorithm that is used to visualize the sculpted objects. Section 2 considers the basic mathematical representation which is used to hold the sculpted objects. Section 3 describes the necessary building blocks of such a system and it's main data structures. Section 4 portrays some examples of sculpted objects using the presented approach, and finally, Section 5 concludes and discusses future work.

2 Background

Before presenting our modeling approach, it seems appropriate to briefly consider the kind of representation that is being used. The sculpted objects are represented as a zero set of a forest of scalar trivariate Bspline functions of the form $w = \mathcal{F}(x, y, z)$.

Just like an (explicit) scalar bivariate surface that assigns a scalar z_0 value to every point (x_0, y_0) in the XY plane, an (explicit) scalar trivariate function assigns a scalar, w_0, value to every point, (x_0, y_0, z_0), in the three dimensional space. Unfortunately, trivariate functions can not be easily visualized since these functions are embedded in a four dimensional space. One way of visualizing such functions is to assign a color and/or transparency factors to every (x, y, z) location following the trivariate function's value, and then integrate this information, possibly via ray casting.

Another way of visualizing implicit functions is by contouring or extracting a constant, w_0, set out of the implicit form, $w_0 = \mathcal{F}(x, y, z)$, resulting in a bivariate function in a three dimensional space. We refer to this constant set as *constant iso-surface*.

In this work, we use scalar tensor product uniform trivariate Bspline functions [12] for the object representation. Like all tensor product Bspline functions, these trivariate functions have a control-volume that consists of scalar coefficients, $P_{ijk} \in \mathbb{R}$. These trivariate functions are of the form:

$$q(u, v, w) = \sum_{i=0}^{l-1} \sum_{j=0}^{m-1} \sum_{k=0}^{n-1} P_{ijk} B_i(u) B_j(v) B_k(w), \tag{1}$$

where $B_i(u), B_j(v), B_k(w)$ are the uniform Bspline basis functions, P_{ijk} are the scalar coefficients in a volumetric mesh of size $l \times m \times n$, and $q(u, v, w)$ is a scalar function.

Sculpting is conducted by modifying the values of the scalar coefficients of the implicit trivariate forms. Arbitrarily accurate constant iso-surfaces are approximated by adaptive sampling of the exact freeform trivariate function, at some prescribed resolution, and by computing a polygonal approximation model of the sculpted shape via the *Marching Cubes* algorithm. This, much like freeform Bspline surfaces that are approximated using polygonal data, for display purposes.

3 System Description

3.1 Overview

Sculpting is allowed only within the boundaries of the trivariate volumetric patches. A patch is a volume prescribed by a bounding box in which an object can be sculpted.

More specifically, each patch holds a three dimensional control-volume over which a scalar trivariate Bspline function is defined, following Equation (1).

Efficient processing of the trivariates is achieved with the aid of another data structure in the form of an Octree that subdivides the working space and optimizes the access time to the different patches.

This synergy of an hierarchical Octree based subdivision of the three dimensional working space and a forest of scalar trivariate Bsplines yields several advantages:

- Ability of sculpting in different resolutions - Trivariate functions can be defined in arbitrary resolutions and different parts of the object can be sculpted in different levels of details. For example, when sculpting a human's head, one would probably sculpt the skull in a low resolution while adding the facial details, like the eyes and the lips, in a finer resolution.

- Arbitrary orientation - patches can be located in an arbitrary orientation. Therefore, the patches can be easily employed to model linear surfaces in arbitrary orientation and hence, yield to linear precision with ease.

- Smooth normal approximation - since the object is represented with the aid of C^k continuous uniform trivariate Bspline functions of order $k+2$, the derivatives of the trivariates can be evaluated at any location in the domain and in an arbitrary resolution. Further, the normals can be easily computed as the gradient of $q(u,v,w)$ (See Equation (1)). Low pass filtering can be achieved via higher order trivariate functions. The higher the order is, the smoother the final result. It should be recalled, however, that singular discontinuous locations may result in the zero set even if the implicit form is continuous. For example, the implicit function of $x^3 - y^2 = 0$ has a discontinuity at the origin.

- Simple instancing and viewing - the object is represented as a zero set of (a forest of) trivariates. Due to the affine invariance property of this trivariate Bspline form, instancing and transformation of the object in the three dimensional space is straightforward.

- Sampling and applying the *Marching Cubes* algorithm provides an efficient approximation of the model. Arbitrary fine sampling can be employed because exact and continuous implicit forms are employed. Further, the subdivision and the convex hull properties of the trivariate Bspline functions allow for a robust as well as an adaptive approach to piecewise linear approximation:

  For each trivariate Bspline patch q_i:
  1) If all the coefficients of the control-volume of q_i are above the constant set threshold, w_0, or below it, return.
  2) If q_i is at a predefined depth, sample it and return.
  3) Otherwise, subdivide q_i and repeat step 1 on the sub-patches.

  Due to the convex hull property of the trivariate Bsplines, it is certain, in step 1) above, that all the volume of the patch is below or above the constant set's threshold, w_0, provided all the coefficients are.

- Subdivision is feasible by taking advantage of the subdivision property of the uniform trivariate Bsplines, splitting a large or difficult modeling problem into two, during the design process. Refinement is also possible if the uniformity of the knot sequence is not mandatory. Alternatively, refinement into twice the resolution can also preserve this uniformity. In both cases, the refinement of the trivariate adds degrees of freedom and hence yields a finer shape control.

3.2 Data Structures

Herein, we consider only data structures. Section 3.3 will discuss the use of these data structures.

We employ three major data structures:

- A forest of scalar uniform trivariate Bspline patches.
- Sculpting tools as implicit forms.
- An Octree that subdivides the three dimensional working space.

3.2.1 The Trivariate Patches

The Bspline patches are formed of three dimensional meshes or control-volumes as was presented in Section 2. Each patch is assigned a resolution, which determines the density of the coefficients of the control-volumes. The higher the resolution is, the finer is the *Marching Cubes'* sampling inside the volume of the patch. Clearly, finer sampling takes more computation time, while allowing one to capture finer details. Nevertheless, the fact that finer resolutions typically occurs in smaller volumes, somewhat balances the tradeoff between the resolution and the required speed. The model itself is described as a forest of such patches summed together, where a patch contributes nothing outside it's domain. A patch can have arbitrary location, orientation and size which are determined by the user.

3.2.2 The Sculpting Tools

Sculpting is conducted with a three dimensional tool that is typically controlled by a three dimensional input device. The volume inside the tool is processed and affected.

Tools can be represented by any three dimensional implicit function of the form $w_0 = \mathcal{T}(x, y, z)$, or alternatively by any Boolean operations over such three dimensional implicit functions. Therefore, a decision can be easily made if a certain location is inside a tool or not. For example a point, P, is inside the tool depicted in Figure 2 if:

$$
\begin{aligned}
&P \in \bigcup_{i=0}^{2} C_i(x, y, z), \text{ where,} \qquad (2)\\
&C_0(x, y, z) = \{(x, y, z) \mid (\sqrt{x^2 + y^2} - a)^2 + z^2 \leq b^2\}, \text{ (the torus),}\\
&C_1(x, y, z) = \{(x, y, z) \mid (x^2 + y^2) \leq a^2 \wedge -b \leq z \leq b\}, \text{ (small cylinder),}\\
&C_2(x, y, z) = \{(x, y, z) \mid (x^2 + y^2) \leq (a + b)^2 \wedge 0 \leq z \leq l_2\} \text{ (large cylinder).}
\end{aligned}
$$

This tool is composed of a torus (C_0), and two cylinders (C_1 and C_2). a and b are the two radii of the torus, and l_2 is the length of cylinder C_2. The volume that is occupied by the tool has a shape of a finite cylinder which is rounded at one of it's ends. Figures 2 shows a side cross-section view of this tool.

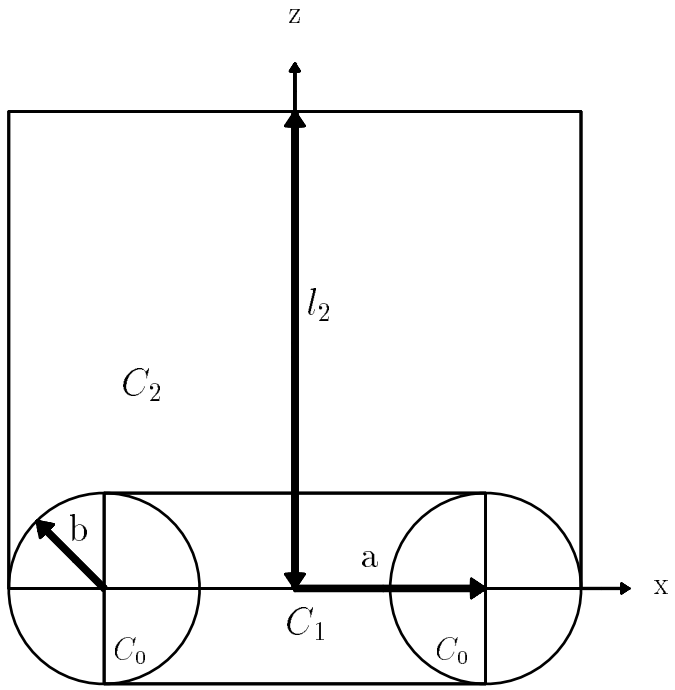


Figure 2: A side cross-section view of a sculpting tool with the shape of a finite cylinder which is rounded at one end. This tool consists of a union of a torus (C_0), a small cylinder (C_1) and a large cylinder (C_2).

The tool undergoes filtering in a similar way to [5, 13] (See Figure 3). Given location $P = P(x, y, z)$, the evaluation function of the tool returns the closest distance from P to the boundary of the tool. This distance is then used to filter the values which are assigned to locations inside the volume of the tool. For example, a linear filter, as shown in Figure 3 (b) assigns the maximal value to a location which is at the center of the tool's volume, and a minimal value to locations on the boundary of the tool's volume. Intermediate locations are assigned values which are a linear interpolation of these two extreme values. The user can control the shape and size of the tool by setting it's parameters i.e. a, b and l_2, as well as selecting a whole different type of tool out of several available tool types.

Other tools such as a chisel and a sand-paper tools may be realized. These tools should change their orientation according to the normal of the surface of the object. This orientation can be determined by computing smooth and accurate normals out of

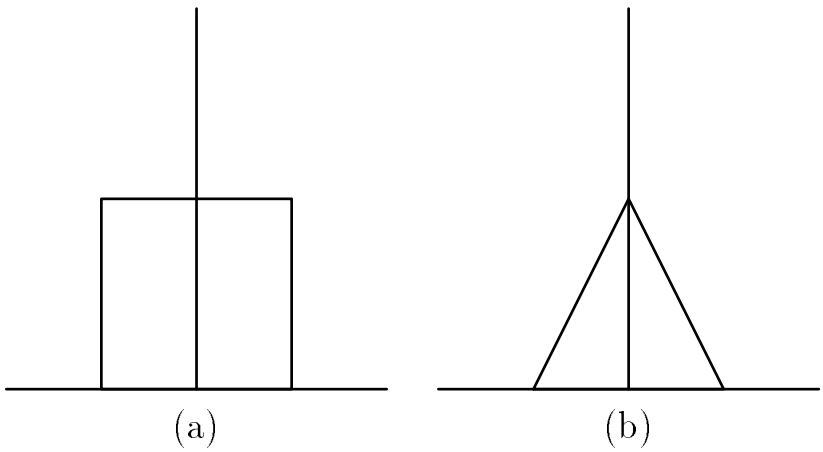


Figure 3: Two types of filters of tools. (a) shows a constant filter - all locations inside the volume of the tool are assigned the same value. This filter is prone to aliasing because of the discontinuities at the boundaries of the tool's volume. (b) shows a linear filter; the values that are assigned, are linearly interpolated according to location along the slope of the filter.

the available continuous implicit form.

In addition, the sand-paper tool must not be allowed to penetrate the object and should affect the surface of the object only. The limitation on the depth of the penetration of a sand-paper tool can be achieved by determining whether the tool is inside or outside the object, by simply evaluating the implicit form. Further, because of the availability of a closed form formulation of the implicit form, one can numerically guarantee that the sand-paper tool stops at a fixed prescribed depth from the surface of the object.

3.2.3 The Octree

An Octree is employed to divide the three dimensional working space according to the resolutions of the patches. Each voxel of the Octree holds references to the patches which overlap it's volume. A patch might be contained in more than one voxel, or alternatively a voxel can hold references to more than one patch.

An example for a subdivision of a two dimensional space can be seen in Figure 4. In this example, the patches are used to sculpt a face. Patch A is used to sculpt the low

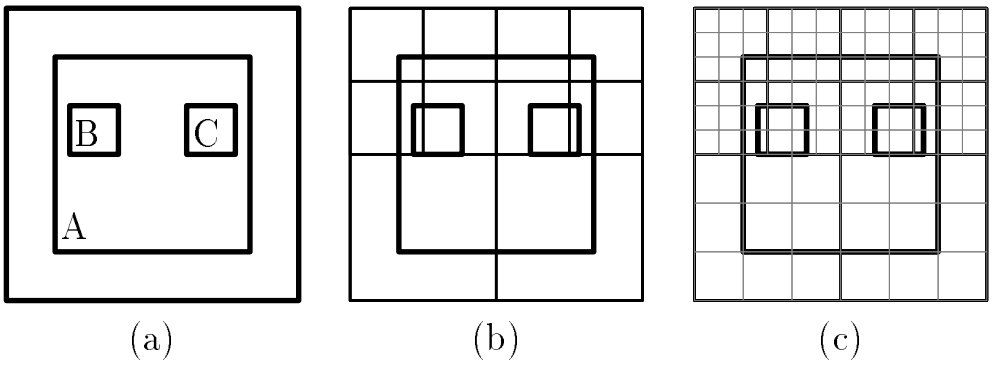


Figure 4: A two dimensional example showing three patches in the plane that is divided by a quadtree. (a) shows the three patches located in the plane. (b) shows how the patches overlay the cells of the quadtree. (c) shows a uniform subdivision of level three of each of the cells of the quadtree, toward the reconstruction of a piecewise linear approximation of the shape.

level detail parts like the cheeks and chin, whereas patches B and C are used to sculpt the eyes which are higher details parts. The area which holds patches B and C has a finer level of subdivision compared to the area which holds only patch A. This subdivision is used to sample the patches and visualize the object via the *Marching Cubes* algorithm.

As mentioned above the visualization of the object is conducted via the *Marching Cubes'* algorithm. The sampling rate for the *Marching Cubes'* algorithm inside each of the voxels of the Octree is determined by the patch with the maximal resolution in that voxel. Sampling is performed by further dividing each voxel with a three dimensional uniform sampling grid in each voxel. See Figure 4 (c). The size of the cubes of the sampling grid is determined by the resolution of the voxel as well as the dimension of the voxel.

An optimal subdivision of the Octree is a subdivision in which every voxel includes patches of similar resolution. Clearly, this is not always feasible, and two different patches with different resolutions that share a voxel would necessitate the sampling of the voxel at the higher resolutions, causing unnecessary sampling of the low resolution patch. In order to minimize this redundancy, when a new patch is created, the Octree is subdivided accordingly until one of the following occurs:

- The voxel is homogeneous in it's resolution.
- The voxel is completely contained inside all the patches which it holds references to.
- A predefined maximal subdivision depth is reached.

In the last two cases, the resolution is set to follow the patch with the maximal resolution.

3.3 Sculpting

A sculpting operation is conducted each time a motion event in the working space is received from the input device. This operation consists of the following stages:

3.3.1 Editing of the Coefficients of the Active Patch

The editing process modifies the values of the coefficients inside the control-volume of a patch. Editing can be conducted to one patch at a time, a patch that is selected by the user and is denoted as the *active patch*. Every motion event from the three dimensional input device assigns the sculpting tool with a new location. Given that location, the bounding box of the tool is computed. The tool's bounding box is then mapped to the local coordinate system of the *active patch*. With the aid of the implicit form that represents the tool, coefficients that are located inside the volume of the tool are determined. In order to add material, these coefficients are assigned with a value which defines a location occupied by material. In order to remove material the coefficients are assigned with a value which defines a location which is not occupied by material. These values are determined by the filtering of the tool as described in Section 3.2.2.

3.3.2 Constant Iso-surface Re-evaluations

The evaluation includes all the steps which are necessary to reproduce a piecewise linear representation of the object.

The first step samples the volume which was edited by the tool. The bounding box of the tool is pushed down the Octree's hierarchy, and is divided into the various

voxels. Each voxel then resamples the portion of the volume it received at the vertices of the cubes of the three dimensional sampling grid. This portion is sampled in the resolution of the voxel. Each of the cubes of the sampling grid in a voxel is sampled at it's eight corners, which are fed as input to the *Marching Cubes* algorithm. Sculpting is performed and evaluated locally, i.e. the patches are sampled, evaluated and modified only in the locations that are found inside or on the boundary of the volume of the tool, an optimization that enables almost interactive sculpting speeds, in reasonable patch resolutions.

Since adjacent cubes share vertices, sampling each cube separately will cause redundant samplings of the same vertex. Therefore, each vertex is sampled once, and then each cube is assigned with the values of it's eight vertices.

Let $A_i : I\!R^3 \longrightarrow I\!R^3$ be an affine transformation from the Euclidean space to the parametric domain $D(q_i)$ of a trivariate Bspline patch $q_i = q_i(u, v, w)$ (see Equation (1)). Simple instancing as described in Section 3.1 is conducted by assigning a different A_i transformation to the same patch. Given location $P_0 = (x_0, y_0, z_0)$, let $\mathcal{Q}_0$ be the set of patches such that

$$\mathcal{Q}_0 = \{q_i \mid A_i(x_0, y_0, z_0) \in D(q_i)\}. \tag{3}$$

Then,

$$\mathcal{F}(x_0, y_0, z_0) = \sum_{q_i \in \ \mathcal{Q}_0} q_i(A_i(x_0, y_0, z_0)), \tag{4}$$

where the trivariate functions employed, $q_i(u, v, w)$, are described in Equation (1).

The use of uniform Bspline basis functions allows one to perform the following computations during the *preprocessing stages*, further improving upon the expected interaction abilities:

- Sampling of the uniform basis functions - The uniform Bsplines basis functions are essentially the same basis function translated along a uniform knot vector. Therefore, once this basis function has been sampled at a desired resolution, the only task to perform during the interactive stage is to compute the proper translation of the indices of the basis functions according to the given parametric value.

- Multiplication of basis functions - For a trivariate of orders l by m by n, $l \times m \times n$ multiplications of the form $B_i(u)B_j(v)B_k(w)$ must be performed in order to evaluate Equation (1) at some location. Using the set of n samples of a single basis function, all possible multiplications can be computed during the preprocessing stage, at the obvious memory cost of $O(n^3)$. For example, for a single evaluation of a cubic trivariate, 4^3 multiplications are required in the evaluation of Equation (1). Then, assuming the basis function is sampled at 10 uniform locations along the unit interval, $64000 = 4^3 \cdot 10^3$ multiplications are to be precomputed and stored.

In order to achieve C^k continuity on the boundaries of the trivariate patches, the first and last $k+1$ rows, columns, and planes of the coefficients are coerced to a value of zero, assuming the order of the Bspline patch is larger than $k+1$.

The derivatives are evaluated in a similar way, using basis functions of one degree lower,

$$\frac{\partial q(u,v,w)}{\partial u} = \sum_{h=0}^{l}\sum_{i=0}^{m}\sum_{j=0}^{n} P_{hij} B'_{h,k}(u) B_{i,k}(v) B_{j,k}(w), \tag{5}$$

where $B'_{i,k}(u) = (k-1)\left\{\frac{B_{i,k-1}(u)}{t_{i+k-1}-t_i} - \frac{B_{i+1,k-1}(u)}{t_{i+k}-t_{i+1}}\right\}$. Assuming $t_{i+k-1}-t_i = t_{i+k}-t_{i+1} = k-1$, $B'_{i,k}(u) = B_{i,k-1}(u) - B_{i+1,k-1}(u)$. $\frac{\partial q(u,v,w)}{\partial v}$ and $\frac{\partial q(u,v,w)}{\partial w}$ are computed in a similar fashion.

The final stage in the constant iso-surface evaluation, derives the polygonal approximation which is computed with the aid of the *Marching Cubes* algorithm. Each invocation of the *Marching Cubes* algorithm is given the size and location of the sampling grid's cube and the scalar values of it's eight vertices. In order to reduce the *Marching Cubes'* computation time, all the 256 different cases of a single cube have been enumerated and pre-computed. This approach is slightly different from the way these cases are enumerated by [8], where 14 basic cases are computed and rotations are performed for all the other symmetric cases.

3.3.3 Rendering

In order to achieve interactive rendering times, a list of the active cubes is maintained. Whenever a cube is assigned a non-empty polygonal set by the *Marching Cubes* algorithm it enters this active list. Similarly, a cube is removed from the active list if it does not

contain any polygons. Hence, only the parts of the volume that intersect the boundary of the object are traversed and rendered.

3.4 Black Holes' Filling

When two adjacent voxels of the Octree are of different size and/or are sampled at different resolution levels, the polygonal surfaces generated by the *Marching Cubes* algorithm might have gaps along their adjacent faces. In order to close these gaps, that are also known as *black holes*, vertices from the voxel with the higher resolution should be moved towards the vertices and/or edges in the voxel with the lower resolution. An example for the correction of the black holes problem is shown in Figure 5.

Nevertheless, it is not always trivial to determine which vertex should be moved towards what vertex. Figure 6 depicts one such nontrivial case. The low resolution cube has two polygons with edges on a face adjacent to higher resolution cubes. It is impossible to decide which of the high resolution polygons should be moved towards the low resolution polygon. This decision is impossible to make, in general, since only a small fraction of the object is included in these cubes and one requires global object information in order to make the proper decision.

Due to the availability of a closed form representation to the shape as an implicit form, the low resolution cube can be subdivided and resampled until we reach cubes which have only one polygonal edge on it's adjacent faces. In other words, here one can employ the exact implicit form to detect the proper topology in this resampling process. The possibility of non-manifold singular locations necessitates a second termination at a predefined depth. Beyond this predefined depth, the object is considered to be a non-manifold.

3.5 User Interface

A main goal in the presented scheme was the desire to create an intuitive sculpting modeling tool. Therefore, the user interface is of great importance. In order to achieve this goal, the interface must enable the user to find her/his way in the three dimensional environment, and have a clear view where is the active patch, where is the object, and

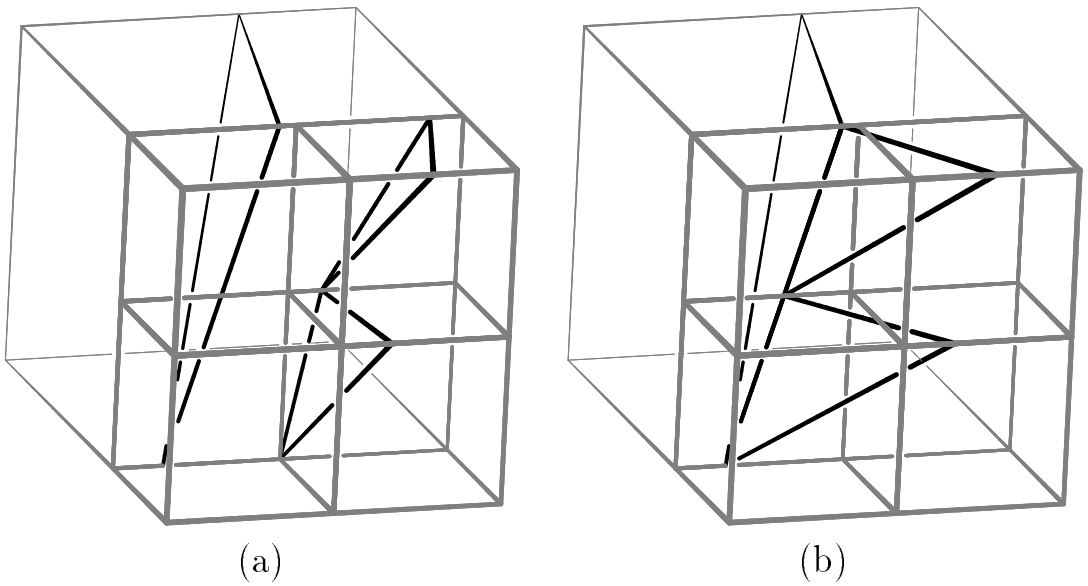


Figure 5: Four cubes of resolution $2R$ (front) with an adjacent cube of resolution R (back). (a) shows the polygons before the black holes filling process, while (b) shows the polygons after the holes have been filled.

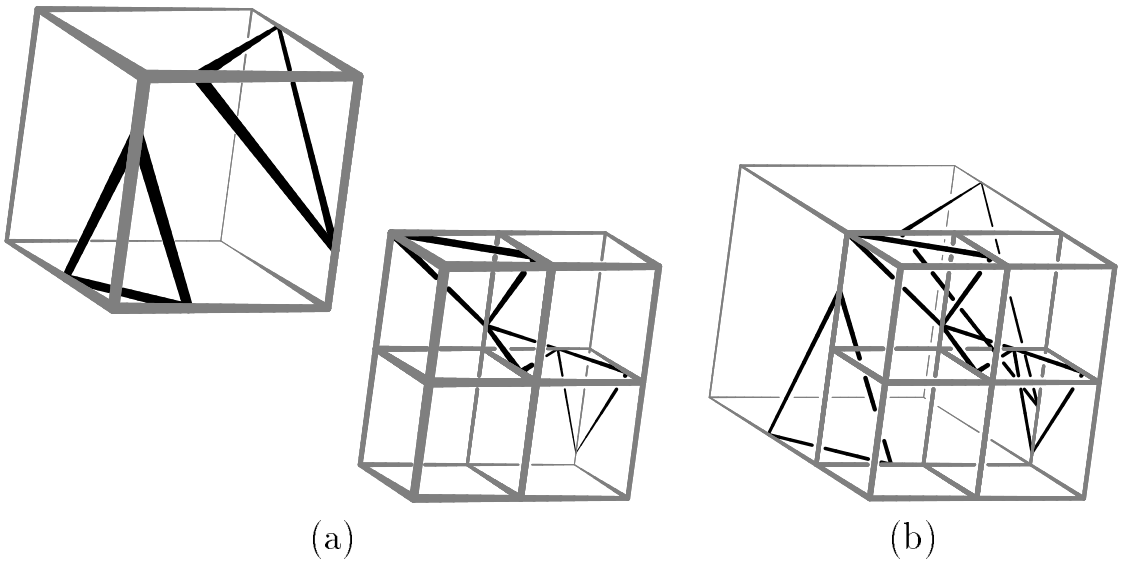


Figure 6: In this example, the cubes are of the same resolution and sizes as in Figure 5. In (a), the high resolution cubes are shifted away from the low resolution cube for better understanding of the scene. Since the low resolution cube has two polygons with edges on the adjacent face, it is impossible to locally decide where to move the vertices of the high resolution polygons. (b) shows all the cubes of (a) joint together.

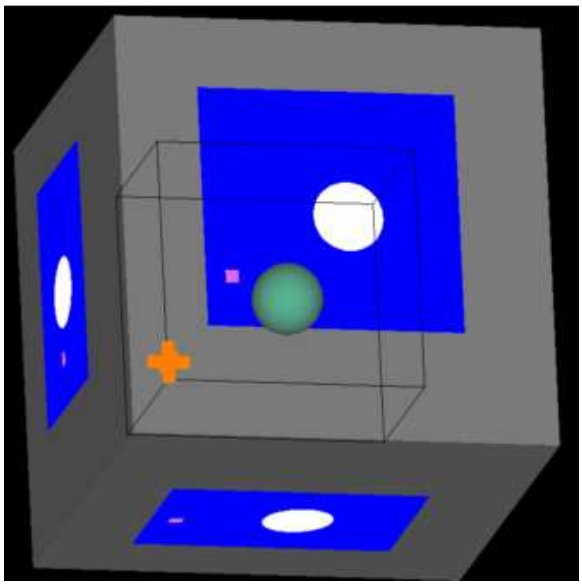


Figure 7: A spherical sculpted object with it's projections on each of the walls. The cross in the middle and it's three projections on the walls is the three dimensional tool. The active patch is marked by it's wireframe bounding box and in three projections as the dark rectangles on the walls.

the relative position of the tool to the previous two. Toward this end, the user interface includes the following (see Figure 7):

- Three projections of the tool - The tool is constantly projected on the walls surrounding the working space. In this way, the location of the tool can be intuitively pinpointed. In order to keep the working space constantly visible, only the three back walls (depending on the viewer's location) are rendered.

- Three projections of the active patch - Understanding the location of the tool is insufficient. For accurate sculpting, the location of the active patch must be described as well. The bounding box of the active patch and the zero set of the object are also projected on the walls. All these projections enable the user to visualize the location of the tool relative to the active patch and the object under construction. In addition, the bounding box of the active patch is constantly shown in three space. Whenever the tool enters the active patch, the color of the bounding box changes. See Figure 7.

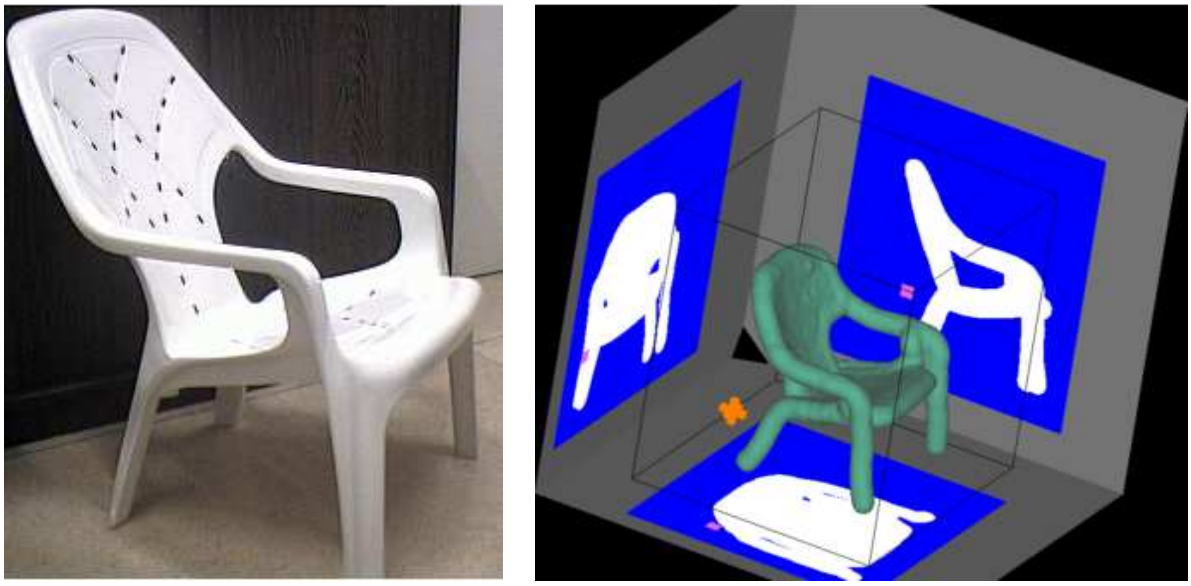
- Creation of new patches - Patches are created by specifying their location, size and orientation with the three dimensional input device. In order to create a new patch, the user has to select a corner and drag the three dimensional input device to the other corner. While dragging, the bounding box of the new patch is rendered. In order to create an oriented patch, the user has to prescribe two corners in a similar way, but now a central axis connecting two opposite faces is displayed during the dragging. Nailing the last degree of freedom can be conducted by specifying a rotation angle around this axis. Thus, the position and orientation of the new patch in relation to the rest of the object is clearly understood.

- Viewing transformations - The object is approximated in a polygonal representation form and hence viewing transformations are simple. The user can view the object from an arbitrary viewing point, interactively.

4 Examples

This section includes examples of several objects that were reverse engineered and sculpted with the aid of the Ascension Flock of Birds [1] and Magellan [9] multi-axis input devices. The developed software is based on the Glut [6] toolkit from SGI and it runs on both SGI and PC WinNT environments. All the examples in this section, and unless otherwise stated, were recorded off an Onyx SGI having a reality engine graphics board. The Bird is a three dimensional input device that is capable of capturing position as well as orientation. The Magellan has manipulation capabilities of three-space positions. Reverse engineering was conducted by moving the Bird along the surfaces of the objects, and assigning each position and orientation to the three dimensional sculpting tool. Sculpting was similarly performed using both the Magellan and the Flock of Birds, this time in free motion.

Figure 8 (a) shows a plastic chair. It's reverse engineered object is portrayed in Figure 8 (b). The size of the working space (which is contained inside the gray walls) is $64 \times 64 \times 64$ unit intervals, and the entire chair is sculpted in a single tri-cubic patch. The resolution of the patch is two control coefficients per unit interval. The chair was



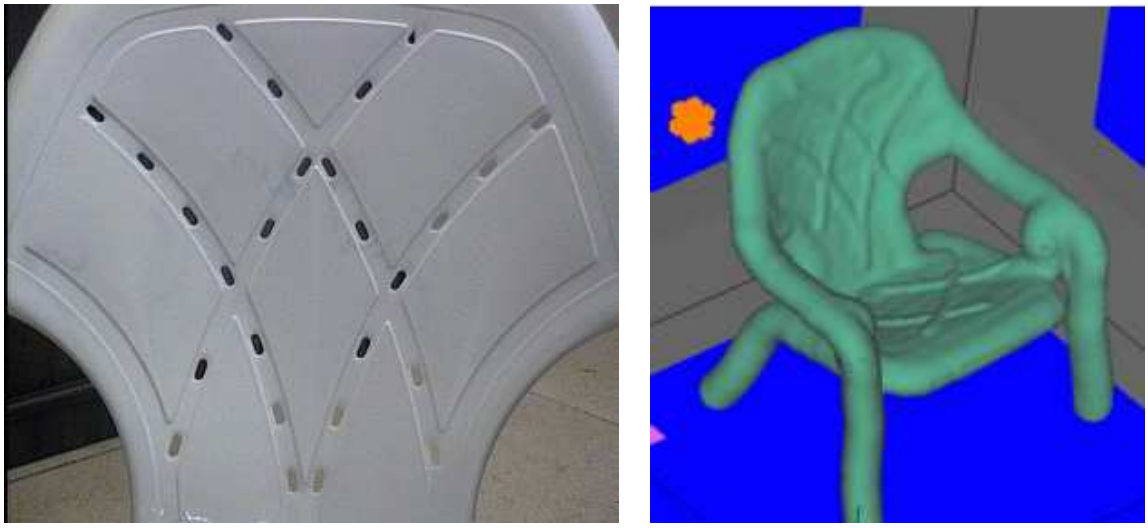
(a) (b)

Figure 8: A plastic chair in (a), and it's sculpted object in (b).

sculpted by a spherical tool with a radius of two unit intervals. Sculpting the chair took about 30 minutes. It is composed of over 30000 polygons.

Figure 9 (a) shows a closeup view of the back of the plastic chair in Figure 8 (a). Apart from the holes, it can be seen that the chair is not smooth but has mild engraved stripes. Figure 9 (b) shows the sculpted chair with the stripes. While stripes were sculpted as protrusions, the holes were not modeled since these fine details were too small to the size of the sensor of the Ascension's input device. The sculpting tool that was used to sculpt the stripes was a rounded cylinder similar to the one depicted in Figure 2. The orientation of the tool was received from the Bird, and the Z axis of the tool was aligned with the normal of the surface. In order to sculpt the stripes, two additional tri-cubic patches are defined around the seating and the back regions of the chair from Figure 8. These patches have a resolution of four control coefficients per unit interval. Adding the stripes took about 10 minutes. The stripes added about 84000 polygons to the original object.

Figure 10 (a) shows a kid's slide. Figure 10 (b) shows it's sculpted object and

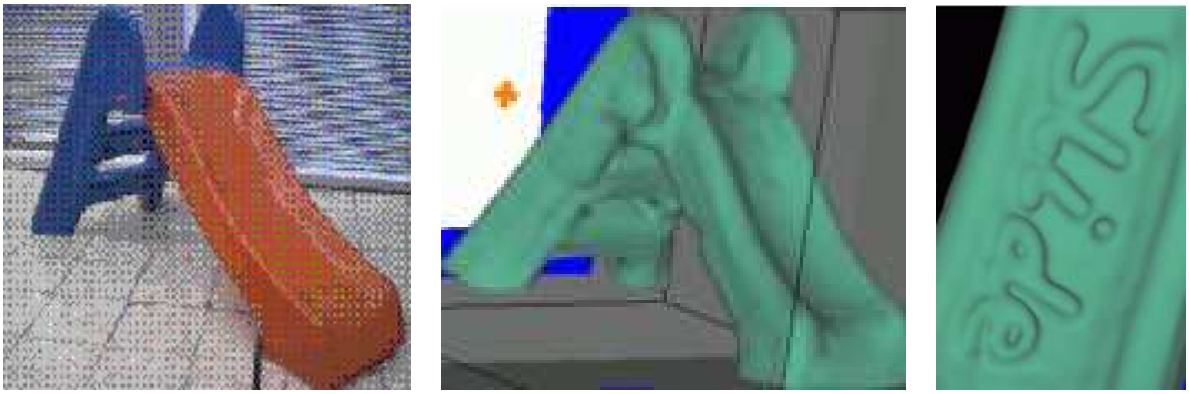


(a) (b)

Figure 9: In (a), a closeup of the back of the plastic chair in Figure 8 is shown, while the sculpted chair with the stripes is shown in (b).

Figure 10 (c) shows the same slide but with a high resolution caption on it. The size of the working space is $64 \times 64 \times 64$ unit intervals, and the entire slide is sculpted in a single tri-cubic patch. The resolution of the patch is two control coefficients per unit interval. The slide is sculpted by a spherical tool with a radius of one unit interval. Sculpting the slide took about 2 hours and it is composed of over 35000 polygons. In order to sculpt the caption an additional tri-cubic patch is defined around slope of the slide. The resolution of this additional patch is four control coefficients per unit interval. The caption is also sculpted by a rounded cylinder tool similar to the one depicted in Figure 2. The engravings in the middle of the letters d and e are sculpted by a rounded cylinder tool which is used as a *chisel tool* to sculpt engravings instead of protrusions. Adding the caption took about 30 minutes. The caption added about 80000 polygons to the original object.

While all the examples in this work employs smooth normals that were derived from the gradient of the implicit form, the capabilities to compute exact and accurate normals are compared and demonstrated in Figure 11, a scene that was captured off an implementation on a PC running Windows NT. The displayed sculpted object is



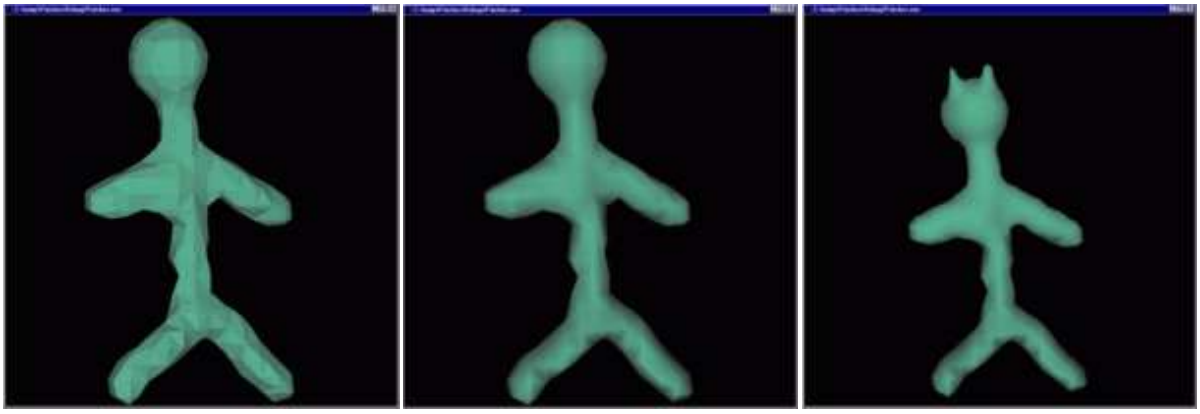
(a) (b) (c)

Figure 10: A kids' slide in (a), and it's sculpted object in (b). In (c), a "slide" caption has been added to the slope.

polygonal and therefore adjacent polygons are highly distinguishable in flat shading (Figure 11 (a)). It can be seen that the separate polygons are far less distinguishable when smooth shading takes place with the aid of smooth normals, computed via the gradient of the implicit form (Figure 11 (b)). In Figure 11 (c), a sculpting operation is conduced, creating two horns. This sculpting operation has been conducted at a higher level of details by defining a second patch with a higher resolution around the head.

Figure 12 shows an example of sculpting with a sand-paper tool. The engraving made by the sand-paper tool consists of three different levels of depth. At Each such level, the depth of penetration is fixed, yielding engravings (pockets) with smooth bottoms. Figure 12 (f) shows the final shape, after a second trivariate Bspline patch has been added to model the legs.

Figure 13 shows a human face sculpted out of data generated by a three dimensional scanner. The generated data consists of a cloud of (x, y, z) points that is shown in Figure 13 (a). Figure 13 (b) presents the recovered polygonal iso-surface out of the sculpted shape. Sculpting was conducted by placing the tool in each (x, y, z) location of the data and once completed, the geometry behind this cloud of points is captured as a trivariate Bspline model, enabling further manipulations. The object is sculpted by a single cubic trivariate patch with a resolution of two control coefficient per unit interval



(a) (b) (c)

Figure 11: Flat shading versus smooth normals shading. (a) shows the object in flat shading, while (b) shows it shaded with smooth normals derived from the tri-quadratic patch. (c) shows the object after conducting a freeform sculpting of two horns. These figures were recorded off a PC running Windows NT.

located in a working space of $64 \times 64 \times 64$ unit intervals. The tool used for sculpting is a sphere with a radius of 1 unit interval. The polygonal approximation of the face model, shown in Figure 13 (b), consists of about 84000 polygons. Figure 13 (c) shows the possibilities in further sculpting of such object, extracting out the two pupils.

5 Conclusions and Future Work

In this paper, we have described a three dimensional interactive sculpting system that employs scalar uniform trivariate Bspline functions. This introduced representation yields several advantages like sculpting in different levels of details and arbitrary orientations, smooth and accurate normals, simple instancing and viewing, and the ability to save the exact implicit form instead of a large polygonal data set that is only an approximation.

The presented approach provides the user with an intuitive sculpting abilities in an interactive speed for modeling arbitrary geometries and/or topologies. Future work involves measurements and different kinds of analyses that can be conducted upon the object due to the fact that the implicit form of the trivariate objects can be evaluated

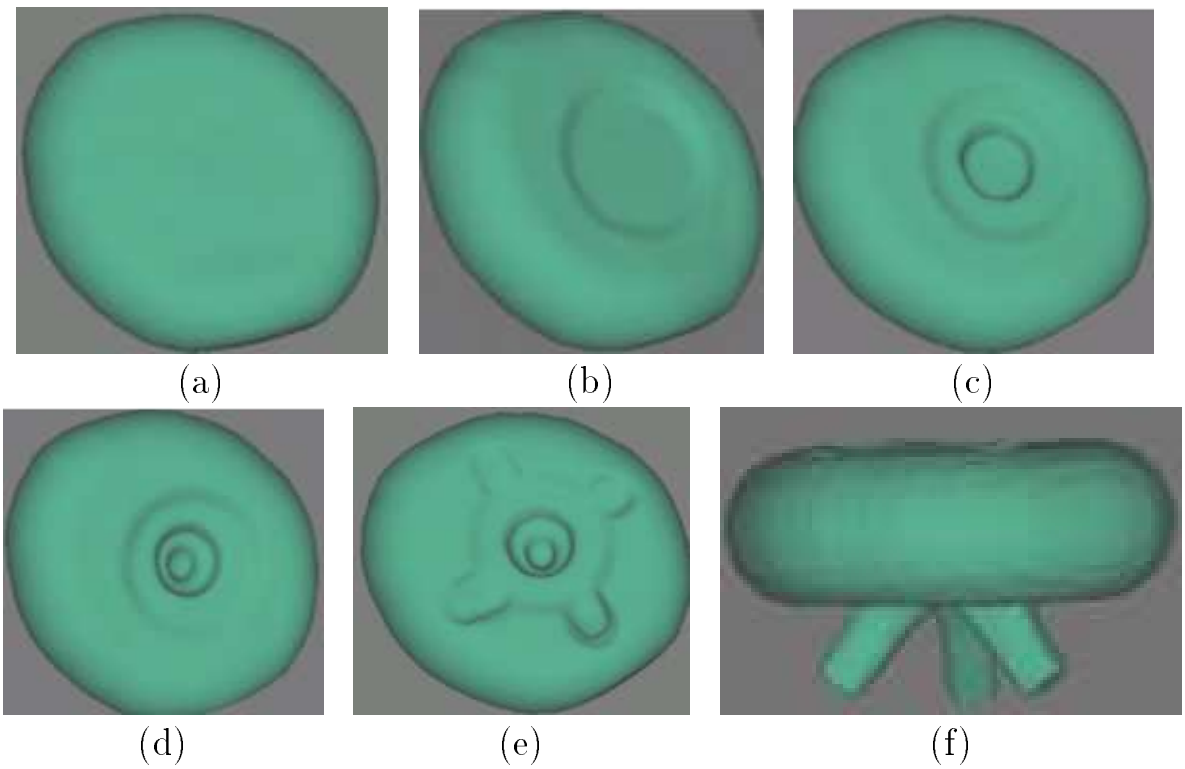


Figure 12: Operations using a sand-paper tool to create an ashtray-like object. (a) shows the initial object. (b), (c), (d) and (e) show the object at different stages. In each stage the depth of penetration is increased. (f) shows the object from a different angle with three supporting legs.

in an arbitrary precision. This is particularly useful for reverse engineering where a computational model can be created from a physical object and then manipulated. The Bspline representation of the objects enables the exploitation of various algorithms like subdivision, refinement and degree raising.

The embedding of force feedback input devices can greatly increase the level of reality to novice as well as experienced users and might prove closer to reality for sculptors. It is simple to check, during the interactive control of the force feedback tool, if a location is inside or outside an implicit form. In addition, the exact implicit form can be employed to simplify the presented black holes filling algorithm because it accurately answers the inside/outside query, at any location in the working space. *Marching Cubes'* ambiguities can be resolved by determining for each vertex, as well as any other point, whether it

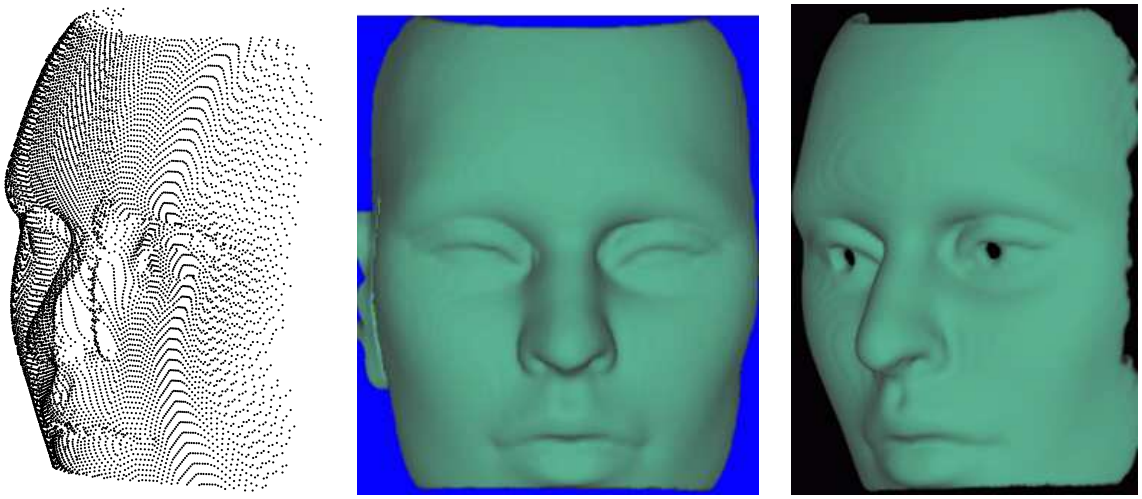


Figure 13: A human face scanned by a three dimensional scanner. (a) shows the cloud of (every tenth) points generated by the scanner. (b) shows the object sculpted out of that data and (c) shows the object after a simple sculpting operation was conducted upon, extracting out the two pupils.

is inside or outside the object. Therefore, a decision can be made where to move the high resolution vertices even in a nontrivial case such as the one depicted in Figure 6. Finally, decimation methods [4] can clearly be applied to the polygonal data sets that are created, significantly reducing the size of the output.

References

[1] Ascension Technologies. http://www.ascension-tech.com

[2] S. Coquillart. A Sculpting Tool for 3D Geometric Modeling. *Computer Graphics, Siggraph* 24, pp. 187-196, August 1990.

[3] D.R. Forsey and R.H. Bartels. Hierarchical B-Spline Refinement *Computer Graphics, Siggraph* 22(4), pp. 205-212, August 1988.

[4] M. Garland and P.S. Heckbert. Surface Simplification Using Quadratic Error Metrics. *Computer Graphics, Siggraph* pp. 209-216, August 1997.

[5] T.A. Gaylean and J.F. Hughes. Sculpting: An Interactive Volumetric Modeling Technique. *Computer Graphics, Siggraph* 25(4), pp. 267-274, July 1991.

[6] Glut 3.0. http://reality.sgi.com/employees/mjk_asd/glut3/glut3.html

[7] Y. Jeng and Z. Xiang. Moving Cursor Plane for Interactive Sculpting. *ACM Trans Graph* 15, pp. 211-222, July 1996.

[8] W.E. Lorensen and H.E. Cline. Marching Cubes: A High Resolution 3D Surface Construction Algorithm. *Computer Graphics, Siggraph* 21(4), pp. 163-169, July 1987.

[9] Magellan Space Mouse http://www.logicad3d.com

[10] S. Mizuno, M. Okada and J. Toriwaki. Virtual Sculpting and Virtual Woodcut Printing. *Visual Computer* 10, pp. 39-51, 1998.

[11] A. Pentland, I. Essa, M. Friedmann, B. Horowitz and S. Sclaroff. The Thing-World Modeling System: Virtual Sculpting By Modal Forces. *Computer Graphics, Siggraph* 24(2), pp. 143-144, August 1990.

[12] T.W. Sederberg and S.R. Parry. Free-Form Deformation of Solid Geometric Models. *Computer Graphics, Siggraph* 20(4), pp. 151-159, August 1986.

[13] W. Wang and A.F. Kaufman. Volume Sculpting. *Symposium on Interactive 3D Graphics Proceedings, ACM Press*, pp. 151-156, April 1995.