International Journal of Computer Information Systems and Industrial Management Applications (IJCISIM)
First Special Issue on "Computer Graphics and Geometric Modeling" (2008)
Selected Papers from CGGM'2007 Workshop

Improved Adaptive Antialiasing for Ray Tracing Implicit Surfaces

Jorge Flórez, Mateu Sbert, Miguel A. Sainz and Josep Vehí
Institut d'Informática i Aplicacions, Universitat de Girona
17071 Girona (Spain)

Abstract: **Interval arithmetic has been extensively used to create guaranteed intersection test when ray tracing implicit surfaces. Aliasing is one of the principal problems in ray tracing, but the interval algorithms are not designed to correct this situation. This paper shows a novel technique to create a guaranteed interval adaptive anti aliasing method (IAA) for interval ray tracing of implicit surfaces. The method is based in the study of the coherence of sets of neighboring rays in a pixel, to detect variations over the hit surface. If those variations are not small enough, the area of the pixel is subdivided and the process is started over the new sub pixels. The subdivision continues until the variations over the surface are small. This method allows us to obtain a better visualization than traditional interval ray tracing.**

I. Introduction

Ray tracing is a technique inspired in the behavior of the rays of light interacting with objects in nature. The rays bouncing on the objects arrive to our eyes and we can see the features of the surface (color, shiness, transparency, etc). Indirect light (light reflected from other objects) can also rebound at the surface and arrive to our eyes. Using this rendering technique, it is possible to obtain realistic visualization of different kinds of scenes.

Ray tracing has been extensively used to render parametric and implicit surfaces. However, there are some problematic situations in which thin features of the implicit surfaces are not correctly rendered. Many authors have reported the situation in which thin features "disappear" when the surface is ray traced [2, 10, 9].

This occurs because the computers can not guarantee the robustness of floating point operations: they can not represent the whole set of real numbers, instead they a set of machine numbers with limited precision which approximates the set of real numbers. Although double precision is used, some problematic surfaces are not correctly rendered.

Some authors have proposed reliable ray tracing algorithms that perform guaranteed intersection test between the rays and the surfaces, based on interval arithmetic [2, 4, 10]. The interval algorithms provide a guaranteed solution for the intersection test, but they do not propose a reliable way to reduce aliasing in the visualization of the surfaces.

Implicit surfaces could have very thin features that are impossible to intersect using rays without thickness. In a theoretical process, an infinite number of rays should be sent for every pixel. But it is only possible to send a representative number of rays (point sampling).

To obtain a completely reliable ray tracing process, the algorithms used to create guaranteed intersection test should be improved with a reliable anti aliasing strategy.

The Aliasing effects are related to the jagged edges that appear on the border of the surfaces. This is due to the high contrast between colors, for example with the border color and background color, or with the color of a shadow crossing a surface in a high illuminated area. Aliasing also occurs when small or thin parts of the surfaces seems to disappear during the visualization of the surface. The last kind of aliasing is more frequently in the ray tracing of complex implicit surfaces.

The guaranteed algorithms can be adapted to work with a regular number of rays per pixel, in which the value of the intensity is obtained from the average value of the sampled rays. However, small features could be missed producing jagged effects in the visualization [6].

Whitted [13] proposed a technique called adaptive sampling, which offers a better solution to this problem. In this technique, rays are traced in the corners of the pixel. If the values of the colors obtained through every point are too different, the pixel is subdivided and new rays are traced through the new corners. When the differences between ray values are less than a specified threshold, the average of the values is taken to shade the pixel.

Because it is still possible to miss thin parts of the surface, Whitted used bounding boxes for small objects. If the ray intersects a bounding box, the sampling rate is increased to guarantee that view rays do not miss the object. Although effective in most of the cases, this technique does not work very well with long thin objects [6].

Other approaches are based on gathering information of a continuous set of rays as cone tracing [1] or beam tracing [8]. The main disadvantage of these proposals is that they require computationally complex intersection tests.

This paper describes a method called interval adaptive anti aliasing (IAA)[5] which produces reliable anti aliased ray traced image of implicit surfaces. This method evaluates areas of the pixel instead of points as in point sampling. Interval arithmetic is used to guarantee that small parts of the surface seen through the pixel are not missed. The set of rays that cover an area of the pixel are treated as a unique beam. The information obtained from the beam (instead of independent rays) is evaluated to determine whether the area covered

by the rays presents much variation over the surface. In that case, the pixel is subdivided and the process continues until the variation is not visible.

This method does not require bounding boxes to detect small features as adaptive sampling does. Also, as the complexity of the intersection test is the same as the traditional interval ray tracing, the method can be easily implemented.

This paper has the following sections. Section 2 describes the work that other authors have developed to perform guaranteed ray tracing processes. Section 3 presents the IAA algorithm, including a method to trace a beam (instead of points). The experimentation and results of the algorithm are presented in section 4.

II. Related Work

The intersection between an implicit function and a ray is defined with the following equation:

$$f(t) = f(s_x+t(x_p-s_x), s_y+t(y_p-s_y), s_z+t(z_p-s_z)) = 0$$

where (s_x, s_y, s_z) is the origin or view point; (x_p, y_p, z_p) is the point in which the ray crosses the screen, and t indicates a magnitude in the direction of the ray. If the parameter t is replaced with an interval T, a set of real values can be evaluated instead of a unique value of t.

The reliability in the intersection test is then assured in the following form:

$$F(T) = F(s_x+T(x_p-s_x), s_y+T(y_p-s_y), s_z+T(z_p-s_z)) \quad (1)$$

in which T is an interval parameter that allows evaluation over interval sections of the ray. This new function is called inclusion function.

Mitchell [10] proposed a method to find the roots of the function in two steps. First, the parameter T is subdivided in intervals $[t_i, t_{i+1}]$. Every interval is evaluated with the inclusion function to find the first interval containing roots. As a second step, this interval is reduced until an accurate size is obtained by means of other numerical techniques. Capriani et al [2] proposed to solve both steps of Mitchell's algorithm using interval arithmetic, proving that the execution time is not increased.

San Juan-Estrada et al [12] used a technique based on a branch-and-bound algorithm to find the nearest intersection point. They defined the intersection test as a minimization problem where

$$t^* = min\{\ t : F(T) = 0,\ t \in \mathbf{R}\}$$

Other methods are intended to improve the efficiency in the interval ray tracing. For instance, Flórez et al [4] proposed a technique to perform a fast elimination of screen space regions without intersections between rays and the implicit surface.

Some authors have introduced other techniques, for example LG surfaces (based on Lipschitz constants)[9] and sphere tracing [7]. The downside of these methods is that they require severe constraints on the surfaces. Also, affine arithmetic has been used as a replacement of the traditional interval analysis in the intersection test [3].

III. Interval Adaptive Anti Aliasing (IAA)

In this paper, we propose an Adaptive Anti aliasing method based in Interval Arithmetic (IAA). This method requires a function to evaluate a beam of rays instead of individual rays. Using this function, a beam with an infinite number of rays crossing an area of the pixel can be evaluated to know if any of them intersects the surface. Knowing the variations within the whole pixel, the adaptive anti aliasing strategy can be established.

A. Evaluation of beams

To cover areas of the pixel instead of points, the real values of x and y of the screen must be considered as interval values in equation 1. A new inclusion function to take into account the new interval values can be defined as follows:

$$F(T) = F(s_x+T(X_p-s_x), s_y+T(Y_p-s_y), s_z+T(z_p-s_z)) \quad (2)$$

To perform the evaluation with equation 2, the intervals X_p and Y_p must be fixed to a range of values inside the pixel. This means that the inclusion function will include all the beams that cross the pixel through this area. The evaluation of the beam is performed in the same way as in classical interval ray tracing. This way, the method guarantees the reliability during the intersection test and also detects all features of the surface in the evaluated area.

Figure 1 shows the shape of the new inclusion function. The beam in the figure is a pyramid, like in other beam tracing processes.

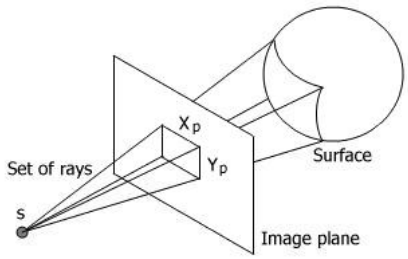

Figure. 1: Shape defined by the new inclusion function $F(X_p, Y_p, T)$

The value of the interval T increases the further the beam is traversed which means a size increase in X and Y intervals. If the beam does not intersect any part of the implicit surface, then the result of the evaluation of equation 2 does not contain zero ($0 \notin F(X_p, Y_p, T)$). In any other case, it is possible that one or more rays inside the beam intersects the surface. In that case, the parameter T must be bisected until the machine precision is achieved.

During the subdivision process, the range of values of the intersection points for the beam must be calculated. To do this, the values of T near the intersection of the beam are saved. When $0 \in F(X_p, Y_p, T)$ the minimum and maximum values of the current interval T are evaluated with the inclusion function. If $F(X_p, Y_p, T.Inf) > 0$, the minimum value of T is saved in a vector. If $F(X_p, Y_p, T.Sup) < 0$, the maximum value is saved in another vector. After that, the smaller of the positive points and the bigger of the negatives are taken to create the interval of the final value of T.

Function $EvalRaySet(X_p, Y_p, T)$
add T to List
While List has elements
T=First element of List
If width(T) reach machine_precision then Exit While
If $0 \in F(X_p, Y_p, T)$
If $F(X_p, Y_p, T.Inf) > 0$ add $T.Inf$ to ListInf
If $F(X_p, Y_p, T.Sup) < 0$ add $T.Sup$ to ListSup
Bisect T in $T1$ and $T2$
Add $T1$ and $T2$ to List
End if
Remove T from List
End While
If List has elements
Inf = smaller value of ListInf
Sup = bigger value of ListSup
return Interval(Inf,Sup)
else return EmptyInterval

Figure. 2: Algorithm for the intersection of the set of rays with the surface

The algorithm to perform the intersection test using a beam is presented in figure 2.

Figure 3 shows the result of the evaluation of the $EvalRaySet$ algorithm. If a part of the surface (even a small part) is inside the pyramid (Figure 3a), or crosses the evaluated area (Figure 3b), that part will never be missed. The same occurs when the beam is completely covered by the surface (Figure 3c). The result of the test is false only if the beam completely misses the surface (Figure 3d)

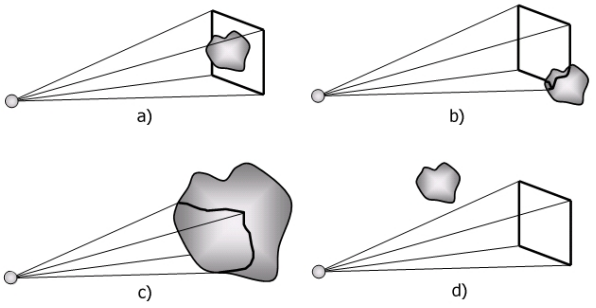

Figure. 3: Different situations encountered during the evaluation of the algorithm $EvalRaySet$

B. Adaptive Anti Aliasing

The IAA algorithm is performed by means of a subdivision process over the pixel. At the beginning of the process, the whole area of the pixel is evaluated to determine if the surface covers the pixel, at least in part. This occurs when the function $EvalRaySet$ returns non empty intervals. An empty interval indicates that the pixel does not have rays intersecting the surface.

The estimator we use to know the variations of the surface is the dot product between the view rays and the surface normals (Section 3.3 explains how to calculate the interval estimator). Although other estimators could be used, we chose this one because it is simple to implement and gives enough information about the variations of the curvature of the surface to detect regions with potential aliasing. If the width of the interval dot product between the set of rays and the normals is bigger than a predefined threshold, or if zero is not contained in the derivative of the equation for the for current value of T ($0 \notin F'(P.X, P.Y, T)$), the pixel is subdivided in four sections or sub pixels. After this, a set of rays is traced for every sub pixel and the process starts in every new area. When the estimator in a sub pixel is equal or less than the threshold, the average of the normals is used to calculate the Phong shading of the sub pixel. Also, the subdivision process continues until the area of the sub pixel is less or equal than another threshold. This threshold determines the number of samples allowed per pixel. In the examples of section 4, the maximum threshold allowed is $1/16$ of the initial area of the pixel.

Finally, the pixel is shaded using the average of the areas and the shading values of every sub pixel. The IAA algorithm is presented in figure 4.

Function IAA(Pixel)
add Pixel to PList
T=(0,∞)
While PList has elements
P=First element of List
delete P from PList
$TR = EvalRaySet(P.X, P.Y, T)$
If Not empty TR
If P <= area_threshold
calculate Phong shading for P
add shade value to ShadeList
Else
calculate estimator for P using TR
If accomplish conditions for estimator
calculate Phong shading for P
add shade value to ShadeList
Else
Subdivide Pixel in P1, P2, P3, P4
Add P1, P2, P3, P4 to PList
End if
End if
Else
shade P with background color
add shade value to ShadeList
End if
End While
Average shade values of ShadeList
(proportional to the areas) and assign it to Pixel

Figure. 4: Adaptive Anti Aliasing Algorithm based on Interval Arithmetic

Figure 5 shows the advantage of IAA over an adaptive anti aliasing method based on point sampling. When point sampling is used (Figure 5a), nothing is known about variations of the surface between neighboring rays. Using IAA (Figure 5b), the details inside the whole pixel are taken into account. In this case, the beam is subdivided because the surface has too much variation in the area covered by the beam. If in the new set of beams, the smaller and the bigger values of the estimators are too different, those beams must be subdivided

again.

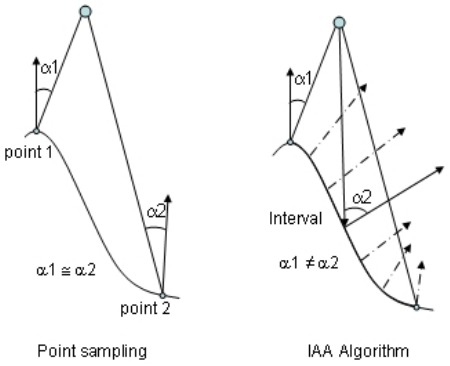

Figure. 5: Estimator using point sampling and IAA

The IAA algorithm can visualize features impossible to detect with ray tracing using point sampling. This is the case with the Teardrop surface (Figure 6). Even using a robust polygonal approximation of this surface, it is not possible to render the thin feature that joins the drop with the main body [11]. A Chub's surface (Figure 7) has many sections connected in just a few pixels which are correctly rendered using IAA algorithm.

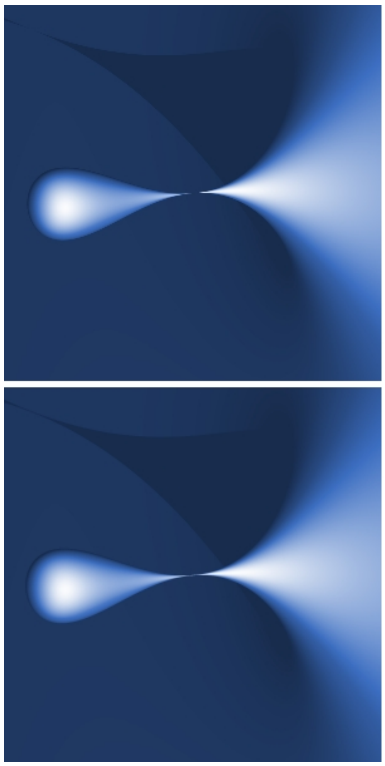

Figure. 6: A Teardrop surface rendered using Anti aliasing based in point sampling and Interval Arithmetic (top). The surface is generated in 26 minutes. Using IAA (bottom), the detail of the drop connecting to the main body is better visualized. The IAA version takes 17 minutes.

C. Calculation of the interval estimator

If a pixel covers part of the surface, the values of T corresponding to the intersecting rays are returned by the algorithm described in figure 4. Those values are used to calculate the interval values of X, Y and Z, corresponding to the set of all the possible intersections of the beam.

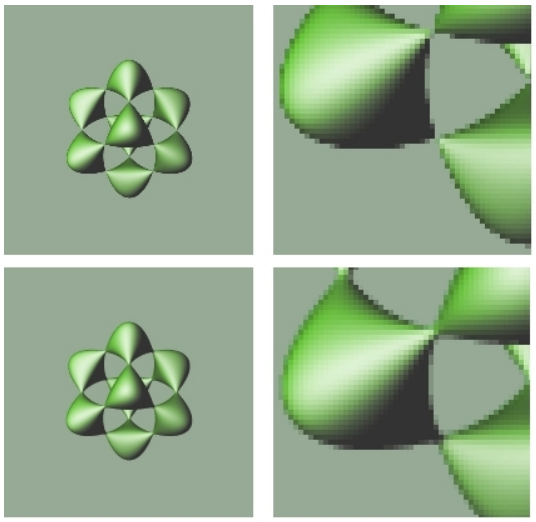

Figure. 7: A Chub's surface was generated using adaptive Anti aliasing and Interval Arithmetic in 19 minutes (top left and right). IAA takes 14 minutes (bottom left and right).

To calculate the interval normal, the interval version of the derivative of the function $f'(x, y, z)$ can be used: $F'(X, Y, Z)$. Also, an interval version of the dot product can be used, in which the real values are replaced by their corresponding interval variables. Given two interval vectors A and B, the interval dot product is $A.X * B.X + A.Y * B.Y + A.Z * B.Z$. This interval operation can be used to calculate the dot product between the interval normals and the beams.

IV. Experimentation and results

The IAA method was tested on the surfaces presented in figure 8. The comparisons have been performed against an adaptive anti aliasing algorithm based on point sampling (using a guaranteed algorithm based on interval arithmetic for the intersection test). Both techniques were applied to the classical interval ray tracing algorithm based on a branch-and-bound strategy. All the examples were generated using a resolution of $500x500$ pixels. The hardware used to test the surfaces was a Pentium 4 (2.6) with 1 Gb. of memory RAM. The algorithms were programmed in `Borland C++` 6.0, using a graphic library created in that language to test the examples presented in this paper. The interval library used was $ivalDb$. This library was developed by the MiceLab group at Girona University, an it is currently used in many tasks related to guaranteed algorithms for structures, control and medicine.

Figure 8 (a and b) shows a Twist with shadows. Fine details of the shadow are not well visualized using point sampling (a). Using IAA, those details are better visualized (b). Problems in the visualization of the point sampling example occur because shadow rays miss the thin details of the Twist. The visualization of figure 8a takes 27 minutes; figure 8b takes 20 minutes. The time difference is due to the fact that IAA detects pixels without much variations inside,

using one single intersection test. In the point sampling test, at least four rays are traced for every pixel. Figure 8c shows an example of a blobby surface visualized with the IAA algorithm in 15 minutes. Figure 8d represents the Tri-trumpet. Details of that surface are presented in figure 8e for adaptive point sampling and figure 8f for IAA. Using our method, the borders look better and the image is created in 8 minutes. Using point sampling, the sections appear separated although interval arithmetic is used for the intersection tests. Using point sampling, the algorithms takes 7 minutes.

V. Conclusions

In this paper we have presented a guaranteed adaptive anti aliasing method (IAA) for the interval ray tracing of implicit surfaces. It can be adapted from a traditional algorithm for ray tracing implicit surfaces, without increasing the complexity of the intersection tests. Also, the proposed technique generates better visualization results than methods based in point sampling, specially for surfaces with thin features. IAA is completely based on interval arithmetic, which guarantees the reliability of the algorithm. Although the IAA algorithm improves the computation time in some cases (as is shown in the presented examples), it is possible that the times obtained would be equal or higher than a guaranteed algorithm based on point sampling and Interval Arithmetic that renders the same surface. However, the latter situation never occurred in the examples presented in this paper.

IAA uses Interval Arithmetic, so the ray tracing algorithms based on floating points can improve the computational time of the rendering of implicit surfaces, but the improvement in the quality of the images generated using IAA over the same images generated using point sampling is clear. IAA is intended to be used in those cases in which the quality of the image is preferred over efficient times.

As future work, we are planning to apply our method to reflections and refractions. In this paper, view and shadow rays are used for visualization of the surfaces.

Acknowledgements

This work has been partially funded by the European Regional Development Fund and the Spanish Government (Ministerio de Ciencia y Tecnología) through the co-ordinated research projects DPI2004-07167-C02-02, DPI2005-08668-C03-02, TIN2004-07451-C03-01, and TIN-2007-68066-C04-01 and by the government of Catalonia through SGR00296.

References

[1] J. Amanatides. Ray tracing with cones. *Computer Graphics*, 18(3):129–135, 1984.

[2] O. Capriani, L. Hvidegaard, M. Mortensen, and T. Schneider. Robust and efficient ray intersection of implicit surfaces. *Reliable Computing*, 1(6):9–21, 2000.

[3] A. de Cusatis, L. de Figueredo, and M. Gatas. Interval methods for ray casting implicit surfaces with affine arithmetic. *Proceedings of SIBGRAPH*, 1999.

[4] J. Flórez, Mateu Sbert, Miguel A. Sainz, and Josep Vehí. Improving the interval ray tracing of implicit surfaces. *Lecture Notes in Computer Science*, 4035:655–664, 2006.

[5] J. Flórez, Mateu Sbert, Miguel A. Sainz, and Josep Vehí. Guaranteed adaptive antialiasing using interval arithmetic. *Lecture Notes in Computer Science*, 4488, 2007.

[6] J. Genetti and D. Gordon. Ray tracing with adaptive supersampling in object space. *Graphics Interface*, pages 70–77, 1993.

[7] John Hart. Sphere tracing: Simple robust antialiased rendering of distance based implicit surfaces. *Modeling, visualizing and Animating implicit surfaces. SIGGRAPH'93 Course Notes*, 1993.

[8] P. Heckbert and P. Hanrahan. Beam tracing polygonal objects. *Computer Graphics*, 18(3):119–127, 1984.

[9] D. Kalra and A. Barr. Guaranteed ray intersection with implicit surfaces. *Computer Graphics (Siggraph proceedings)*, 23:297–206, 1989.

[10] Don Mitchell. Robust ray intersection with interval arithmetic. *Proceedings on Graphics interface '90*, pages 68–74, 1990.

[11] A. Paiva, H. Lopez, T. Lewiner, and L.H. de Figueredo. Robust adaptive meshes for implicit surfaces. *Computer Graphics and Image Processing, SIBGRAPH*, 2006.

[12] J.F. Sanjuan-Estrada, L.G. Casado, and I. García. Reliable algorithms for ray intersection in computer graphics based on interval arithmetic. *XVI Brazilian Symposium on Computer Graphics and Image Processing*, pages 35–44, 2003.

[13] T. Whitted. An improved illumination model for shaded display. *Communications of the ACM*, 23(6):343–349, 1980.

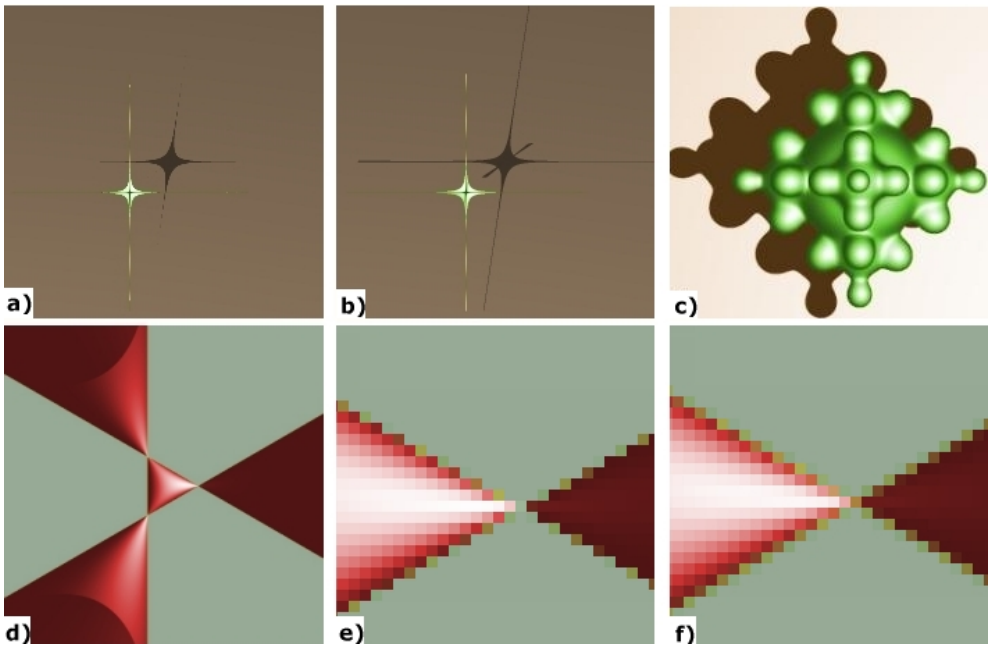

Figure. 8: Experimentation images. An image with thin features is not correctly rendered using point sampling (a).Using IAA the result is better (b). The method can be used for general surfaces, like a set of blended spheres (c). Also, a surface with critical points like the Tri-trumpet (d) is not correctly rendering using point sampling (e), but the critical points are detected using IAA (f).