An improved splat-based ray tracing for point-based objects

Ali Beddiaf
Department of computer science
University Hadj Lakhdar of Batna
05000 Batna, Algeria
Email: alibeddiaf@gmail.com

Med Chaouki Babahenini
LESIA Laboratory, Department of computer science
University Mohamed Khider of Biskra
07000 Biskra, Algeria
Email: chaouki.babahenini@gmail.com

***Abstract*—This paper aims to improve the process of photo-realistic images production in matter of quality of the point-based scenes rendered by splats using the ray tracing technique, where we exploit the intrinsic overlap of the splats to get per-pixel shading that is very close to Phong shading (in the case of polygonal meshes), without an additional preprocessing phase (no normal field computation for each splat). Moreover, this work is mainly introduced to have more accurate rendering on the edges and sharp regions. Note that our model does not require point cloud with high density, but they have to be hole-free and closed. The quality of the resulted images proves the efficiency of the proposed approach.**

***Keywords*—Ray tracing, point-based rendering, splatting, overlap, normal interpolation.**

I. Introduction

The problem of illumination is considered as the final stage in the rendering process for any graphics project. It concerns the interactions between the objects of the scene and the light sources. This may be done locally where the surface of the objects contains itself information about the portion of the light that it receives and the final image will be simply the projection from the object space to the screen space like flat, Gouraud or Phong shading. Or globally, here more complicated illumination has to be computed because both direct and indirect shading are taken into account. Ray tracing is one of the techniques that simulates the real radiant intensity exchange between surfaces of the scene.

Point as a primitive of representation has been introduced firstly by Levoy and Whitted [1]. Then an important local illumination model has been proposed by Zwicker et al. [2] where they replace each point by a disc called splat. The efficiency of this last has been proven but the only lack was the realism which is low because the visualization is made by rasterization technique. For more realistic rendering, visualization techniques based ray tracing must be used. Various approaches have been proposed in this context [3] [4] but many of them did not interest on solving conflicts during the intersection points computation that requires a specific visibility computation especially in the region of high curvatures. Our work is proposed mainly for this requirement in order to have a high quality rendering. Similar and recent works [5] [6] interest to the same problem of splats overlap in the regions of high curvatures but they only treat the situation by proposing clipping techniques as an extension of the original Zwicker's approach [2] where the visualization is made by rasterization and not ray tracing.

In section 2, we discuss some previous work related to ray tracing and splatting. The model we propose is detailed in section 3, while the experimentation is made in section 4. In the last section we give some conclusions and perspectives.

II. Related work

A. Phong splatting

Phong splatting has been proposed initially by Botsch et al. [7] in order to extend surface splatting introduced by Zwicker et al. [2], with acoustic effects (specular reflection, transparency and shininess). These acoustic effects were generated through what Botsch et al. [7] called the normal field. Each splat has to be associated with additional information: normal field. This last is computed in preprocessing step basing on the principal curvatures of subsets of point cloud. The result will be: the central normal, two principal directions $(\overrightarrow{u}, \overrightarrow{v})$ and two coefficients (α, β), for each splat!.

Phong splatting approach was used with rasterization visualization technique (based shaders) where the final image is simply projected to the screen space. Recently the normal field has been reused with ray tracing visualization technique by Linsen et al. [3], and the main goal was still to get normals as accurate as possible along the splats surface of the object.

B. Ray tracing point sampled geometry

Schaufler et al. [4] have introduced a global illumination model that works on scattered point could. Their approach gathers subsets of point cloud hit by the ray in a cylinder, and the final attributes (position, normal and color) will be weighted according to the distance between these points and the cylinder axis. Actually, the main drawback lies in the fact that the final image is view-dependent [4], because the intersection points position between the rays and the surface is weighted according to a given subset of points, and as this subset may change from one view point to another, the same patch of surface may have two different positions according to the view point.

C. Splat-based ray tracing of point clouds

Linsen et al. [3] have proposed a splat-based ray tracing for point clouds where they reuse the normal field technique as a preprocessing step. The way they weight the attributes (normals) of the overlapped splats is the same of Schaufler et

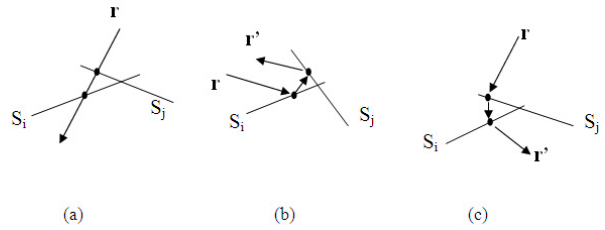

Fig. 1. Wrong intersections: (a) intersection (b) reflection (c) refraction.

al. [4]. As usual, an arbitrary splat-based object may have high curvatures in the corners, edges and sharp regions, but nothing has been mentioned about intersections computation in these particular regions which surely need special treatment.

III. Proposed model

Note that the input data in our illumination model is simply discrete points with only low density. Each point has a position, a normal and an initial size. Each one will represent one circular splat.

A. Intersections computation

As mentioned by previous works [3], the intersection computation in the ray tracing algorithm on splats needs some fitting because the splats are overlapped. Therefore, there is a set of wrong intersections (Fig. 1) that must be ignored during the computing of the intersections ray/splat, as follows:

a) Wrong intersection
b) Wrong reflection
c) Wrong refraction

B. Bounding volumes

Ray/object intersection is one of the first problems encountered when we want to make a ray tracer. One of the simplest and fastest intersections is the ray/sphere one. In this purpose, we use a sphere as bounding volume of the splat. With this way, we propose to compute the intersection at two levels:

1. Ray/sphere intersection
   This will avoid the ray/splat intersection computation when this test fails.
2. Ray/splat intersection
   When the incoming ray intersects the sphere, here we look if the ray also intersects the splat.

C. Ray/sphere intersection

We consider the ray R(t) formulated in parametric way, as follows:

$$R(t) = (o, \overrightarrow{d}) = o + t\overrightarrow{d} \tag{1}$$

Where o is the origin of the ray, $\overrightarrow{d}$ is its direction and $t > 0$.

Also, the implicit equation of a sphere is:

$$(S_x - c_x)^2 + (S_y - c_y)^2 + (S_z - c_z)^2 = r^2 \tag{2}$$

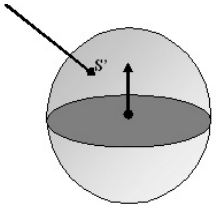

Fig. 2. A ray intersecting a sphere.

Where $c(c_x, c_y, c_z)$ is the center of the sphere, r is its radius, and $S(S_x, S_y, S_z)$ are points that lie on the sphere surface.

The first intersection $S' = R(T')$ between the ray and the sphere (Fig. 2) will be found through the solving of system of equations in order to find T' that is given as following:

$$T' = min(max(t0, 0), max(t1, 0)) \tag{3}$$

Where t0 and t1 are:

$$\begin{aligned} t0 &= \frac{-B - \sqrt{B^2 - 4AC}}{2A} \\ t1 &= \frac{-B + \sqrt{B^2 - 4AC}}{2A} \end{aligned} \tag{4}$$

A, B and C are constant:

$$\begin{aligned} A &= d_x^2 + d_y^2 + d_z^2 \\ B &= 2(d_x(o_x - c_x) + d_y(o_y - c_y) + d_z(o_z - c_z)) \\ C &= (o_x - c_x)^2 + (o_y - c_y)^2 + (o_z - c_z)^2 - r^2 \end{aligned} \tag{5}$$

D. Ray/splat intersection

In the case of ray/sphere intersection finding (called S'), we check if this ray intersects the splat or not. Starting from the intersection point S', we look for the intersection point P' between the ray $(S', \overrightarrow{d})$ and the plane P of the splat. Then, if the distance between P' and c is less or equal than r, here we can say finally that the ray $(o, \overrightarrow{d})$ intersects the splat on the point P' (Fig. 3).

In order to compute the intersection ray/plane, we let $P(x, y, z)$ the points set that lies on the plane defined by the point c (sphere/splat center) and the normal $\overrightarrow{N}$. The following equation must verified:

$$\forall p \in P : (p - c).\overrightarrow{N} = 0 \tag{6}$$

Similarly, by solving a system of equations (equations 1+6), the solution P' will be:

$$\begin{aligned} P' &= S' + T'\overrightarrow{d} \\ T' &= \frac{(c - S').\overrightarrow{N}}{\overrightarrow{d}.\overrightarrow{N}} \end{aligned} \tag{7}$$

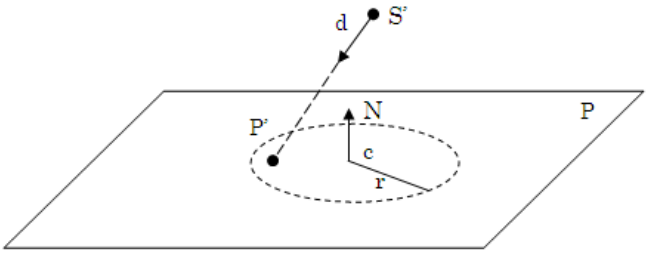

Fig. 3. A ray intersects the splat's plane.

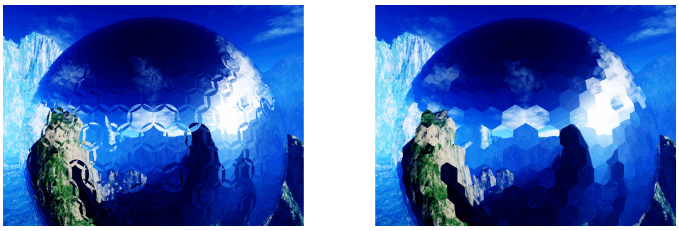

Fig. 4. Rendering of a splat-based mirror sphere by a conventional ray tracing without (left) and with wrong intersections elimination (right).

E. Normal interpolation

As mentioned, the splats have an initial size and an overlap will appear between these splats (Fig. 4).

Taking into account the elimination of the wrong intersections (as mentioned in section III-A), the ray tracing will give a flat shading (Fig. 4).

Our technique of normal estimation is slightly close to the Phong splatting technique, but we do it using ray tracing technique (without the need for shaders). In addition we don't need a preprocessing phase of normal field computation for each splat.

The way we estimate the normal of the intersection is very close to Schaulfer's way [4], in another words, the final normal will be the interpolation of the overlapped splats normals (Fig. 5). This interpolation is weighted according the distance between the intersection points and their splats centers, as follows:

$$Normal_x = \sum_{i=0}^{n} \frac{Normal_i.d_i^{'}}{r}$$

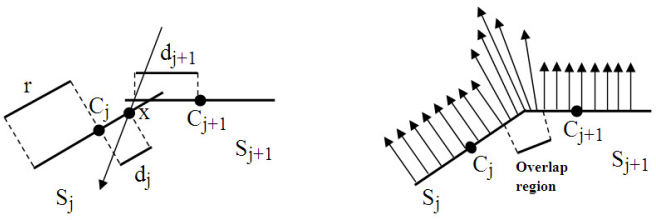

Fig. 5. Polynomial interpolation of the normal.

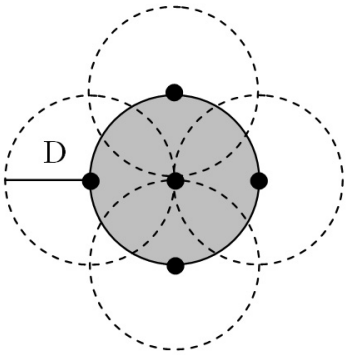

Fig. 6. The whole surface of the middle splat is fully overlapped by its neighbors.

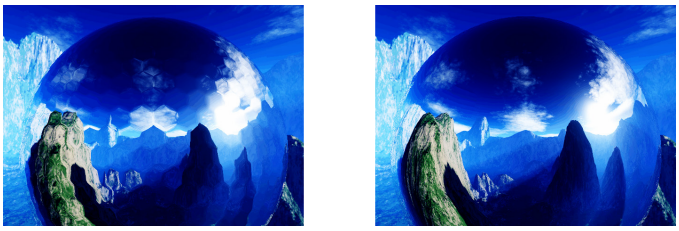

Fig. 7. Rendering of a splat-based mirror sphere by ray tracing with wrong intersections elimination and normal interpolation before (left) and after size updating (right).

$$d_i^{'} = (r - d_i) \tag{8}$$

With the mentioned way, we'll obtain interpolated normals only in the overlap regions; however the rest will stay with flat shading (Fig. 7).

If we admit that the input point cloud was uniformly sampled, and each point maintains the same distance D with its neighbors then the updating of the splats size with the new size D will ensure a full overlap (Fig. 6), and consequently the normal will be interpolated throughout the splat surface (Fig. 7).

F. Visibility computation

The elimination of the wrong intersections for simple convex or concave models was sufficient to give an accurate visibility and normal computation. However, more complicated visibility computations must be done in the general case where arbitrary objects contain both low and high curvatures. For this purpose, we present in this section a set of visibility tests in order to establish a correct visibility. Our visibility algorithm proceeds in two phases:

1) Far intersections phase:

a) One or multiple face/backface intersections
In this case, we take the first face intersections inside a small interval ϵ, which come before a backface intersection (Fig. 8).

b) One or multiple face intersections
Here, the face intersections are not taken into account because there is no backface intersection

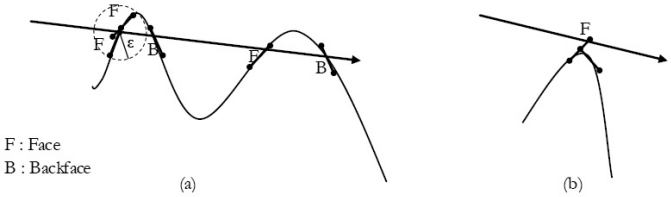

Fig. 8. (a) Multiple face/backface intersections. (b) Unique face intersection.

coming after (the ray is about hitting an edge). Note that the objects must be hole-free and closed (Fig. 8).

2) Near intersections phase: After finding the good far intersection, we base on the curvature (convexity/concavity) of the overlapped splats (inside a small interval ϵ) to fix the position and the normal of the intersection point:

a) Concavity
The decision of the concavity is based on the splats centers and their normals, as follows:

$$\begin{aligned}\omega &= \omega_1 + \omega_2 \\ &= \arccos((\overrightarrow{c_j c_{j+1}}.\overrightarrow{N_j})/(\|\overrightarrow{c_j c_{j+1}}\|.\|\overrightarrow{N_j}\|)) \\ &+ \arccos((\overrightarrow{c_{j+1} c_j}.\overrightarrow{N_{j+1}})/(\|\overrightarrow{c_{j+1} c_j}\|.\|\overrightarrow{N_{j+1}}\|))\end{aligned} \quad (9)$$

if ω is less than $180°$, then the curvature is concave (Fig. 9). In this case, we keep the first face intersection and we leave the second, and we continue this process on the rest of overlapped splats till having finally one splat. This last will give us the right intersection position, however all the overlapped splats inside the interval ϵ will just contribute to the final normal using the same weighting of Schaufler et al. [4].

b) Convexity
The decision of the convexity is based on the splats centers and their normals; if ω is more than $180°$, then the curvature is convex (Fig. 9). In this case, we keep the second face intersection and we leave the first, and we continue the same process aforementioned.

Note that Linsen et al. [3] take the position of the intersection point which has the smallest distance with its splat center, among the rest of overlapped splats. The latter seems correct; however, it is not always the case, as depicted in Fig. 10, choosing $p1$ as an intersection point is a mistake even if $r1$ is less than $r2$. Schaufler et al. [4] take the interpolated position in order to have more continuous surface, but this has unfortunately made the final image dependent to the view point (view-dependent) [4].

IV. Experimental results

The experimentation was made on scenes containing one point-based object (35945 points) and the produced images have 640x480 pixels as dimension. The rendering time was between 7 and 10 seconds.

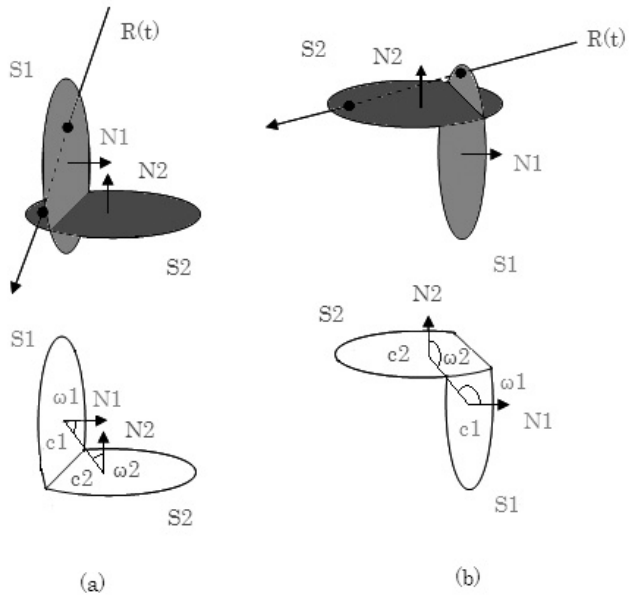

Fig. 9. Ray intersecting two splats forming: (a) Concave curvature. (b) Convex curvature.

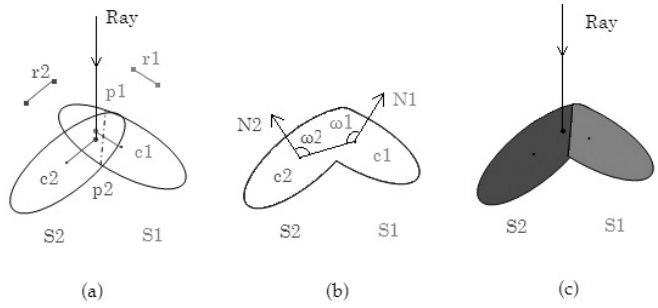

Fig. 10. (a) Ray intersecting two splats. (b) The overlapped splats represent a convex curvature. (c) The second intersection is taken while the first is rejected (according to our model).

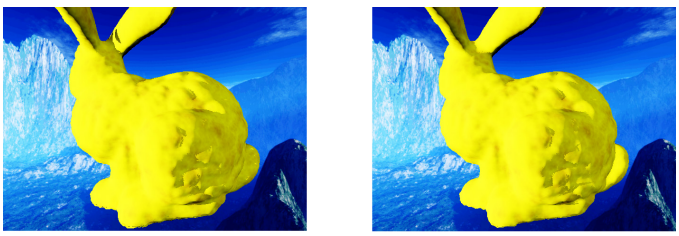

Fig. 11. First scene: general splat-based ray tracing (left) and proposed model with our visibility computation (right).

Fig. 12. Second scene: general splat-based ray tracing (left) and proposed model with our visibility computation (right).

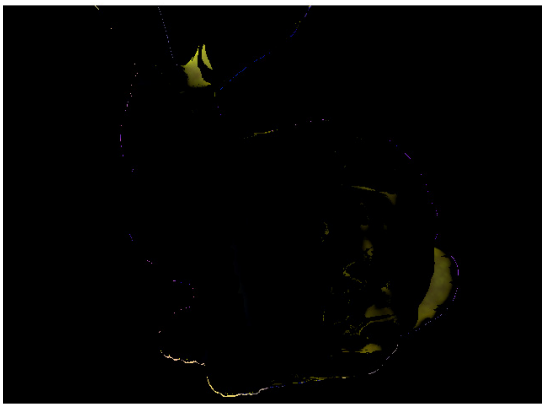

Fig. 13. Result of image differencing between the two images (first scene).

As depicted in Fig. 11 the general splat-based ray tracing has generated a wrong image in the edges of the object (the bunny) and also wrong occlusions (consequently wrong shadows), that's because the splats at those regions can not fit to the underlying surface and they have to be as small and dense as possible to represent truly the surface.

By applying our model, the surface position is correctly computed even in the edges and the regions of high curvatures, so no bad occlusions are computed and consequently no wrong shadows are generated (Fig. 11, Fig. 12).

Using the image differencing technique, we observe clearly that the general splat-based ray tracing makes mistakes on the edges, and the regions with high curvatures and consequently bad surfaces/shadows are computed, as shown in Fig. 13 and Fig. 14.

V. Conclusion and Future Work

We have proposed two contributions in this paper. The first one concerns a per-pixel illumination model for splat-based objects using the ray tracing by exploiting the full overlap of the splats to ensure high quality normal interpolation, and the second is a visibility computation that removes the conflicts between overlapped splats especially in the regions of high curvatures which results a more accurate rendering.

Recall that the more complex the scene is, the slower rendering will be, so for a fast rendering, this work is still open for hardware accelerations [8].

Fig. 14. Result of image differencing between the two images (second scene).

References

[1] M. Levoy and T. Whitted, *The Use of Points as a Display Primitive*, ser. UNC report. University of North Carolina, Department of Computer Science, 1985. [Online]. Available: http://books.google.dz/books?id=wUmGHAAACAAJ

[2] M. Zwicker, H. Pfister, J. van Baar, and M. H. Gross, "Surface splatting." in *SIGGRAPH*, 2001, pp. 371–378. [Online]. Available: http://dblp.uni-trier.de/db/conf/siggraph/siggraph2001.htmlZwickerPBG01

[3] L. Linsen, K. Muller, and P. Rosenthal, "Splat-based ray tracing of point clouds," *Journal of WSCG*, vol. 15, no. 1-3, pp. 51–58, 2007. [Online]. Available: http://scholar.google.com/scholar?hl=enbtnG=Searchq=intitle:Splat-based+Ray+Tracing+of+Point+Clouds0

[4] G. Schaufler and H. W. Jensen, "Ray tracing point sampled geometry," in *In Rendering Techniques 2000: 11th Eurographics Workshop on Rendering*, 2000, pp. 319–328.

[5] R. Ivo, C. Vidal, and J. Cavalcante-Neto, "A new method for cutting splats of models with sharp features," in *Proceedings of the 2011 24th SIBGRAPI Conference on Graphics, Patterns and Images*, ser. SIBGRAPI '11. Washington, DC, USA: IEEE Computer Society, 2011, pp. 17–24. [Online]. Available: http://dx.doi.org/10.1109/SIBGRAPI.2011.1

[6] R. F. Ivo, C. A. Vidal, and J. B. C. Neto, "A method for clipping splats on sharp edges and corners," *The Visual Computer*, vol. 28, no. 10, pp. 995–1004, 2012.

[7] M. Botsch, M. Spernat, and L. Kobbelt, "Phong splatting," in *Proceedings of the First Eurographics conference on Point-Based Graphics*, ser. SPBG'04. Aire-la-Ville, Switzerland, Switzerland: Eurographics Association, 2004, pp. 25–32. [Online]. Available: http://dx.doi.org/10.2312/SPBG/SPBG04/025-032

[8] S. Kashyap, R. Goradia, P. Chaudhuri, and S. Chandran, "Real time ray tracing of point-based models," in *Proceedings of the ACM SIGGRAPH Symposium on Interactive 3D Graphics and Games: Posters*, ser. I3D '10. New York, NY, USA: ACM, 2010, pp. 4:1–4:1. [Online]. Available: http://doi.acm.org/10.1145/1730971.1730976