Adaptive Isocurves Based Rendering: the Hardware Way *

Gershon Elber
Department of Computer Science
Technion, Israel Institute of Technology
Haifa 32000
Israel
Email: gershon@cs.technion.ac.il

Abstract

In a recent work [Elber 92a], an almost optimal algorithm to provide a coverage based on the isoparametric curves of a surface was presented. This approach was combined successfully with curve rendering techniques and used to directly render surfaces using isoparametric curves instead of polygons.

In this paper, we describes an adaptation of the rendering algorithm that uses adaptive isoparametric curves as the surface coverage, to a generic hardware. We also discuss the feasibility of implementing the adaptive isocurve extraction algorithm itself in hardware. The presented results make the surface coverage using adaptive extraction of isoparametric curves a tool for competitive freeform surface rendering in both software and hardware.

Several results, including a videotape recording of a real time display, are demonstrated.

Key Words: Hardware Rendering, Surface Coverage, Direct Surface Rendering.

1 Introduction

Traditionally, surfaces are rendered using a polygonal approximation. Rendering of linear primitives, as polygons are, is usually more efficient and numerically more robust than direct rendering of polynomial and piecewise polynomial surfaces. Unfortunately, the polygonal model is only an approximation to the real surface and aliasing usually occurs.

Intensity (Gouraud) and normal (Phong) interpolation schemes [Foley 90] were developed almost two decades ago, in order to alleviate the visual effects of the plane discontinuities between the polygons that approximate the surface. Although the number of polygons in the approximation can be increased to improve the appearance of the surface, this number grows quadratically with the accuracy. Further, the polygonal representation is only an approximation. Aliasing along boundary and silhouette curves, manifested as piecewise linear limiting curves, is not alleviated by the Gouraud and Phong interpolation schemes. Trimming surfaces is an increasingly common representation that attempts to circumvent the topological difficulties in the tensor product surface representation. Appropriately trimming a polygonal surface approximation according to the trimming curves of the surface is considered a difficult problem [Rockwood 89].

Polygonal approximation techniques with their inherited difficulties, the introduced aliasing and the large size of the resulting data set, are known for almost two decades. Considering the size of the polygonal approximation data that severely hinders real time display of complex scenes, it is perplexing how little these techniques have improved weighing how advanced the contemporary computer graphics field is.

On the other hand, direct surface rendering is appealing since some of the difficulties experienced with the polygonal approximation can be circumvented. Smaller data size is to be expected and aliasing is virtually

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

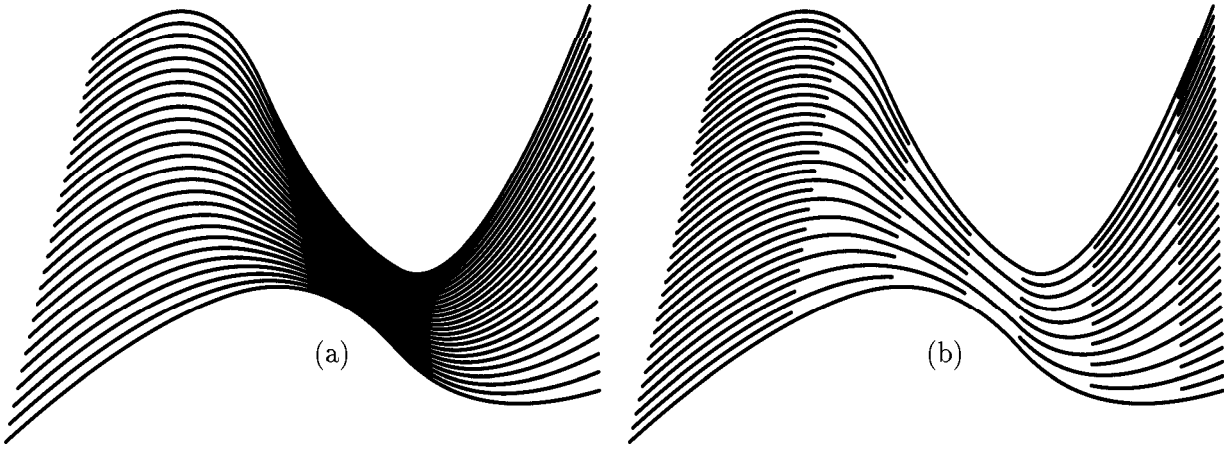

Figure 1: Isocurves are obviously not an optimal solution as a valid coverage for this surface (a). Adaptive isocurves are more optimal and their coverage is valid as well (b).

eliminated due to the exactness of the derived data. Unfortunately, direct surface rendering is complex and numerically difficult, making surface rendering using polygonal approximation the prevailing method of choice.

In recent years, several methods were developed to directly render curves and attempt to render surfaces using isoparametric curves [Chang 89, Klassen 91, Lien 87, Rappoport 91, Shantz 87, Shantz 88] or isocurves. Considering the number of pixels that are processed and the redundancy that occurs [Elber 92a], previous methods do not automatically extract an optimal or almost optimal set of isocurves from a surface S. Rendering an image of S using existing isocurve based methods cannot always guarantee to scan all the pixels in the representation of the surface in image space, and only those pixels. Further, none of the methods in [Chang 89, Klassen 91, Lien 87, Rappoport 91, Shantz 87, Shantz 88] provide an optimal or almost optimal solution, in the total length of the isocurves eliminating redundancies, to the surface *coverage* problem, a term that will be defined more rigorously shortly.

In [Rappoport 91], the optimality of the surface *coverage* is partially addressed using a heuristic subdivision based approach. The surface is subdivided each time the isocurve's spacing varies more than a specified tolerance, an approach which can lead to the rendering of a large number of small patches. The fixed initialization cost per curve when using the (adaptive) forward differencing algorithm [Chang 89, Klassen 91, Lien 87, Rappoport 91, Shantz 87, Shantz 88] would greatly increase the total cost of rendering that surface. In [Shantz 87], isocurves at equally spaced parametric intervals, $u = n\delta_u$, $n = 0, 1, 2 \cdots$, and δ_u is the step size, are used and which can lead to pixels either being missed or rendered more than once, as can be seen in the middle areas of Figure 1 (a) and Figure 2 (a). In [Shantz 88], the isocurves are adaptively spaced using bounds extracted from the convex hull of the distance function between two adjacent isocurves, $d(v) = f(u+\delta_u, v) - f(u, v)$. Each isocurve continues to span the entire v domain of the surface. Therefore, the redundancy demonstrated in Figures 1 (a) and 2 (a) would continue to reduce optimality in [Shantz 88].

In a related paper [Elber 92a], an algorithm that *adaptively* extracts isocurves and covers the entire surface in an almost optimal way was introduced (See Figures 1 (b) and 2 (b)). Toward this end we defined *surface coverage* using isocurves to be:

Definition 1 *A set of isocurves $\mathcal{C}$ of a given surface S is called a* valid coverage with respect to some constant δ *if for any point p on S there is a point, q, on one of the isocurves in $\mathcal{C}$, such that $\|p - q\|_2 < \delta$, where $\|\cdot\|_2$ denotes Euclidean distance.*

The polygonal primitive is replaced by an isocurve drawn with a finite thickness. The potential of the coverage algorithm of a freeform surface was proven for both freeform surface rendering [Elber 92a] and

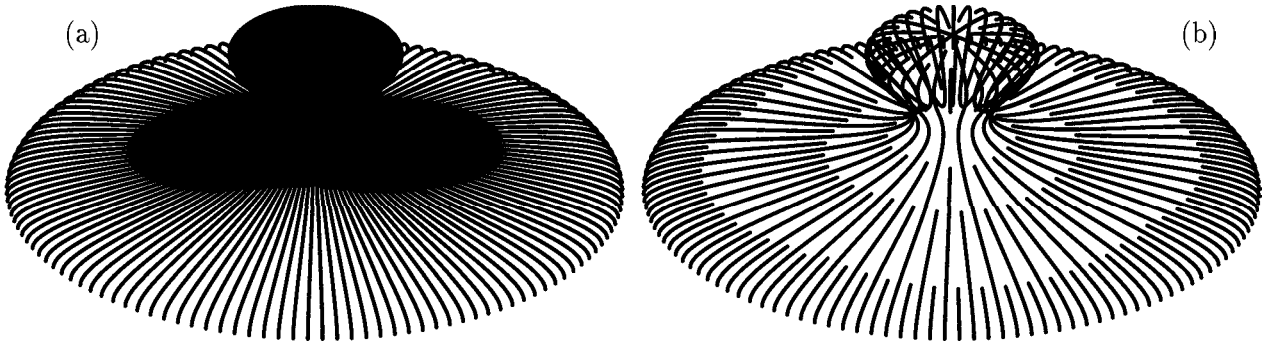

Figure 2: Utah teapot lid. Constant parameter spacing causes redundancy in coverage (a) mostly eliminated by adaptive extraction of isocurves (b). Both provide a valid coverage with respect to same δ.

for toolpath generation for machining of freeform surfaces [Elber 93]. In this paper, we present another application for the adaptive isocurve coverage algorithm and show that this algorithm is competitive in hardware surface rendering to rendering techniques that are based on polygonal surface approximations.

In section 2, we briefly present the ideas behind the adaptive isocurve coverage algorithm. In section 3, we discuss the hardware implementation that is based on the 4D series of Silicon Graphics and the support for hardware polyline rendering with color/normal interpolation. In section 4, we draw some conclusions from the reported results. The implementation of this algorithm is based on the IRIT [IRIT 93] solid modeler, developed at the Technion.

2 Adaptive Isocurves' Coverage Algorithm

Using isocurves as the coverage for a surface, we define *adjacency* and *iso-distance* between isocurves.

Definition 2 *Two (sub) isocurves of surface $S(u,v)$, $C_1(u) = S(u,v_1)$, $u \in [u_1^s, u_1^e]$ and $C_2(u) = S(u,v_2)$, $u \in [u_2^s, u_2^e]$, $v_1 \leq v_2$, from a given set $\mathcal{C}$ of isocurves forming a valid coverage for S are considered* adjacent *if, along their common domain $\mathcal{U} = [u_1^s, u_1^e] \cap [u_2^s, u_2^e]$, there is no other isocurve from $\mathcal{C}$ between them. That is, there does not exist $C_3(u) = S(u,v_3) \in \mathcal{C}$, $u \in [u_3^s, u_3^e]$ such that $v_1 \leq v_3 \leq v_2$ and $[u_3^s, u_3^e] \cap \mathcal{U} \neq \emptyset$.*

Definition 3 *The* iso-distance *function $\Delta_{12}(u)$ between two adjacent (sub) isocurves along their common domain $\mathcal{U}$ is equal to*

$$\begin{aligned} \Delta_{12}(u) &= \|C_1(u) - C_2(u)\|_2 \\ &= \sqrt{(c_1^x(u) - c_2^x(u))^2 + (c_1^y(u) - c_2^y(u))^2 + (c_1^z(u) - c_2^z(u))^2}. \end{aligned} \tag{1}$$

Given two adjacent isocurves, $C_1(u)$ and $C_2(u)$, on a polynomial Bézier or piecewise polynomial NURBs surface $S(u,v)$, one can symbolically compute and represent the square of the iso-distance, $\Delta_{12}^2(u)$, between them as a scalar field NURBs or Bézier curve [Elber 92b]. Computing the coefficients for the representation of $\Delta_{12}^2(u)$ requires the difference, sum, and product of curves, all computable and representable in the polynomial and piecewise polynomial domains [Elber 92b]. Furthermore, given some tolerance δ, it is possible to compute the parameter values where the iso-distance between $C_1(u)$ and $C_2(u)$ is exactly δ by computing the zero set of $(\Delta_{12}^2(u) - \delta^2)$ [Lane 81]. By subdividing the two curves at these parameters, new sub-isocurve

pairs, $\{C_1^i(u), C_2^i(u)\}$, are formed with the characteristic that each pair is always iso-distance smaller or always larger than δ, in their open interval domains. If the two curves in the pair $\{C_1^i(u), C_2^i(u)\}$ are closer than δ in the iso-distance metric then the Euclidean distance tolerance condition is met for that pair. That is, if $\Delta_{12}^2(u) < \delta^2$, $\forall u$, then the minimal Euclidean distance from point $p \in C_1(u)$ to $C_2(u)$ in not greater than δ, for all $p \in C_1(u)$. If, however, the two curves' iso-distance is larger than δ, a new sub-isocurve, $C_{12}^i(u)$, is introduced between $C_1^i(t)$ and $C_2^i(t)$ along their common domain $\mathcal{U}$ and the same iso-distance computation is recursively invoked for the two new pairs $\{C_1^i(u), C_{12}^i(u)\}$ and $\{C_{12}^i(u), C_2^i(u)\}$.

Starting with the two U boundaries or two V boundaries of the surface, the algorithm can invoke this iso-distance computation recursively and ensure two adjacent isocurves will always be closer than some specified distance δ by verifying that their iso-distance is not larger than δ. Since a middle isocurve is introduced iff the iso-distance is larger than δ and δ is small, resulting iso-distances between adjacent isocurves, as computed, are rarely less than $\frac{\delta}{2}$. Furthermore, since the resulting set of isocurves covers the entire surface S, it can serve as a valid coverage for S with distance δ.

Algorithm 1, the adaptive isocurve extraction algorithm, generates a valid and more optimal coverage by minimizing redundancies while providing a bound on the distance between two adjacent isocurves. More details and discussion on end conditions of this algorithm can be found in [Elber 92a].

3 The Hardware Implementation

By using isocurves instead of polygons, aliasing is greatly reduced. Unlike the polygonal approximation, isocurves are exact and the only aliasing that might occur is in rendering stage at the pixel level due to the discrete image representation. Because the polygonal or the isocurve approximation of the surface is precomputed, both methods may be massively parallelized by allocating a rendering processor for each polygon or curve. However, since polygons as two dimensional entities are more complex to deal with, the hardware needed to render a curve is simpler [Chang 89, Klassen 91, Lien 87, Shantz 87, Shantz 88] making it potentially more attractive.

In section 3.1, we discuss the isocurve rendering that was implemented using a generic polyline hardware renderer. In section 3.2 we discuss the possibility of implementing in hardware the surface coverage computation using the adaptive isocurve extraction algorithm.

3.1 Isocurve rendering using hardware

A dedicated curve rendering hardware required financial investment beyond our available resources, so it was decided to use generic hardware that supports polyline rendering with color (Gouraud) or normal (Phong) interpolation. Such support is available on VGX systems of the 4D series of Silicon Graphics.

By using a generic hardware, some compromises were made. Isocurves are no longer rendered directly, and instead are approximated into polylines. It is clear the such a concession can degrade the performance of the system and that real dedicated curve rendering hardware schemes, such as the ones proposed by [Chang 89, Klassen 91, Lien 87, Shantz 87, Shantz 88] can significantly improve the performance. Further, some aliasing is introduced because of the use of polyline approximation of curves. However, even with this generic hardware, rendering was found to be at a reasonable speed and with superior shading quality.

Figure 3 shows a quadratic by quartic Bézier surface. The accompanied video, from which Figure 3 is extracted, contains real time display of this surface using both traditional polygonal approximation rendering and adaptive isocurve based rendering. Quantitative comparison is difficult because the time to render the polygonal approximation of the surface depends on the number of polygons in the approximation.

The quality of the highlights, resulting from the surface reflecting the light of the light source, is superior and more continuous using the adaptive isocurve algorithm, as can be seen in the video.

Algorithm 1 Adaptive isocurve extraction. Iso-v curves are assumed.

Input:

Surface $S(u,v)$.

Iso-distance tolerance δ.

Output:

Adaptive isocurve coverage for $S(u,v)$.

Algorithm:

AdapIsoCurve($S(u,v)$, δ)
begin
  $C_1(u)$, $C_2(u) \Leftarrow$ two u boundary curves of $S(u,v)$.
  return $\{C_1(u)\} \bigcup$ **AdapIsoCurveAux**($S(u,v)$, δ, $\{C_1(u), C_2(u)\}$) $\bigcup$ $\{C_2(u)\}$.
end

AdapIsoCurveAux($S(u,v)$, δ, $\{C_1(u), C_2(u)\}$)
begin
  $\Delta^2_{12}(u) \Leftarrow \|C_1(u) - C_2(u)\|_2$, iso-distance between $C_1(u)$ and $C_2(u)$.
  if ($\Delta^2_{12}(u) < \delta^2$, $\forall u$) then
    return $\emptyset$.
  else if ($\Delta^2_{12}(u) > \delta^2$, $\forall u$) then
    begin
      $C_{12}(u) \Leftarrow$ Middle isocurve between $C_1(u)$ and $C_2(u)$.
      return **AdapIsoCurveAux**($S(u,v)$, δ, $\{C_1(u), C_{12}(u)\}$) $\bigcup$ $\{C_{12}(u)\}$ $\bigcup$ **AdapIsoCurveAux**($S(u,v)$, δ, $\{C_{12}(u), C_2(u)\}$).
    end
  else
    begin
      $\{C^i_1(u), C^i_2(u)\} \Leftarrow$ subdivided $\{C_1(u), C_2(u)\}$ at all u such that $\Delta^2_{12}(u) = \delta^2$.
      return $\bigcup_i$ **AdapIsoCurveAux**($S(u,v)$, δ, $\{C^i_1(u), C^i_2(u)\}$).
    end
end

Because the isocurves are exact, normal computation is accurate (up to the polyline approximation compromise taken herein). Hence, the shading computation is correct for every pixel in the image. This, in contrasts with the polygonal case where Gouraud interpolation is used in a polygonal based rendering. Given a surface $S(u,v)$, its unnormalized normal vector field, $n(u,v) = \frac{\partial S}{\partial u} \times \frac{\partial S}{\partial v}$, is computed symbolically [Elber 92b]. Every isocurve, $C(u)$, in the coverage of $S(u,v)$, is then escorted by a unnormalized normal vector field curve, $n(u)$, that is extracted from $n(u,v)$. At every location $C(u_0)$, the unit normal of the *surface* is equal to $\frac{n(u_0)}{\|n(u_0)\|}$. Both $C(u)$ and $n(u)$ are piped into the curve renderer to provide exact information on the curve and its normal, making it feasible to perform accurate shading computation, for every pixel.

Figure 3 contains snapshots from the polygonal and adaptive isocurve rendering, using the SGI's hardware. All hardware renderings were performed on a SGI 310, 33MHz R3000 and VGX graphics hardware.

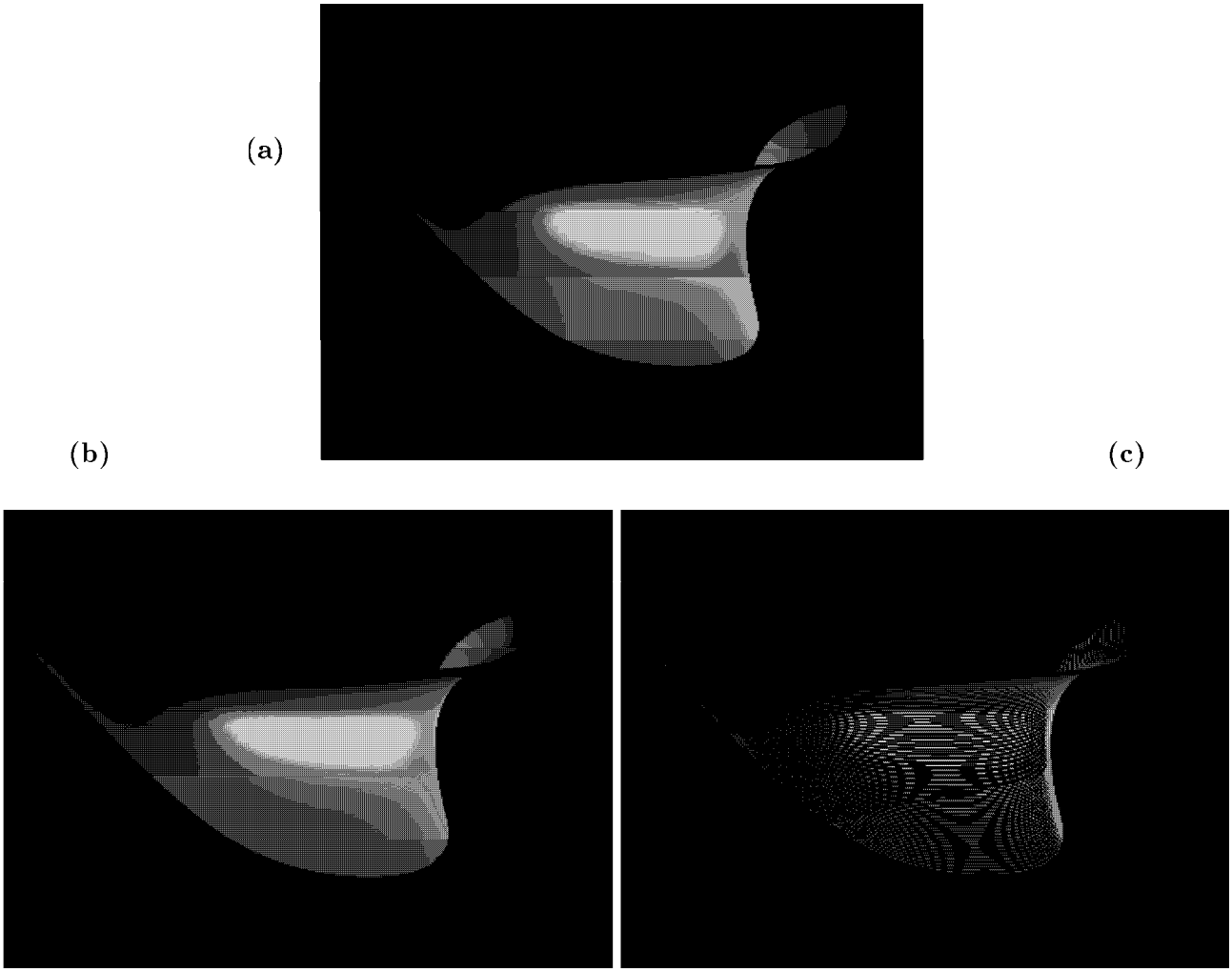

Figure 3: A quadratic by quartic Bézier surface rendered using SGI graphics hardware. Shown in (a) is the surface rendered with polygonal approximation. Adaptive isocurves coverage is used in (b) to render the surface, while in (c) a low density isocurve coverage is shown. These images were captured from the accompanied video.

The freeform surface rendering can be highly parallelized using the isocurve coverage since each isocurve primitive, as a one dimensional entity, needs a less complex scan converter compared to a two dimensional polygon primitive. If the number of hardware isocurve scan converters is smaller than the number of isocurves in the surface coverage, a rough shape of the surface can be rendered with fewer, less dense isocurve coverage. Thicker lines will be generated by each renderer to provide the necessary surface coverage.

Once the rough shape is rendered and displayed, a more refined and accurate shape can be rendered and displayed by rendering the isocurves at a continuously denser isocurve coverage level. Figure 4 shows some examples of this approach. The thick isolines rendering technique can also be used to provide faster display, compromising the quality of the rendering.

Figure 5 shows a more complex biquadratic NURBs surface with 48 patches (a patch is considered a polynomial bivariate, resulting from subdividing the NURBs surface at all its interior knots). The accompanied video also contains real time display of this surface using both traditional polygonal approximation rendering and almost real time display using adaptive isocurve based rendering. Figure 6 contains snapshots from the hardware rendering of the NURBs surface.

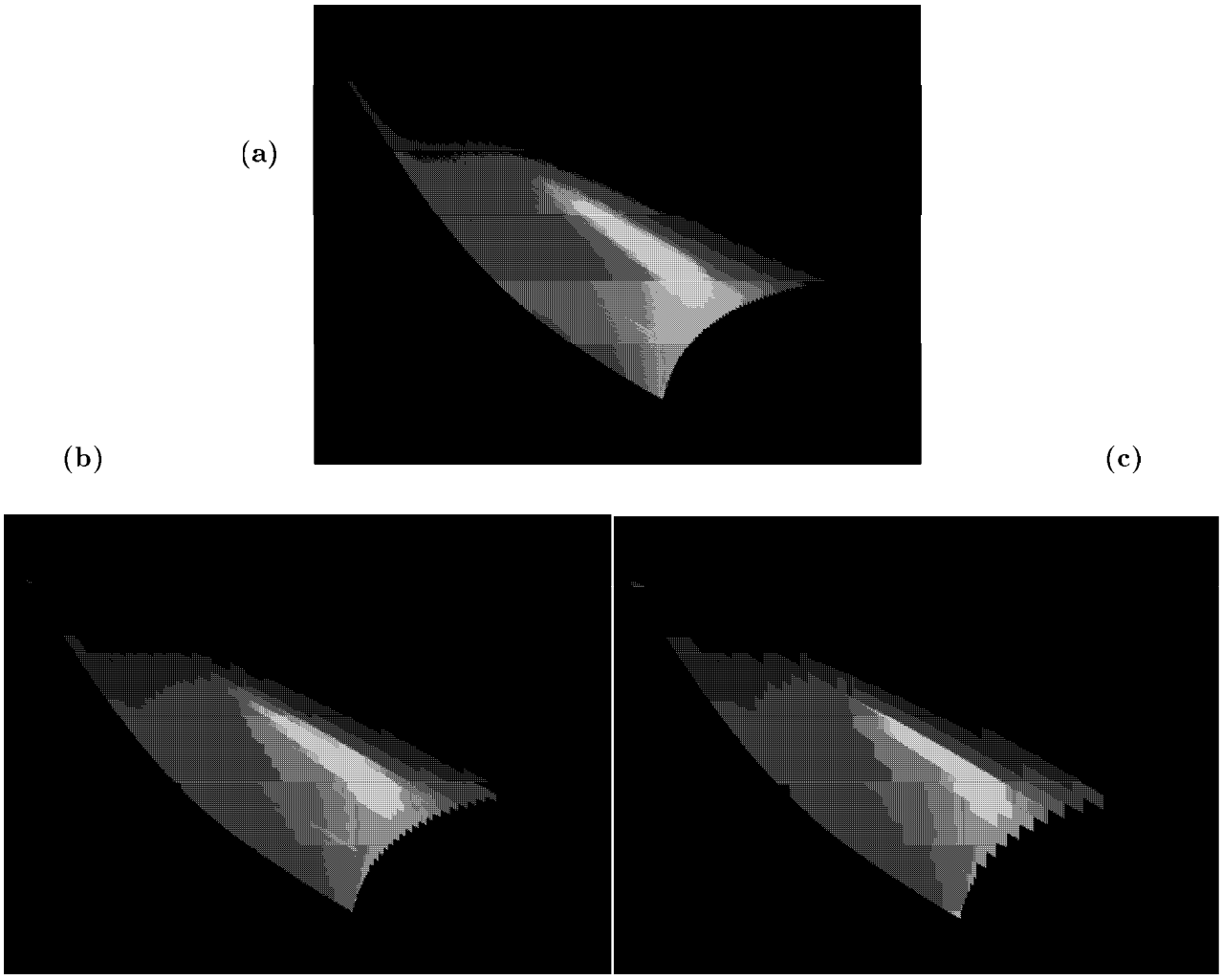

Figure 4: The Bézier surface of Figure 3 rendered using hardware, low tolerance surface coverage, and thick lines. Three levels of different line thicknesses and coverage tolerances are shown. These images were captured from the accompanied video.

3.2 Isocurve extraction using hardware

All surface rendering techniques reduce the complexity of the rendering problem by introducing intermediate and simpler representations, as polygons or as isoparametric curves. The intermediate representation is computed once but one could consider its implementation as part of a dedicated and complete hardware that gets freeform surfaces and processes them all the way to the display. Given a polynomial of fixed order, say cubic, the computation of equation (1) can be greatly simplified. The seven coefficients of the resulting degree six polynomial could be hard coded as sums and products of the four coefficients of each of the two adjacent cubic isoparametric curves.

Furthermore, the adaptive isocurves extraction algorithm extracted isocurves at parameter values which are powers of two fractions along the other parametric domain of the surface. For a domain of zero to one, the first level occurs at the parameter value $v = \frac{1}{2}$, the second level occurs at the parameter values $v = \left\{\frac{1}{4}, \frac{3}{4}\right\}$, the third level occurs at the parameter values $v = \left\{\frac{1}{8}, \frac{3}{8}, \frac{5}{8}, \frac{7}{8}\right\}$, and so forth.

The fixed parametric locations, at powers of two fractions, make it possible to precompute the blending coefficients of the control mesh that will extract isoparametric curves at parameter values of $\frac{i}{2^n}$, $0 \leq i \leq 2^n$. Assuming a bicubic surface and in order to select n, several assumptions must be made. Assume a single projected surface covering the entire screen spans W pixels. Assuming uniform distribution of isoparametric

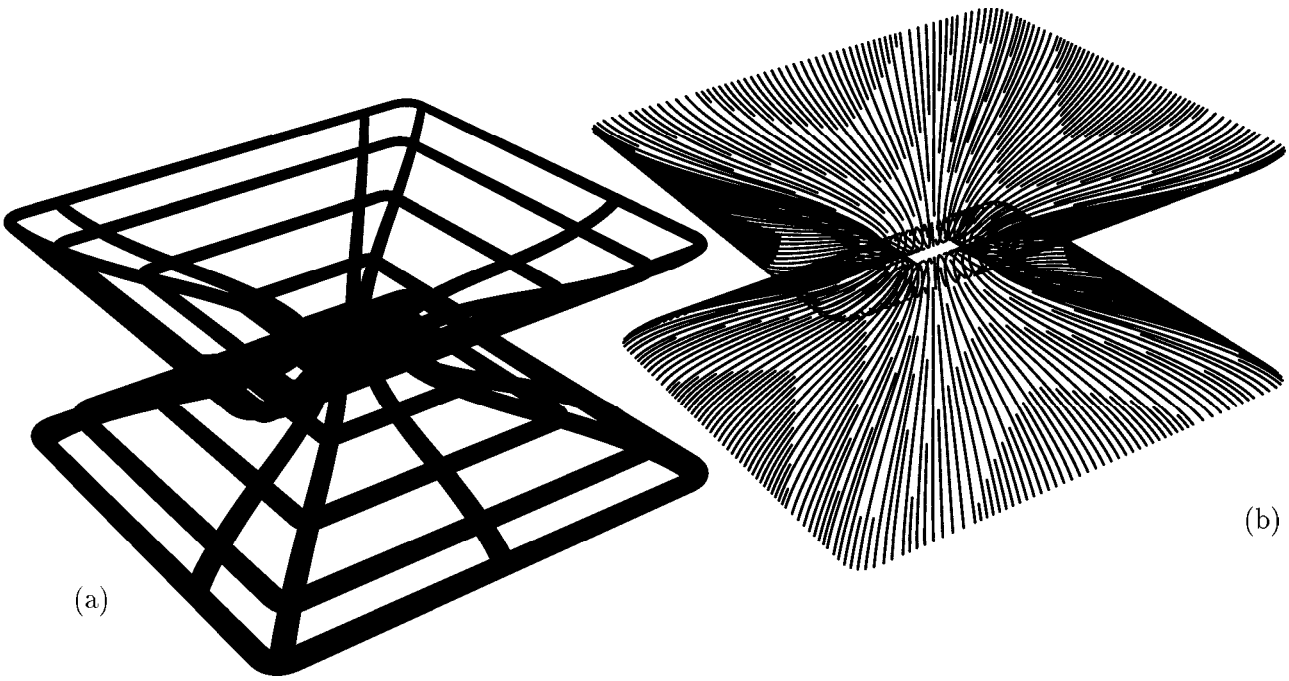

Figure 5: A connector surface. Shown is the surface (a) and a fine adaptive isocurve representation (b) for it, computed for the purpose of isocurve based rendering (see Figure 6).

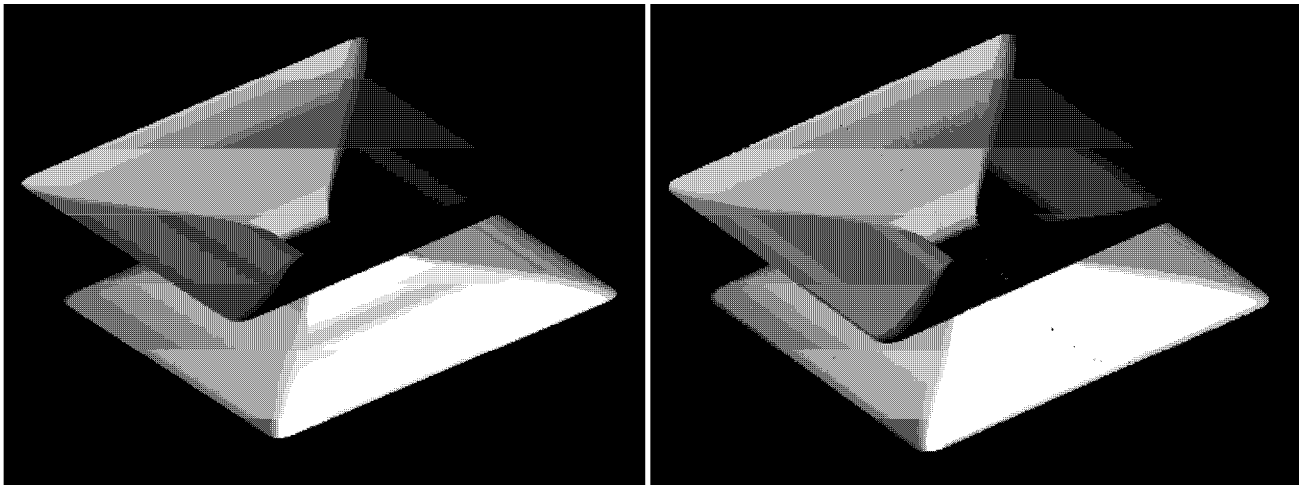

Figure 6: A connector surface rendered using hardware. Shown is the surface rendered using polygons (a) and using adaptive isocurves (b). These images were captured from the accompanied video.

curves and one isocurve per pixel, n must be of the order of $\log_2(W)$. In reality, projected surfaces are usually smaller than the screen size but the isoparametric curves have a non uniform distribution. An extreme control point distribution condition for the rendering of a cubic polynomial Bézier curve would occur for,

$$C(t) = \sum_{i=0}^{3} P_i B_i(t), \tag{2}$$

when $P_0 = P_1 = P_2 = 0$ and $P_3 = W$. Clearly $C(t) = Wt^3$ and $C(1) = P_3$. This control point distribution would lead to extreme non uniform speed along the cubic polynomial curve, with $C'(1) = 3W$ and $C'(t)|_{t\to 0} \to 0$. Notice the curve is not regular at $t = 0$. We therefore ask for which n does the following hold:

$$C\left(\frac{2^n - 1}{2^n}\right) - C(1) = W\left(\frac{2^n - 1}{2^n}\right)^3 - W \leq 1. \tag{3}$$

One can easily find that for a typical screen of size $W = 1024$ and $n \geq 12$, inequality (3) holds, making $n = 12$ a reasonable selection for a coverage of bicubic polynomial surfaces. A table of $2^{12} = 4096$ different blending coefficients would be sufficient to extract all the isocurves required for normal rendering.

The hardware implementation of only bicubic polynomial surfaces is not a restrictive assumption. The majority of surfaces created in the modeling process are either bicubic or biquadratic Bézier and NURBs surfaces. These representations can always be exactly and efficiently converted, using degree raising and subdivision [Farin 90], to bicubic polynomial surfaces. Further, rational curves and surfaces, as well as high order curves and surfaces can be efficiently approximated, for display purposes, by bicubic curves and surfaces [Elber 92b, Hoschek 87, Hoschek 89].

One part of the adaptive isocurve extraction algorithm cannot be implemented in hardware as easily. It is the computation of a constant set of the resulting polynomial in equation (1), a numerical iterative algorithm. The implementation of a fast approximation to the computation of a constant set that is feasible and efficient in hardware, is an open question.

4 Conclusion

We have shown that direct hardware rendering of surfaces using the isocurve surface coverage is competitive and feasible. The adaptive isocurve coverage can be used to directly render surfaces with virtually no aliasing compared to polygonal approximations. The direct surface rendering can be highly parallelized, by allocating a dedicated hardware renderer for every isocurve. The isocurve primitive is a one dimensional entity and therefore simpler to manipulate than a two dimensional polygon. Hence, the hardware necessary for rendering an isocurve primitive is potentially simpler than the one required for a polygon rendering.

We expect that a dedicated curve rendering hardware, such as the curve rendering methods proposed in [Chang 89, Lien 87, Shantz 87, Shantz 88], highly parallelized and combined with the adaptive isocurve algorithm can provide a fast and automatic way to directly render freeform surfaces with virtually no aliasing.

References

[Elber 92a] G. ELBER and E. COHEN. Adaptive Iso-Curves Based Rendering for Free Form Surfaces. Submitted for Publication. Also Technical Report UUCS-92-040, Department of Computer Science, Univeristy of Utah.

[Foley 90] J. D. FOLEY et al. Computer Graphics, Principles and Practice, Second Edition. Addison-Wesley Systems Programming Series, Jul. 1990.

[Rockwood 89] A. ROCKWOOD, K. HEATON, and T. DAVIS. Real-Time Rendering of Trimmed Surfaces. Computer Graphics, Vol. 23, Num. 3, pp. 107-116, Siggraph Jul. 1989.

[Chang 89] S. CHANG, M. SHANTZ and R. ROCCHETTI. Rendering Cubic Curves and Surfaces with Integer Adaptive Forward Differencing. Computer Graphics, Vol. 23, Num. 3, pp. 157-166, Siggraph Jul. 1989.

[Klassen 91] R. V. KLASSEN. Integer Forward Differencing of Cubic Polynomials: Analysis and Algorithms. ACM Transaction on Graphics, Vol. 10, Num. 2, pp. 152-181, Apr. 1991.

[Lien 87] S. LIEN, M. SHANTZ, and V. PRATT. Adaptive Forward Differencing for Rendering Curves and Surfaces. Computer Graphics, Vol. 21, Num. 4, pp. 111-118, Siggraph Jul. 1987.

[Rappoport 91] A. RAPPOPORT. Rendering Curves and Surfaces with Hybrid Subdivision and Forward Differencing. ACM Transaction on Graphics, Vol. 10, Num. 4, pp. 323-341, Oct. 1991.

[Shantz 87] M. SHANTZ and S. L. LIEN. Shading Bicubic Patches. Computer Graphics, Vol. 21, Num. 4, pp. 189-196, Siggraph Jul. 1987.

[Shantz 88] M. SHANTZ and S. CHANG. Rendering Trimmed NURBS with Adaptive Forward Differencing. Computer Graphics, Vol. 22, Num. 4, pp. 189-198, Siggraph Aug. 1988.

[Elber 93] G. ELBER and E. COHEN. Tool Path Generation for Freeform Surface Models. The Second ACM/IEEE Symposium on Solid Modeling and Applications, Montreal Canada, May 1993.

[IRIT 93] IRIT. Irit 4.0 User's Manual. October 1993.

[Elber 92b] G. ELBER. Free Form Surface Analysis using a Hybrid of Symbolic and Numeric Computation. Ph.D. thesis, University of Utah, Computer Science Department, 1992.

[Lane 81] J. M. LANE and R. F. RIESENFELD. Bounds on a Polynomial BIT 21 (1981), 112-117.

[Farin 90] G. FARIN. Curves and Surfaces for Computer Aided Geometric Design Academic Press, Inc. Second Edition 1990.

[Hoschek 87] J. HOSCHEK. Approximate Conversion of Spline Curves. Computer Aided Geometric Design 4, pp. 59-66, 1987.

[Hoschek 89] J. HOSCHEK, F. J. SCHNEIDER, and P. WASSUM. Optmial Approximate Conversion of Spline Surfaces. Computer Aided Geometric Design 6, pp. 293-306, 1989.