Artistic Surface Rendering Using Layout of Text

Tatiana Surazhsky* and Gershon Elber†
Technion, Israel Institute of Technology, Haifa 32000, Israel.

Abstract

An artistic rendering method of free-form surfaces with the aid of half-toned text that is laid-out on the given surface is presented. The layout of the text is computed using symbolic composition of the free-form parametric surface $S(u, v)$ with cubic or linear Bézier curve segments $C(t) = \{c_u(t), c_v(t)\}$, comprising the outline of the text symbols. Once the layout is constructed on the surface, a shading process is applied to the text, affecting the width of the symbols as well as their color, according to some shader function. The shader function depends on the surface orientation and the view direction as well as the color and the direction or position of the light source.

Keywords: non-photorealistic rendering, line art, font art.

ACM CCS: I.3.3 [Computer graphics] Line and curve generation, I.3.5 [Computer graphics] Curve, surface, solid, and object representations.

1 Introduction

Non photorealistic rendering (NPR) is a widespread area of study in contemporary computer graphics research. Algorithms for line art rendering [7,9,10,13,14,19,20], artistic screening [16], artistic dithering [17], digital engraving [15, 18], or pen-and-ink illustration and animation [14, 21, 25] have been proposed in recent years as alternatives to traditional photorealistic synthetic imagery. Non photorealistic rendering techniques are aimed at better conveying shape, possibly providing more details, or, alternatively, present the shape in a more illustrative and appealing way.

NPR techniques produce different kinds of non-photorealistic images in numerous applications. Looking at all sorts of architectural sketches or illustrations in technical textbooks and manuals, one can find hand-made line art or other classes of drawing styles, developed in fine arts. Many of the contemporary NPR methods employ the three dimensional geometry in order to convey more information than just the shading. Consider an image of a statue with two different body parts that are equally illuminated. A photorealistic algorithm will paint both of them indistinguishably. Nonetheless, in the art of wood carving (see [12]) one may find the stream lines of the object, i.e. sets of lines drawn along the general shape of the object, which make the different body parts unique. The objects are rendered with *curves* that are drawn along

*Applied Mathematics Department, E-mail: tess@cs.technion.ac.il
†Faculty of Computer Science, E-mail: gershon@cs.technion.ac.il

some intrinsic geometrical features such as curvature. Recent computer graphics research strived for an automatic method that produces synthetic images with the similar look of wood carving.

In [18], the authors introduce a technique that simulates the production of engraved plates from the information contained in two dimensional image. The effect is achieved with the aid of a grid-less half-toning that employs curves as the drawing primitives. In [15], the basic techniques for digital facial engraving are introduced.

The line art rendering technique of [7] utilizes a coverage that is based on isoparametric curves of the surfaces of the object. The rendering process is performed by varying the density of the isoparametric curves as a function of the illumination of the surface. In [19, 20], three dimensional properties of the surfaces such as the normal vectors and principal directions are employed in order to create the line art drawing. Provided with curvature analysis, the authors generate stream lines that are traced in the two dimensional directions in order to define the line strokes of the drawing. The user is allowed to interactively select reference lines, and then all the additional strokes are produced automatically. In [13], a line art rendering is performed by generating hatch marks that can convey surface shape. In [17], a method for multi-color artistic dithering is proposed. This method produces a multi-color non-overlapping surface coverage using a barycentric combination of color intensities. In [25], an algorithm for rendering parametric free-form surfaces in pen-and-ink is presented, while in [21], the authors describe an interactive system for creating pen-and-ink style drawings from grey scale images.

Many traditional half-toning techniques use small repetitive screen elements that characterize the color at different screen locations. Usually, if such elements are perceived by the human eye, the visible shape of the elements is considered as a half-toning artifact.

One approach for half-toning may be found in [16]. In their work, the authors tune a shape of the screen elements and create screening effects that are considered undesirable in traditional half-toning processes but may be modified to convey additional information for artistic purposes. Some more sophisticated artistic shapes such as the ornaments of Escher, are used as screen basic elements, and the smooth alteration of the shape of the artistic screen basic element is controlled to offer a half-toning method. The inspiration for such techniques may be found in the work of medieval artists (see [5]), where polygonal patterns have been used in order to create ornaments, in each separate tile. Plenty decorative motives that incorporate beautiful calligraphic work with letter shapes are used in the Islamic art [5], decorations that are well-distributed over a given geometric surface.

The results of [16] demonstrate that contour-based generation of half-tone screens provides a novel way of conveying information. One example that is demonstrated in [16] is the use of half-toning to avoid

counterfeiting. Bank notes may include half-toning images with intensity levels that are produced by micro letters of varying size and shape. The US treasury uses micro-printing techniques for generating letters along the curved contours in order to protect the bank notes.

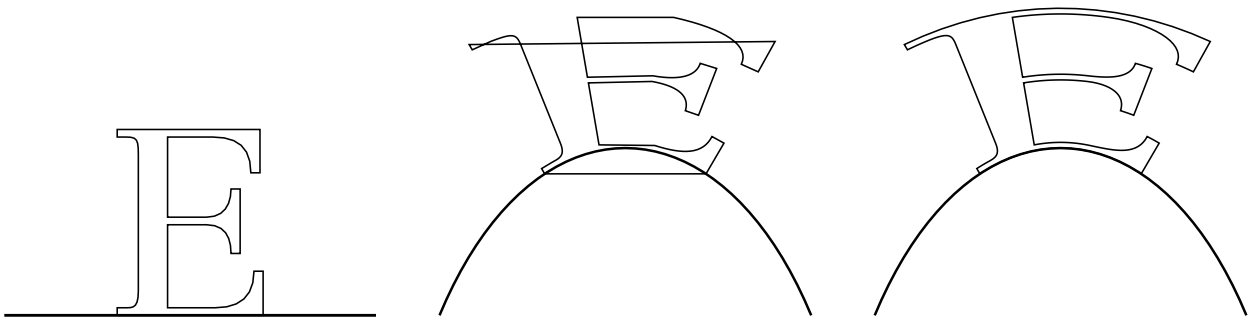
In this work, we introduce a new line art rendering scheme of free-form surfaces that exploits the layout of text over the given parametric surface, in three-dimensional space. The illumination intensity of the surface as computed using traditional shaders is mapped into the width of the text. By uniformly placing symbols over the surface, the width of the curves representing the text symbols is linearly modified and based on the computed illumination this width controls the tone reproduction. Hence, the curves representing the text serve as *tone reproduction curves*.

This work is organized as follows. In Section 2, we consider some basic definitions. A smooth deformation of text in two dimensions is discussed in Section 3. The details of the three dimensional deformation and the surface rendering method could be found in Section 4. In Section 5, some examples of the presented method could be found. Finally, we conclude in Section 6.

2 Background

We now consider several definitions that we are about to employed. We use the term *symbol* as a synonym for a *text character* or *letter*. A *font* is a set of printable or displayable symbols of specific style and size. A design of a set of fonts is called a *type-face*. The two most widespread type-faces are the *True Type* that is used by Microsoft Windows [2] and the Adobe's *Type 1* fonts (see [3, 4]). These two types of type-faces are representatives of *scalable* or *vector fonts*. Such fonts are also called *outline fonts*. The outline of characters of scalable font is defined using a geometry that consists of vector curves. Thus, any kind of affine transformations can be easily applied to the text. The curves between the end-points of the vectors are usually specified by using either cubic and linear *Bézier spline curves* in Type 1 [4] fonts or quadratic and linear *Bézier spline curves* in True Type fonts [2]. While the most common use of text is printing and publication, there are plenty of other applications that require text manipulation functions such as computer animation and computer aided design and art. Hereafter, we employ Type 1 fonts and hence consider only linear and cubic Bézier spline curves.

Consider the following problem: given a text string and a free-form parametric curve $\gamma(t)$, denoted a *base curve*, layout the predefined text string to follow the path of curve $\gamma(t)$. In [14], strokes are bent along a path using "skeletal stroke deformation" toward better control of vector graphic strokes, for example for animation. Existing work on text deformations manipulates the text using one of two methods. The most simple approach defines a best suited rigid motion transformation for each symbol (see pages 171 –



(a) (b) (c)

Figure 1: A deformation of the letter 'E' using bending of a base line (thick). The original character is shown in (a). Character deformation using mapping of the control polygons is shown in (b). The deformation of the symbol using the algorithm we adopt here is shown in (c). Base lines are shown thick.

173, "Program 11/Placing Text Along an Arbitrary Path" in [1], for an example). A second technique, based upon a deformation transformation, maps the control points of the Bézier curves comprising the text geometry [16,21] following some deformation of the parametric domain. This second method is more accurate than the rigid motion transformation and may produce better results if the text symbols are approximated well enough by the control points of the curves. However, both methods do not guarantee that self-intersection free letters will remain self-intersection free after the deformation. Moreover, both methods cannot change the shape of linear segments even if the domain of the layout has been continuously and smoothly transformed by a non linear deformation function with high curvature. Undesirable artifacts may appear, that affect the continuity of adjacent segments (see Figure 1).

3 Smooth Deformation of Text in the Plane

A free-form deformation technique for solid geometric models is introduced in [22]. Having a deformation mapping function $\mathcal{M} : \mathrm{R}^n \longrightarrow \mathrm{R}^n$ and an object $\mathcal{O} \subset \mathrm{R}^n$ as an input, one could warp $\mathcal{O}$ following $\mathcal{M}$ as $\mathcal{M}(\mathcal{O})$. The selection of the mapping provides a precise control over the process of the deformation. The method of [22] resembles the technique we employ herein, yet for volumetric solid models. In this work, we deform geometry that represents text lying inside a planar domain $\mathcal{D}$. The geometry of the string is expected to lie in the parametric domain $\mathcal{D} = [u_1, u_2] \times [v_1, v_2]$ of the deforming surface patch.

A text deformation method that provides smooth and accurate results may be found in [24]. Consider a text string that has been created with the aid of an outline font. The geometry of the text is prescribed via a set of linear and cubic Bézier curve segments $\{C_i(t)\}_i$. The Bézier curves are defined via a finite number of control points (two in the linear case, and four in the cubic case).

(a) (b)

Figure 2: Two linear and parallel edges of a symbol with a base line (thick) are shown in (a). The deformed base line with the mapped end control points and edges, is shown in (b). The straight lines intersect each other in (b). Moreover, part of the symbol even expand below the base line. Note that the lengths of the dashed lines, h_1 and h_2, (that are normal to the base line) have not been modified.

The main idea of the text deformation method used in [24] employs the natural and precise solution of the deformation problem that is presented by a symbolic composition (see [8]). In [24] and for the planar case, we first define a planar deformation surface $S(u, v)$ as a mapping from $\mathrm{R}^2 \longrightarrow \mathrm{R}^2$. The text to be deformed, $\{C_i(t)\}_i$, is assumed to be contained in the parametric domain of $S(u, v)$. $S(u, v)$ is then composed with one curve segment, $C_i(t) = \left\{c_u^i(t), c_v^i(t)\right\}$, at a time. The result is a new smooth geometry, defined by

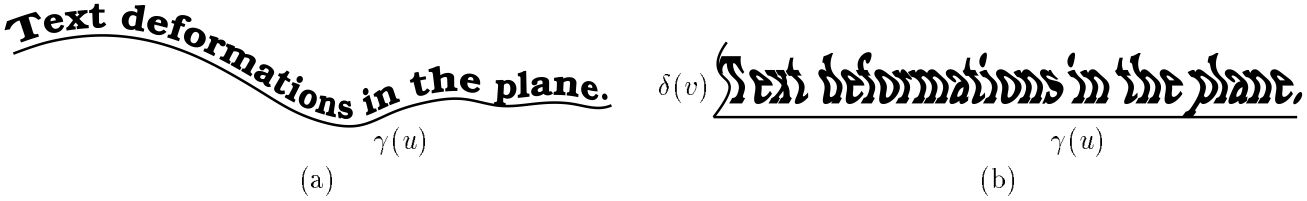
$$S \circ C_i(t) = S\left(c_u^i(t), c_v^i(t)\right), \tag{3.1}$$

again assuming all the geometry of the original symbols lie in the parametric domain of the surface $S(u, v)$.

Provided that surface $S(u, v)$ is self intersection free, the resulting layout of the given text (that is originally self intersection free as well) can have no self intersections or intersections between its symbols, see [24] for more. This statement does not hold if one considers either one of the two previous deformation methods of [16] or [1]. The technique of [16] may produce further undesirable artifacts, as can be seen in Figures 1 and 2.

4 Text Deformation in Three Dimensions

In [24], the definition of the planar surface $S(u, v)$ has several degrees of freedom. The user provides the text string and specifies the lower boundary of the surface $S(u, v)$, denoted as the *base curve*. A vertically



(a) (b)

Figure 3: Planar examples of text deformation, following [24]. An example of a text layout over the base curve $\gamma(u)$ is shown in (a). In (b), a text deformation guided by a shaping curve $\delta(v)$ is presented.

oriented left boundary of $S(u,v)$ that is called the *shaping curve* (see Figure 3) might also be specified, prescribing further alterations in the shape of the output.

In this work, we are interested in placing text on a given three dimensional free-form parametric surface. Thus, $S(u,v)$ is assumed to be a general three dimensional free-form surface.

4.1 Text Deformation Algorithm

Let $S(u,v)$ be a general B-spline surface. We now review the algorithm that maps the linear/cubic Bézier geometry, $\{C_i(t)\}_i$, of the given text onto a general B-spline surface, $S(u,v)$.

4.1.1 The Surface Subdivision Stage

As a first step, $S(u,v)$ is subdivided in the v direction into several strips, the number of which is equal to the number of desired lines of text.

We seek a distribution of the lines of text on $S(u,v)$ that is fair. Having a prescription of the number of lines of text N, to be placed on the surface, we employ the parameterization of the surface to make all the surface strips appear of similar height, in Euclidean space. Typically, the field of the parameterization is not uniform along $S(u,v)$. Assume u_{min} and u_{max} are the minimal and the maximal values of the u parameter in the parametric domain of $S(u,v)$, while v_{min} and v_{max} are the bounds of the parameter v. Calculate the velocity vector field $\frac{dS(u,v)}{dv}$, and consider it at $\overline{u} = \frac{u_{min}+u_{max}}{2}$ for all v and while subdividing the surface (see Figure 4). For a given number of lines of text, N, we approximate the value of the parameter v_i for the bottom cut of the i-th strip so that the following holds:

$$\frac{i}{N} \int_{v_{min}}^{v_{max}} \left\| \frac{dS(\overline{u},v)}{dv} \right\| dv = \int_{v_{min}}^{v_i} \left\| \frac{dS(\overline{u},v)}{dv} \right\| dv. \tag{4.1}$$

With the assumption that the value v_i in Equation (4.1) does not change much for different values of the u parameter, a uniform distribution of the N lines could by derived. The actual computation of v_i at $\overline{u} = \frac{u_{min}+u_{max}}{2}$ could be conducted via, for example, a binary search over v and a piecewise linear approximation of the arc-length function of $\int \left\| \frac{dS(\overline{u},v)}{dv} \right\| dv$.

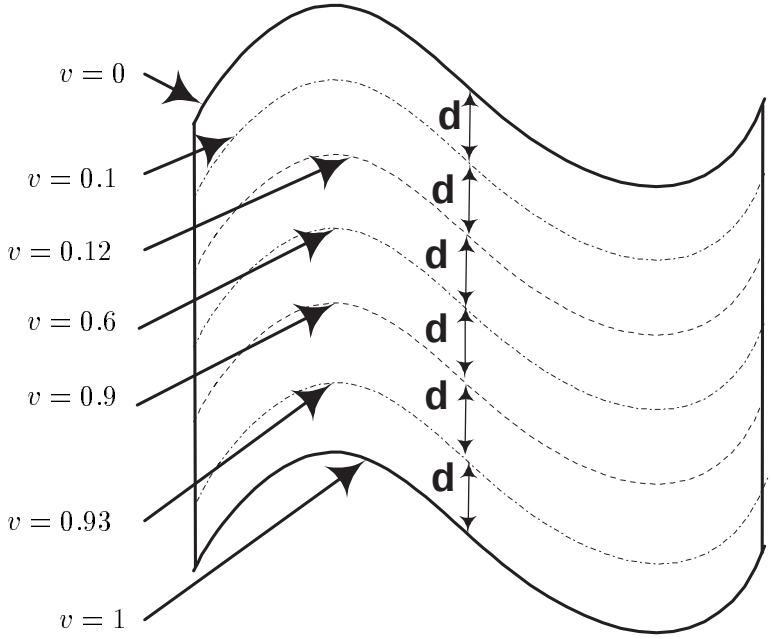


Figure 4: The surface $S(u, v)$ must be subdivided into strips with similar height, d, for any given parameterization of $S(u, v)$.

As a second step, the input text is split into the selected number of lines, so that all the lines would have a length of text that is proportional to the arc-length of that line on the surface. This process is conducted by computing the length of the entire body of text in relations to the accumulated lengths of all the base lines of surface strips $S_j(u, v_j)$ as derived using Equation (4.1). The body of text is then broken into N lines, each of which with a length that is proportional to the length of the corresponding surface strip. We strive to break the input text into lines in such a way that the following relation holds:

$$\frac{\int_{u_{min}}^{u_{max}} \left\| \frac{dS_i(u,v_i)}{du} \right\| du}{\sum_{j=1}^{N} \int_{u_{min}}^{u_{max}} \left\| \frac{dS_j(u,v_j)}{du} \right\| du} = \frac{\mathcal{L}_i}{\sum_{j=1}^{N} \mathcal{L}_j}, \tag{4.2}$$

where $\mathcal{L}_j$ is the length of the text that is assigned to the j-th surface strip.

4.1.2 Composing 3D Text

The composition stage of the presented algorithm resembles the algorithm of [24], for each surface strip $S_j(u, v)$. Recall that the corresponding symbols of text are represented by sets of linear and cubic Bézier curves $C_i(t)$. The B-spline surface representing the j-th strip, $S_j(u, v)$ is first converted into a set of Bézier surfaces by subdividing $S_j(u, v)$ at all its internal knots, u_k or v_l (See Figure 5). Further, all the Bézier

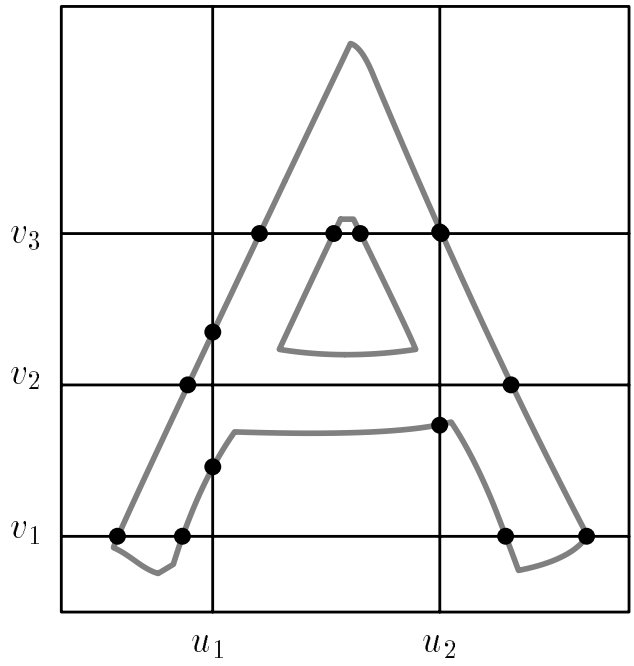


Figure 5: The letter 'A' is shown in the parameter space of the surface strip $S_j(u,v)$. The surface as well as the curve are subdivided at all the interior knots, u_k or v_l, of $S_j(u,v)$.

curves are similarly subdivided at the internal knots of u_k and v_l, solving for the following:

$$c_u(t) = u_k \quad \text{and/or} \quad c_v(t) = v_l. \tag{4.3}$$

The solution set of Equation (4.3) may be found analytically, solving for either a linear, or a cubic equation, and considering only the real roots in the range of the parametric domain of $C_i(t)$.

The deformation is completed by composing the subdivided segments of the Bézier curves with the corresponding subdivided Bézier patches. Let $C(t) = (u(t), v(t))$ be a Bézier curve such that $u(t), v(t) \in (0, \ldots, 1), \forall t$ and let S be a Bézier surface. Then,

$$\begin{aligned} S(u(t), v(t)) &= \sum_{i=0}^{n} \sum_{j=0}^{m} P_{ij} B_j^m(v(t)) B_i^n(u(t)) \\ &= \sum_{i=0}^{n} \left(\sum_{j=0}^{m} P_{ij} B_j^m(v(t)) \right) B_i^n(u(t)), \end{aligned} \tag{4.4}$$

where B_i^n is the i-th Bézier basis function of degree n and P_{ij} are the control points of S.

The Bézier curve-surface composition is now narrowed to the problem of computing the composition of $B_i^n(c(t))$, where $c(t)$ is a scalar curve. Assuming one can compute and represent the composition $B_i^n(c(t))$, $c(t) \in [u_{min}, u_{max}]$, as a curve, the curve $S(u(t), v(t))$ is also representable because it involves sums and

products of polynomial $B_i^n(c(t))$ terms only, which is polynomial (See, for example, [11] for the product of Bézier basis functions). But now we have,

$$\begin{aligned} B_i^n(c(t)) &= \binom{n}{i}(1.0 - c(t))^{n-i}(c(t))^i \\ &= \binom{n}{i}\left(1.0 - \sum_{k=0}^{o} Q_k B_k^o(t)\right)^{n-i}\left(\sum_{k=0}^{o} Q_k B_k^o(t)\right)^i, \end{aligned} \tag{4.5}$$

which is again reduced to sums and products of polynomial $B_k^o(t)$ terms only. See [8] for more on the computation of Bézier composition.

4.1.3 Shading Computation

As a final step, the text symbols must be properly shaded, taking into account the local shading information of the three dimensional surface. In traditional photo-realistic rendering techniques, the pixel is the smallest picture element considered. Each pixel in the rendered image is assigned some color value that corresponds to the intensity level sampled and computed for that pixel by some shading model. The shaders typically take into account the surface orientation and the view direction as well as the color and the direction or position of the light source(s).

Emulating this traditional rendering process, we evaluate a single intensity level $\mathcal{I}$ for each visible symbol. $\mathcal{I}$ is derived based upon the shading intensity computed on the surface at the center location of the symbol on that surface. The shading intensity could be derived from one of many shaders that are employed in computer graphics. This shading computation could also be computed in colors employing some color basis such as the RGB and deriving $\mathcal{I}$ as a vector in this RGB space. The magnitude of $\mathcal{I}$ directly controls the width of the symbol. Some examples of shaders are now considered. The cosine shader is one of the most commonly used in computer graphics:

$$\begin{aligned} \mathcal{I}_c(u,v) = \omega(&I_a + \\ &I_d \langle N(u,v), L\rangle + \\ &I_s \langle R(u,v), V\rangle^\alpha); \end{aligned} \tag{4.6}$$

where

$$\begin{aligned} R(u,v) =& \frac{\bar{N}(u,v) - L}{\left\|\bar{N}(u,v) - L\right\|}; \\ \bar{N}(u,v) =& 2N(u,v)\,\langle N(u,v), L\rangle\,. \end{aligned} \tag{4.7}$$

and (u,v) are the coordinates of the center of the bounding box of the symbol in the parametric domain of $S(u,v)$, ω provides some bias, L is the unit vector toward the light source, V stands for the unit

vector toward the viewer, R is the direction of the reflected light, and I_a, I_d and I_s are the coefficients for the ambient, diffuse and specular components of the illumination. Moreover, by letting the different I components be color vectors, one can support color shading as well (see Figure 10).

One could clearly employ other shaders as well. Two examples of such additional shaders are a *silhouette enhancement shader* and a *distance dependent shader*. The intensity level could be computed for these two shaders in the following way:

$$\mathcal{I}_s(u,v) = \omega\,(1.0 - \langle V, N(u,v)\rangle^{\gamma})\,; \tag{4.8}$$

$$\mathcal{I}_d(u,v) = \omega \frac{\mathcal{I}_c(u,v)}{\|L - S(u,v)\|^{\gamma}}, \tag{4.9}$$

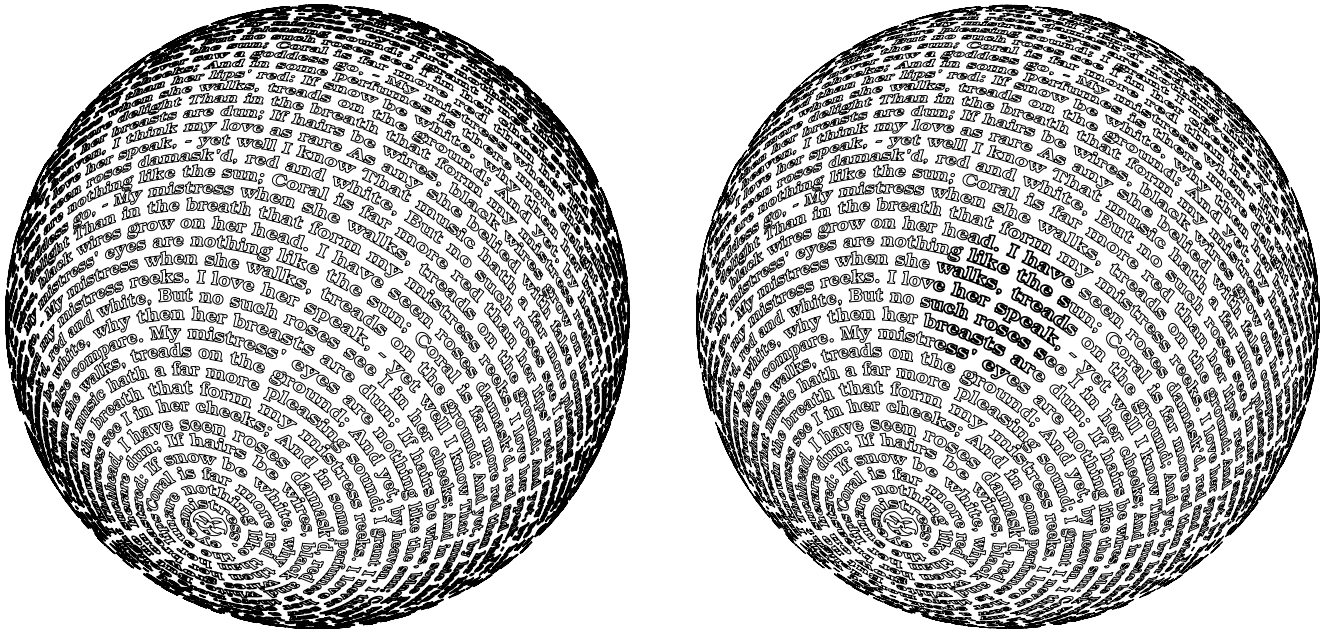
where $\mathcal{I}_s(u,v)$ is a shader that enhances silhouette areas and $\mathcal{I}_d(u,v)$ is a distance dependent shader, ω provides bias control as before and γ is a decay factor. $\mathcal{I}_c(u,v)$ is defined by Equation (4.6). The silhouette shader emphasizes the silhouette areas of the surface, while the distance dependent shader is similar to traditional rendering, but assigns larger intensity to the surface regions (symbols) that are closer to the point light source.

4.1.4 Surface Singularities

The presented scheme supports arbitrary parametric surfaces. Moreover, the text is laid out along isoparametric curves, taking into consideration the arc-length of the isoparametric curves as well as the distances between adjacent isoparametric curves.

Surfaces that are non regular at certain points have, by definition, a vanishing normal field at these locations. This singularity pose difficulties only at the shading computation stages, a problem that is shared by all surface rendering schemes. One can approximate the normal at non regular locations, using near by regular locations. Alternatively, but more difficult, one could compute the normal by exploiting the intrinsic geometry of the surface, overcoming the irregularity of the parameterization.

Other potential difficulties might stem from surfaces of varying arc-length along isoparametric curves and/or surfaces of varying distance between isoparametric curves. Varying arc-length along isoparametric curves is taken care of by the proper placement of the symbols along the curve (See Equation 4.2)). Nonetheless, the presented algorithm does assume a moderate change of distance between adjacent isoparametric curves, sampling only the middle of the domain (Equation 4.1)). Large changes in the distance between two isoparametric curves could yield uneven distribution of symbols between two adjacent lines of text. This uneven distribution could be partially compensated for by coercing the symbols near large gap to be wider or darker, employing anti-aliasing ideas from [17].



(a) (b)

Figure 6: An example of the precise layout of text over the sphere, that is rendered by enhancing silhouette areas in ($\mathcal{I}_s$, see Equation (4.8)) (a) and using traditional cosine shading ($\mathcal{I}_c$, see Equation (4.6)) in (b).

5 Examples

Before presenting some examples, we briefly discuss the hidden symbol removal process that was taken in this work. The hidden symbols on the hidden surface regions were removed in all examples with the aid of a z-depth map created using a polygonal approximation of the surface and the Open GL library, using an image of resolution 500x500. A polygonal approximation of the surface(s) is rendered into a regular z-buffer. The z-depth of the center of each symbol is then compared with the corresponding depth of the z-buffer at that center location. A symbol is visible (hidden) if the center of the symbol is found visible (hidden) by this z-depth test. No partial symbols are displayed.

We have used the text of the sonnet number $CXXX$ by William Shakespeare (see [23]) for rendering all the surfaces. *Precise* composition between the linear or cubic Bézier curves of the text and a Bézier surface is computed throughout the presented examples. Nevertheless, the order of the resulting composed curves is usually higher than three (cubic). Type 1 PostScript language supports either linear, or cubic Bézier curves. Thus the geometry of the resulting higher order text is approximated by a set of either linear or cubic Bézier curves to an arbitrary precision [6], in order to further use the composed letters in the PostScript representation. In the examples presented in this section, we have used linear approximation of the text geometry.

We start with a simple example that emphasizes the properties of the shading model, using a sphere (see Figure 6). A shader that enhances silhouette areas can be seen in Figure 6 (a). The highlight spot of the reflected light is clearly seen on the sphere of Figure 6 (b).

A classic model in the computer graphics field is the Utah teapot. In Figure 7, two examples of rendering text over the surfaces of the Utah teapot using the standard cosine shading model, that was presented in Equation (4.6), are shown. In this work, we have used the maximal width of the outline of the symbol if this symbol has the maximal intensity level (Figure 7 (a)). This is an appropriate way when one uses black color for the symbols rendering and white background for non illuminated areas. Though it could be natural to look at the negative image, i.e. the darker areas for the background and the bright ones for the illuminated parts of the picture. One option to express the negative colors without altering the background is to use minimal outline width for the illuminated symbols, having the symbols with the smaller intensity level producing the outline with the maximal width (see Figure 7 (b)).

In Figure 8, the infamous Utah teapot model is rendered again using a shader that enhances silhouette areas in the object, following the shader of Equation (4.8). The square region of the image in Figure 8 (a) is enlarged in Figure 8 (b), better exhibiting the individual symbols.

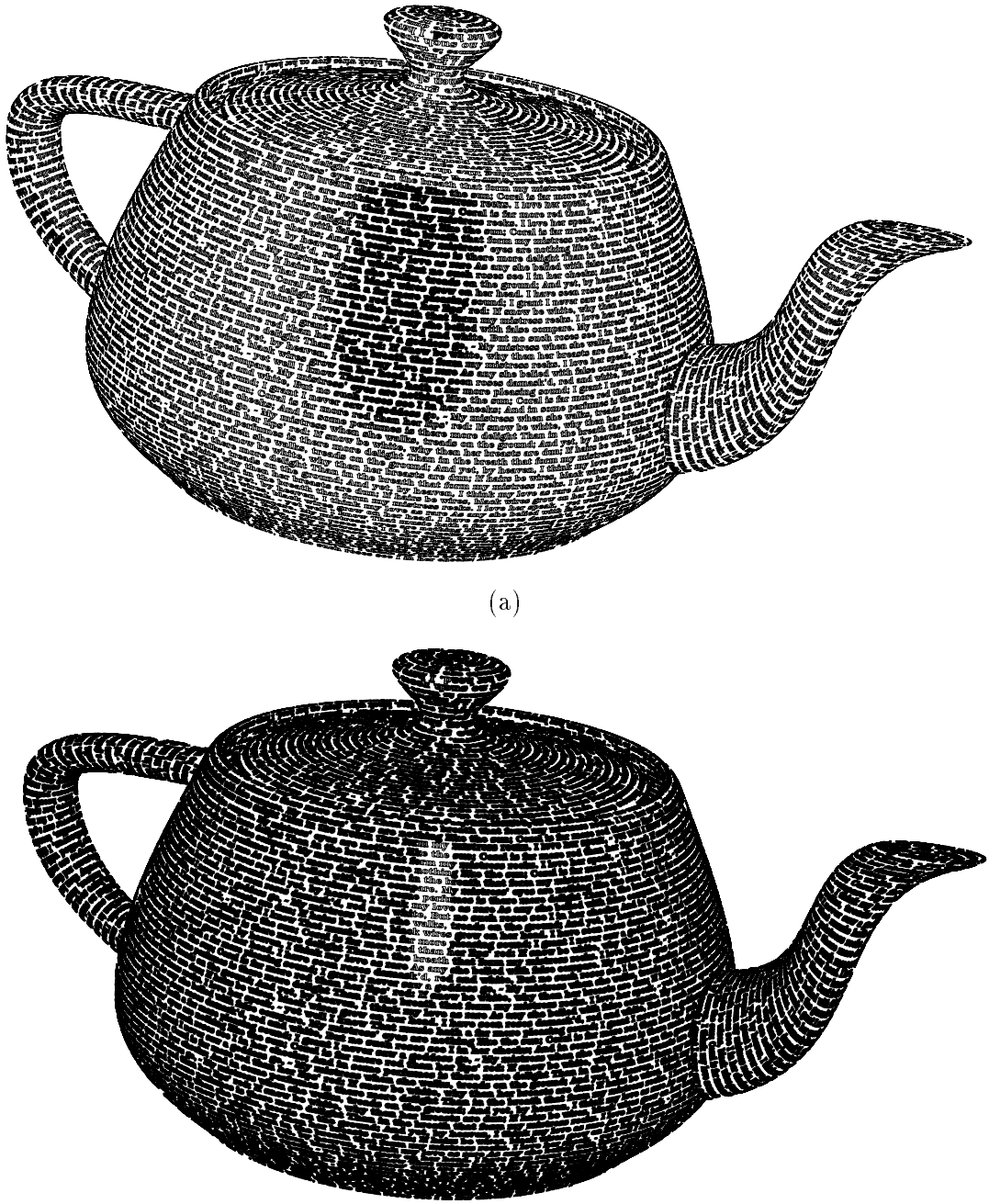
In Figure 9, two more examples of the application of different shaders are presented. The silhouette shader has been applied to the pawn surface in Figure 9 (a), while the distance dependent shader was used for a glass surface in Figure 9 (b).

Figure 10 presents color enhanced cosine shader examples of two different models. As in Figure 8 the square image region in Figure 10 (a) is enlarged in (b) for better inspection. In Figure 10 (c), the same shader is used for rendering the pawn surface.

It is much easier for the human eye to comprehend the picture when darker colors are used for non illuminated areas. Compare Figure 7 with Figure 11. These two pictures have identical geometry while the intensities of Figure 11 are negated with the respect to those in Figure 7.

The quality of the font deformation could be closely examined in Figure 12, where the text on the Utah teapot surface is enlarged and easily readable.

All these examples have been generated on A PC AMD Athlon 1.2GHz system running Windows 2000. The computation times for these examples are given in Table 1. Interestingly enough, all the Teapot renderings of same font size (Figures 7 (a), 7 (b), 8 (a), 10 (a), 11) show up with identical computational times, regardless of the shading model employed. This hints on the negligible cost of the shading computation compared to the overwhole computations.



(a)

(b)

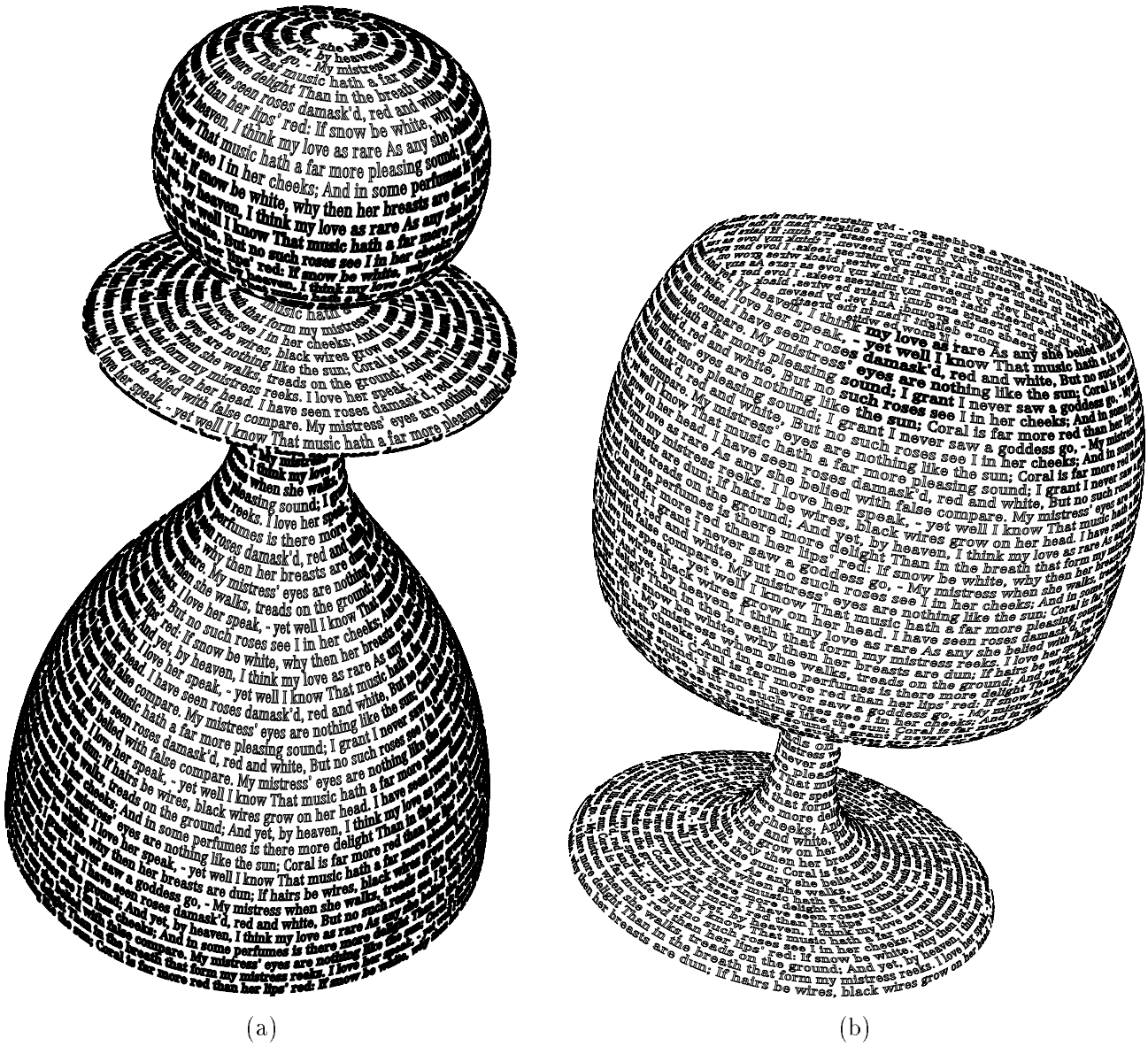
Figure 7: A standard cosine shader ($\mathcal{I}_c$, see Equation (4.6)) applied to the model of the Utah teapot. The illumination intensity defines the width of the outline of each symbol. In (a), the widest parts correspond to the highest intensity levels, while in (b), the lowest intensity levels are associated with the minimal outline width.

(a) (b)

Figure 8: An example of a rendering with a shader that enhances silhouette areas ($\mathcal{I}_s$, see Equation (4.8)). (b) presents the enlarged square region of (a).

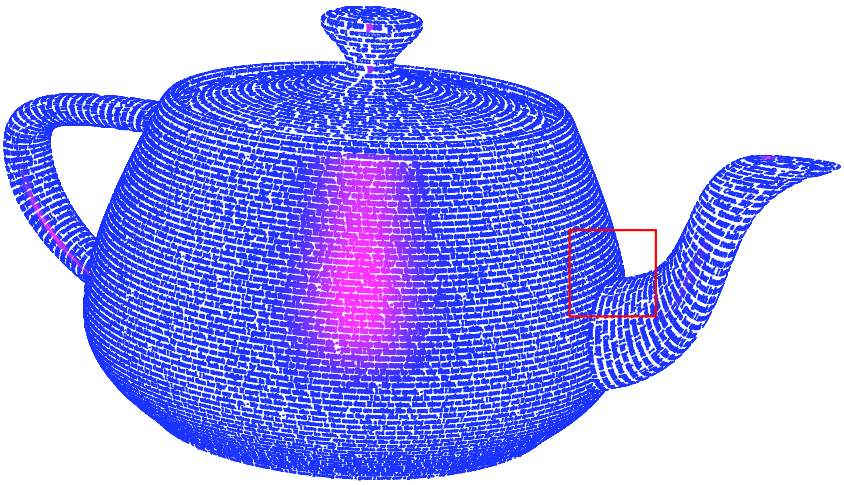
Figure	**Computation time**
6 (a)	21 sec.
6 (b)	21 sec.
7 (a)	2 min. 11 sec.
7 (b)	2 min. 11 sec.
8 (a)	2 min. 11 sec.
9 (a)	47sec.
9 (b)	33 sec.
10 (a)	2 min. 11 sec.
10 (c)	47 sec.
11	2 min. 11 sec.
12	26 sec.

Table 1: Running times (AMD Athlon 1.2GHz running Windows 2000) for the given examples.

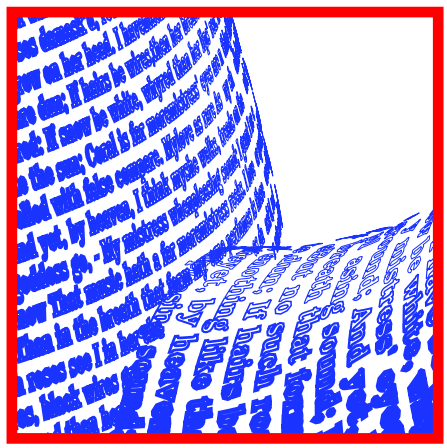


(a) (b)

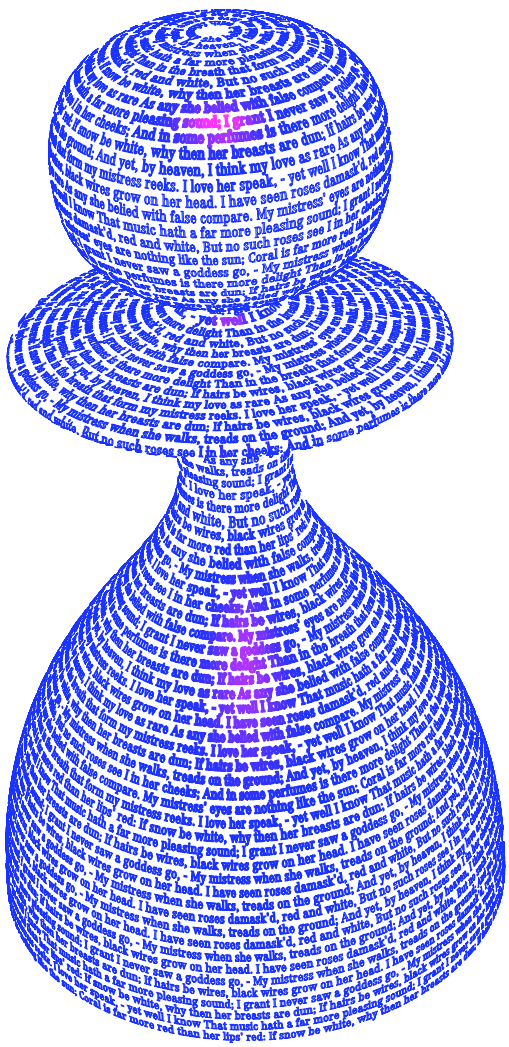
Figure 9: Examples of the silhouette shader ($\mathcal{I}_s$, see Equation (4.8)) application on a pawn (a) and the distance dependent shader ($\mathcal{I}_d$, see Equation (4.9)) application on a glass (b).



(a)



(b)



(c)

Figure 10: Examples of the color enhanced cosine shader application on a Utah teapot on (a) and (b) and a pawn (c). (b) presents the enlarged square region of (a). The specular component adds the red color to otherwise blue surfaces.

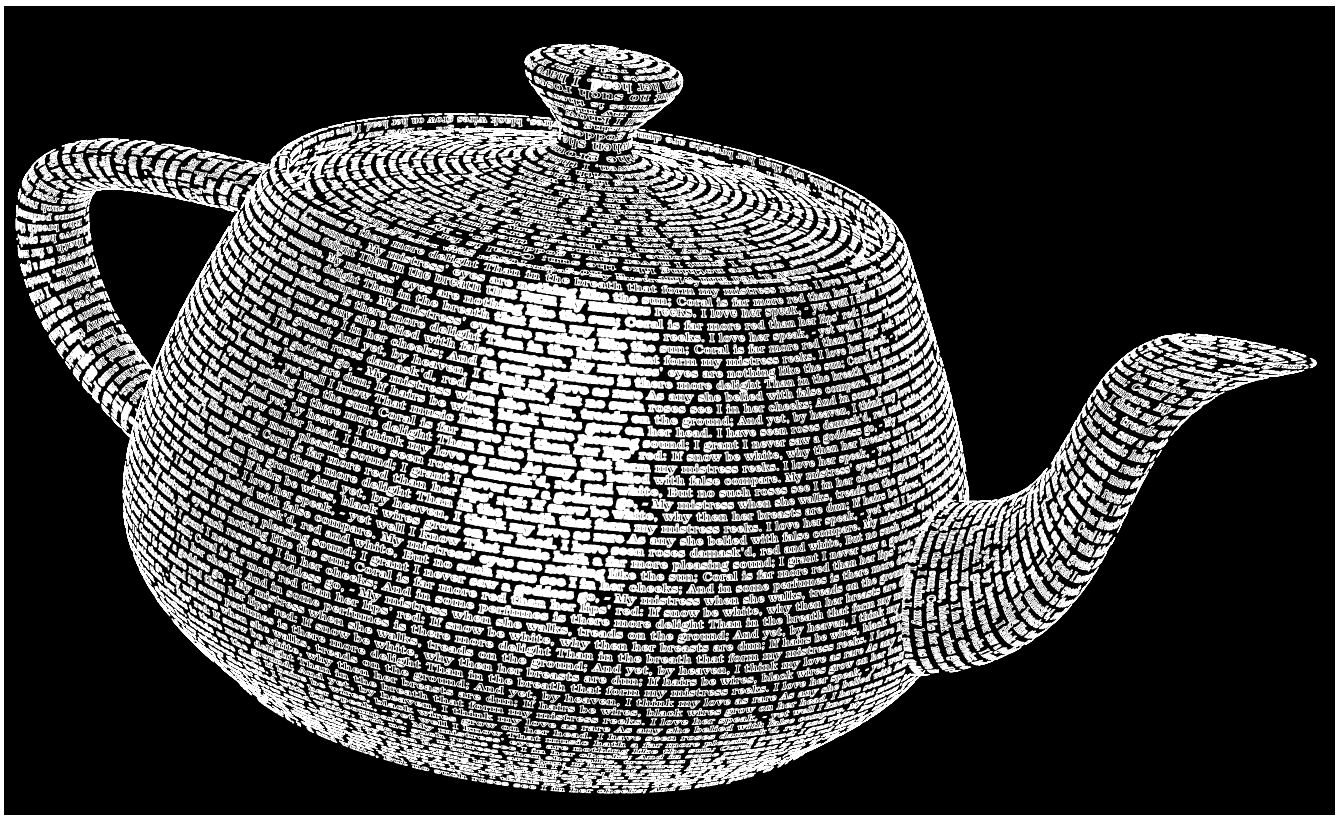


Figure 11: The negated version of a Utah teapot example of Figure 7 (a).

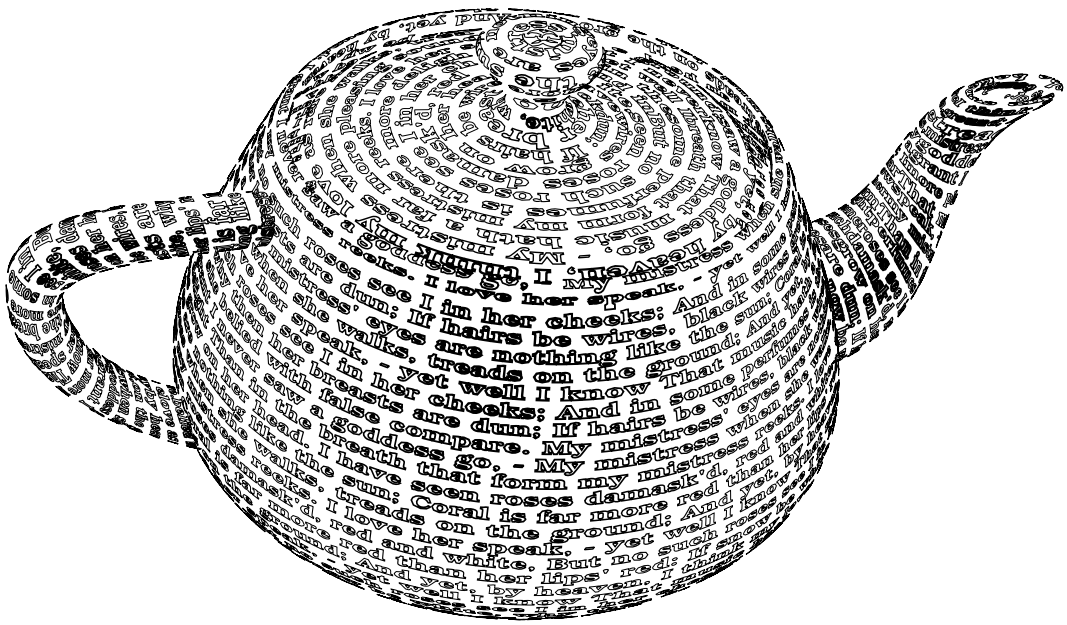


Figure 12: The low resolution version of a Utah teapot example of Figure 7 (a). Note the precise shape of the deformed symbols.

6 Conclusions

In this work, a text based NPR method for free-form parametric surfaces has been presented. The technique exploits a layout of half-toned text over the given surface. The key step of the method is a symbolic composition between the given three dimensional free-form parametric surface $S(u, v)$ and the geometry of the text that is represented by set of Bézier curves $\{C_i(t)\}_i$. The shading method may be arbitrary and is applied to the text on the surface by varying the width and the color of the symbols.

This work could be extended in several ways. One could attempt to use solid filled text on the surface instead of text outline. In order to enhance the illuminated parts of the surface, we changed the width of the outline of the text. If a layout of a filled text on the surface is used for rendering, then the width or area of the filled symbols must be modified to reproduce the illuminated regions.

The extension of the presented method to non-parametric surfaces such as polygonal meshes should also be investigated. Our algorithm needs a parametric surface to evaluate the composed letter. Moreover, isoparametric curves are employed as the letter-path to place the letters along. Yet, the letter-path placement need not be isoparametric. Any devised path that *covers* the surface could be exploited much like a tool path generation in NC machining applications. A projection of a fixed, planar, predefine path is a common practice in NC machining that could be employed for parametric surfaces as well as other surface types, including polygonal meshes. The reconstruction of the composed letter on the surface shape is a more difficult issue to be investigated. If only linear transformations are to be considered, one can employ a discrete approximation of the first fundamental form of the polygonal mesh at the sample point to warp the symbol.

Aliasing effects could result from different symbol shapes and different distances of adjacent symbols as discussed in Section 4.1.4. In [17], a method to alleviate these effects is described. The dither matrix equilibration algorithm of [17] corrects for uneven spacing of letters by affecting their intensity levels, a scheme that should be adapted to the rendering method presented in this work.

Acknowledgments

We would like to express our gratitude to the anonymous reviewers for their helpful comments.

References

[1] *PostScript Language Tutorial and Cookbook*, Adobe Systems Incorporated, Addison-Wesley Publishing Company, 1989.

[2] *The True Type font format specification*, Microsoft Corporation, 1990.

[3] *Adobe type 1 font format: multiple master extensions*, Adobe Developer Support, 1992.

[4] *PostScript language reference manual*, Adobe Systems Incorporated. Addison-Wesley Publisher Company, second ed., November 1994.

[5] K. Critchlow, *Islamic Patterns*, Thames & Hudson, 1989.

[6] G. Elber, *Free form surface analysis using a hybrid of symbolic and numeric computation.*, PhD thesis, Computer Science Dept., University of Utah, 1992.

[7] G. Elber, *Line art rendering via a coverage of isoparametric curves*, IEEE Transactions on Visualization and Computer Graphics, 1 (1995), pp. 231 – 239.

[8] G. Elber, *Symbolic and numeric computation in curve interrogation*, Computer Graphics forum, 14 (1995), pp. 25 – 34.

[9] G. Elber, *Line art illustrations of parametric and implicit forms*, IEEE Transaction on Visualization and Computer Graphics, 4 (1998), pp. 1 – 11.

[10] G. Elber, *Interactive line art rendering of freeform surfaces*, Computer Graphics Forum (EUROGRAPHICS'99 Proceedings), (1999), pp. 1 – 12.

[11] G. Farin, *Curves and Surfaces for Computer Aided Geometric Design, a Practical Guide.*, Academic Press, Inc., fourth ed., 1997.

[12] P. Hasluck, *Manual of Traditional Wood Carving*, New York: Dover Publications, 1977.

[13] A. Hertzmann and D. Zorin, *Illustrating smooth surfaces*, Computer Graphics (SIGGRAPH'2000 Proceedings), (2000), pp. 517–526.

[14] S. C. Hsu and I. H. H. Lee, *Drawing and animation using skeletal strokes*, Computer Graphics (SIGGRAPH'94 Proceedings), (1994), pp. 24–29.

[15] V. Ostromoukhov, *Digital engraving*, Computer Graphics (SIGGRAPH'99 Proceedings), (1999), pp. 417 – 424.

[16] V. Ostromoukhov and R. D. Hersch, *Artistic screening*, Computer Graphics (SIGGRAPH'95 Proceedings), (1995), pp. 219 – 228.

[17] V. Ostromoukhov and R. D. Hersch, *Multi-color and artistic dithering*, Computer Graphics (SIGGRAPH'99 Proceedings), (1999), pp. 425 – 432.

[18] Y. Pnueli and A. Bruckstein, *Digidurer — a digital engraving system*, The Visual Computer, 10 (1994), pp. 277 – 292.

[19] C. Rössel and L. Kobelt, *Line-art rendering of 3d-models*, Pacific Graphics 2000 Proceedings, (2000), pp. 87 – 96.

[20] C. Rössel, L. Kobelt, and H. P. Seidel, *Line-art rendering of triangulated surfaces using discrete lines of curvature*, WSCG'2000 Proceedings, (2000), pp. 168 – 175.

[21] M. P. Salisbury, M. T. Wong, J. F. Huges, and D. H. Salesin, *Orientable textures for image-based pen-and-ink illustration*, Computer Graphics (SIGGRAPH'97 Proceedings), (1997), pp. 401 – 406.

[22] T. W. Sedergerg and S. R. Parry, *Free-form deformation of solid geometric models*, Computer Graphics (SIGGRAPH'86 Proceedings), 20 (1986), pp. 151 – 160.

[23] W. Shakespear, *The complete works*, Gramercy Books, 1990.

[24] T. Surazhsky and G. Elber, *Arbitrary precise orientation specification for layout of text*, Pacific Graphics 2000 Proceedings, (2000), pp. 80 – 86.

[25] G. Winkenbach and D. H. Salesin, *Rendering parametric surfaces in pen-and-ink*, Computer Graphics (SIGGRAPH'96 Proceedings), (1996), pp. 469 – 476.