STAG: Smart Tools & Apps for Graphics (2014)
Andrea Giachetti (Editor)

Adaptive clipping of splats to models with sharp features

R. Ivo[1] & F. Ganovelli[2] & C. Vidal[1] & R. Scopigno[2]

[1] Federal University of Ceará, Brazil [2] Istituto di Scienza e Tecnologie dell′Informazione, CNR, Italy

Abstract

Splat-based models are a good representation because of its absense of topology, making complex modeling operations easier, but keeping the same approximation ratio from triangular meshes. However corners cannot be properly represented by splats without clipping them. We present a new method for clipping splats in models with sharp features. Each splat is an ellipse equipped with a few parameters that allow to define how the ellipse can be clipped against a bidimensional rational Bézier curve and thus it can be used for all those surfaces that show a large number of edge features and different sampling rate around them. The simple and uniform data used to define the clipping curve makes easy the implementation in GPU. We designed and implemented an automatic computation of the clipping curves and a pipeline for sampling a generic surface with splats and render it. In this paper we show how this technique outperforms the previous clipping techniques in precision for objects such as mechanical parts and CAD- like models keeping the rendering speed.

1. Introduction

In the last three decades, the consistent improvements in representation, modeling, processing and rendering of point-based models [LW85, GD98, PZvBG00, ABCO*01, AD03, FCOAS03, PKKG03, WTG04, AA06] led to increased interest and use of those types of models in many applications, ranging from CAD-like models to deformable models and fluids. Clouds of points are an interesting alternative to mesh-based models in situations that require highly complex polygonal meshes. Moreover, points are a natural primitive in models generated with the use of 3D scanning devices, which are becoming more and more popular recently [KB04, GP07]. However, since the point is a primitive without dimension, its representation is usually extended to that of a splat, which includes orientation and a flat domain around the point, in order to fill the gaps between the samples [ZPvBG01]. Each splat can be viewed as a small circle or ellipse that locally approximates the surface.

In many applications, such as the models used in engineering, the ability to render edges and corners is essential. Also, sharp features often appear in models generated by Boolean operations (CSG) [AD03, PKKG03] and in physical simulations of cracking and breaking of materials [PKA*05]. Because of the circular or elliptical nature of the splats, an infinite amount of them would be required in order to represent edges perfectly, unlike triangle meshes, which can be aligned around the sharp feature line. However, if the locations of discontinuities are known, the splats can be adapted either by changing the sampling rate, by rearranging the samples or by clipping.

Figure 1: *The proposed method of clipping splats allows to adapt them to curved edges regardless of the number of samples of the model and the distance from the viewer. The clipping curve is sampled in pixel precision.*

Usually, when the number of splats increases, the surface is represented more accurately. Thus, artifacts caused by the crossing of splats in sharp edges can be minimized through an increase of sampling rate. However, even if the surface's splats can be processed extremely fast by exploiting the programmable features of current graphics hardware, the processing time is proportional to the number of geomet-

DOI: 10.2312/stag.20141235

ric primitives. Besides, the quality of that kind of representation depends upon the distance from the viewer to the object.

Sharp features are usually interpreted as regions with infinite curvature in a certain direction. For this reason, a finite amount of samples is not enough to represent them properly. Thus, sharp features have to be explicitly represented. The most common way of doing that is to use one line to represent edges, and multiple lines to represent corners. However, when the samples have different distributions and sizes, on both sides, around a curved edge, to approximate that curve by a line results in noticeable artifacts.

When a splat occupies a large area in screen space and is crossed by a curved edge, it requires special treatment so that its rendering does not result in artifacts. For those situations, we devised a way of clipping a splat with respect to a local curve. This is more precise and less costly than clipping a splat by a set of straight lines. In our approach, we send the minimum information possible to the GPU - in fact, the amount of information sent to GPU is almost equal to that of other techniques -, but we obtain a substantial improvement in edge representation in splat-based models.

The contributions of this work can be listed as follows:

1. We present a novel way of clipping splats that occupy a large area in image space and that are close to a sharp feature. In those situations, a splat is clipped by a local curve and the quality of the resulting model's representation is independent of the viewer's distance, guaranteeing the same pixel precision.
2. We define a light clipping data structure, with fixed parameters, that is easy to implement on GPU.
3. To illustrate an application of the technique, we propose a new way of converting a mesh-based model into a splat-based model with fewer samples, but maintaining a good representation of sharp features.

2. Related Work

Splatting-based techniques are the most common approaches for rendering point clouds because they offer good balance between performance and quality [ZPvBG01, BSK04, ZRB*04, BHZK05]. During rendering, blending of the overlapping regions of nearby splats usually produce the visual impression of a smooth surface. In regions where the curvature of the surface is high, blending of overlapping splats produce some artifacts that can be mitigated by increasing sampling density. However, in sharp features, even when the geometry is very simple, such as the intersection of two planes, that approach is not practical.

Boolean operations on point-sampled models usually generate sharp edges. In [AD03], all the splats that crossed the intersection line between two objects are replaced with smaller ones. Despite the reduction of artifacts, adding more samples makes the rendering more expensive and does not conceal the artifacts completely. In [PKKG03], the sharp feature is resampled in such a way that, in each sampling position along the ridge line, two new concentric disks are placed so that their respective planes locally coincide with the two faces on each side of the ridge. Then, those two disks are clipped with respect to each other's planes. That type of resampling approximates the ridge line with a set of straight line segments. Therefore, in regions where the ridge curve has high curvature, a lot of samples have to be used to approximate it well. Wicke et al. [WTG04] do not add new splats to the model. Instead, during rasterization, for each fragment of a splat that crosses a ridge line, their algorithm finds the two closest splats on the other side of the ridge line, the so-called clip partners, and use the CSG tree for an inside/outside classification in order to decide if the fragment is removed or not. Their approach is view-dependent, needs modeling information, and cannot be rendered on a GPU, which makes the rendering of complex models very slow.

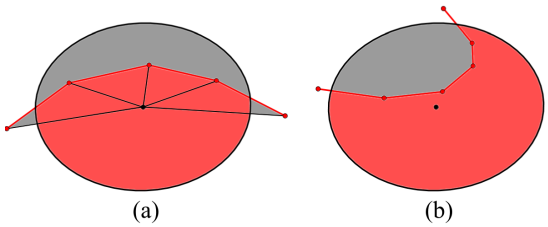

Figure 2: *Problem in Zhang et al. approach [ZK07]. (a) To implement the rendering in GPU, the authors use a triangle fan between the splat's center and the polyline vertices. (b) This approach requires that the center of the splat is on the same side of all segments, not supporting some simple cases like this.*

Zwicker et al. [ZRB*04] present a hardware renderer that can clip splats with one or more clipping planes without modeling information. However, when more than two clipping planes affect one splat, the results are ambiguous. The technique shares the same problem with the technique proposed in [PKKG03] because, the edge's curve cannot be properly adapted by a straight line.

Zhang et al. [ZK07] use a hybrid data structure: in addition to the splats, which they name surface points, a set of polylines represent the model's ridge lines, which can be input either by the user or by an edge detection method for point clouds or meshes [GWM01, KBSS01, PKG03, DOHS08]. These polylines are usually more refined than the cloud of splats. When a given splat intersects at most two of a polyline's segments, the clipping of the splat is performed as in previous algorithms. However, when the splat intersects more than two segments of a ridge line, the splat is rendered using a triangle fan as shown in Figure 2a. This approach allows splat rendering in GPU, even if the number of clipping segments varies. However, the technique is restricted to

cases in which the center of the splat lies on the same side of every clipping segment. Therefore, it cannot handle cases such as that depicted in Figure 2b. Moreover, to render these splats, many points are sent to the GPU, making the rendering slower.

3. Clipping of Splats

One of the main problems faced by other techniques is using straight lines to approximate an edge locally. If the size difference of the splats around the edge is big, the error in approximating a curved edge by a straight line becomes too large. Sending more complex structures, as polylines, can slow down rendering, even with several restrictions on input. Using a curve to clip a splat has some advantages over previous approaches:

1. the approximation error of the model relative to the original surface is lower;
2. the clipping is smooth regardless of the density of splats on the other side of the ridge and the viewer distance from the model;
3. the data structure for each sample is fixed, making it easy to implement on GPU.

In Section 3.1, we describe how the clipping curve is represented and stored. In Section 3.2, we define the clipping area and the method used for determining when a certain point is located in this area.

3.1. Clipping Curve

In this work, we use a rational Bézier curve for clipping (see Equation 1). Its is added adjustable weights for each control point allow for better approximations to arbitrary curves. This curve is defined as:

$$\mathbf{B}(t) = \frac{\sum_{i=0}^{n} b_{i,n}(t) w_i \mathbf{P}_i}{\sum_{i=0}^{n} b_{i,n}(t) w_i} \tag{1}$$

where w_i and $b_{i,n}(t)$ are, respectively, the weights and the blending Bernstein polynomials associated with the control points. The Bernstein polynomials are defined as

$$b_{i,n}(t) = \binom{n}{i} t^i (1-t)^{n-i}. \tag{2}$$

These curves can be used, among other purposes, to represent conic sections exactly, which is very useful in engineering models. Since implementation on GPU does not allow different types of data to be rendered, we used only one class of those curves, the rational Bézier curve with 3 control points, $\mathbf{P_1}$, $\mathbf{P_2}$ and $\mathbf{P_3}$. The weights for the endpoints are fixed to 1.0 and the weight, w, for the middle control point is variable. Figure 3 shows different rational Bézier curves for different values of w. With this simplification, Equation 1 can be rewritten as:

$$\mathbf{B}(t) = \frac{(1-t)^2 \mathbf{P_1} + 2t(1-t)w\mathbf{P_2} + t^2 \mathbf{P_3}}{(1-t)^2 + 2t(1-t)w + t^2} \tag{3}$$

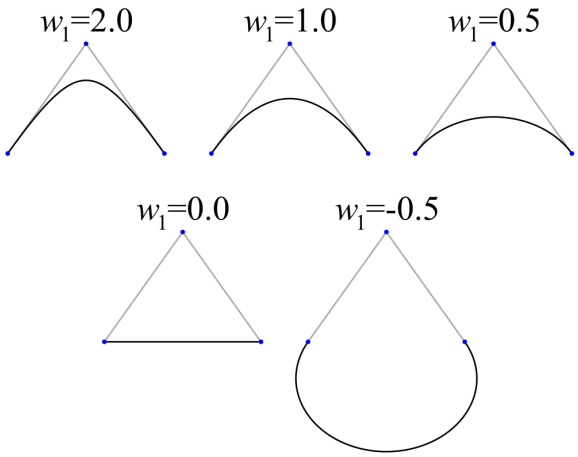

Figure 3: *Different curves formed by the variation of the weight applied to the middle control point. The wights associated with the end points are set to 1.0.*

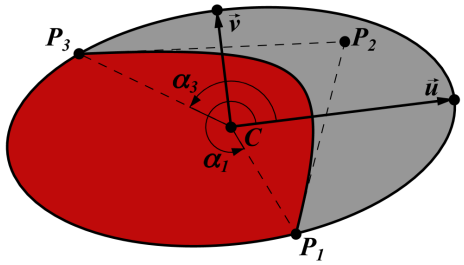

Figure 4: *Elements used to clip a splat. The clipping curve is defined by three control points, where the endpoints are fixed on the splat's border and represented by its central angles α_1 and α_3, respectively. The clipping area is bounded by the splat's arc starting in $\mathbf{P_1}$ and finishing in $\mathbf{P_3}$ in counterclockwise and the rational Bézier curve.*

A splat S can be defined by its center $\mathbf{C}$ and two orthogonal vectors $\mathbf{u}$ and $\mathbf{v}$, the main axis of the ellipse. The unit vectors $\mathbf{u_n}$ and $\mathbf{v_n}$ have the same direction as the vectors $\mathbf{u}$ and $\mathbf{v}$, respectively. These two unit vectors form the local basis of the splat. Each splat has its own clip curve, which can be defined locally using this local coordinate system. The curve's endpoints are located on the splat border, so, the points $\mathbf{P_1}$ and $\mathbf{P_3}$ can be represented by the angles α_1 and α_3, respectively, where α_1 is the central angle between the vector $\mathbf{u}$ and the vector $\mathbf{P_1} - \mathbf{C}$ in counterclockwise and α_3 is defined in a similar way with respect to point $\mathbf{P_3}$ (Figure 4). This representation is used to reduce the quantity of floating point numbers sent to shaders. In the vertex shader, for example, that representation is needed to compute the coordinates of these points from the angles. The distance from the center of the ellipse to a point on its border is computed

as:

$$r(\alpha) = \frac{|\mathbf{u}||\mathbf{v}|}{\sqrt{(|\mathbf{v}|cos(\alpha))^2 + (|\mathbf{u}|sin(\alpha))^2}}. \quad (4)$$

Thus, a point on the splat's border computed from its central angle is given by:

$$\mathbf{P_i} = (r(\alpha_i)cos(\alpha_i); r(\alpha_i)sin(\alpha_i)) \quad (5)$$

in splat's coordinate system.

Point $\mathbf{P_2}$ is represented by the coordinate pair $(P_{2_u}; P_{2_v})$ in the splat's coordinate system.

3.2. Clipped Area

Analogous to what is proposed in [ZRB*04], where the order of the clipping segment's vertices define the clipped area, the order of the control points define the clipped area of the curve on the splat. This clipped area is bounded by the elliptical arc starting at $\mathbf{P_1}$ and finishing at $\mathbf{P_3}$ in counterclockwise sense and the curve.

In order to determine if a point will be clipped, it is necessary to classify the point relative to the clipping curve defined by Equation 3. Considering arbitrary values of the parameter t, that infinite plane curve divides its plane into two semi-spaces: a convex semi-space, which we denote the internal region, and a concave semi-space, which we denote the external region. Since the weight, w, will always be considered positive here, the clipping curve segment will always be inside the triangle formed by its control points. Thus, the midpoint, M, of the segment $\overline{\mathbf{P_1P_3}}$ is always in the internal region.

Considering that we want to classify a point $\mathbf{P_q}$ on the splat's plane and that is represented in the splat's local coordinates, then, if the segment $\overline{\mathbf{P_qM}}$ intersects the curve, $\mathbf{P_q}$ is in the external region, otherwise, it is in the internal region. The parametric line that passes through points $\mathbf{P_q}$ and $\mathbf{M}$ is defined as:

$$\mathbf{R}(t_l) = \mathbf{P_q} + t_l(\mathbf{M} - \mathbf{P_q}). \quad (6)$$

By matching equations 3 and 6, we have a system of two equations and two unknowns, the line and curve parameters, t_l and t_b, respectively. The exact solution for t_b at the intersection points given by:

$$t_b = \frac{a \pm \sqrt{a^2 - bc}}{c}, \quad (7)$$

where:

$$\begin{aligned} a &= \mathbf{d_2} \cdot [\mathbf{P_1} - \mathbf{P_q} + w(\mathbf{P_q} - \mathbf{P_2})] \\ b &= \mathbf{d_2} \cdot (\mathbf{P_1} - \mathbf{P_q}) \\ c &= \mathbf{d_2} \cdot [\mathbf{P_1} + \mathbf{P_3} + 2(w\mathbf{P_q} - \mathbf{P_q} - w\mathbf{P_2})] \\ \mathbf{d} &= \mathbf{M} - \mathbf{P_q} \\ \mathbf{d_2} &= (d_y; -d_x) \end{aligned} \quad (8)$$

Thus, the points $\mathbf{X_1} = \mathbf{B}(t_{b_1})$ and $\mathbf{X_2} = \mathbf{B}(t_{b_2})$ are the intersection points between the line and the curve. Using these

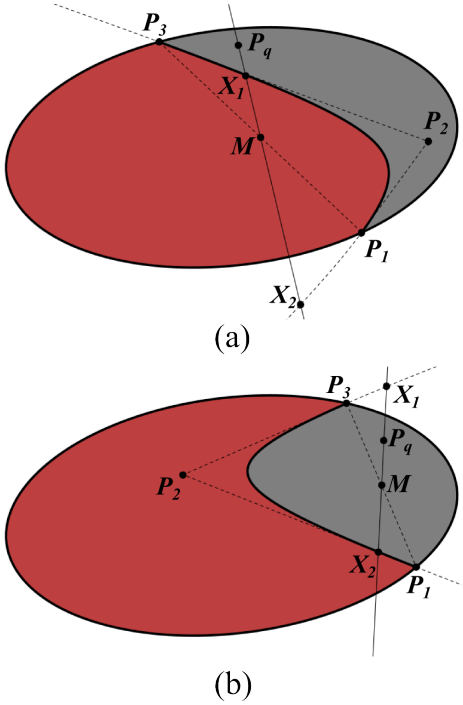

Figure 5: *Clipped area definition. The clipped area is shown in gray. (a) The clipped region is the external region because, going from* $\mathbf{P_1}$ *to* $\mathbf{P_3}$ *through* $\mathbf{P_2}$*, one makes a left turn at* $\mathbf{P_2}$*. For any point* $\mathbf{P_q}$ *in this region,* $\mathbf{X_1}$ *belongs to the line segment* $\overline{\mathbf{P_qM}}$*. (b) The clipped region is the internal region because, going from* $\mathbf{P_1}$ *to* $\mathbf{P_3}$ *through* $\mathbf{P_2}$*, one makes a right turn at* $\mathbf{P_2}$*. For any point* $\mathbf{P_q}$ *in this region, the line segment* $\overline{\mathbf{P_qM}}$ *does not contain either* $\mathbf{X_1}$ *or* $\mathbf{X_2}$*.*

point in Equation 6, we can compute the corresponding line parameters, t_{l_1} and t_{l_2}, respectively. If one of these is between 0 and 1, the curve is crossed by the $\overline{\mathbf{P_qM}}$ which indicates that $\mathbf{P_q}$ is in the external region (Figure 5a). Otherwise, point $\mathbf{P_q}$ is located in the internal region (Figure 5b).

When traveling from $\mathbf{P_1}$ to $\mathbf{P_3}$ along the polyline $\mathbf{P_1} - \mathbf{P_2} - \mathbf{P_3}$ one makes a left turn at $\mathbf{P_2}$, the region to be clipped is the external region (Figure 5a). If, on the other hand, one makes a right turn at $\mathbf{P_2}$, the region to be clipped is the internal region (Figure 5b).

To reduce the number of operations in the shader, only the points inside the triangle formed by the control polygon are checked with respect to the curve. The remaining points can be clipped using the oriented lines $\overrightarrow{\mathbf{P_1P_2}}$ and $\overrightarrow{\mathbf{P_2P_3}}$. If the region to be clipped is the external region, a point is removed if it is to the right of any of those lines (Figure 5a). However, if the region to be clipped is the internal region, a point is removed only if it is to the right of both lines (Figure 5b).

3.3. Some implementation optimization

The GPU implementation is done in the vertex and fragment shaders. All operations common to all fragments must be computed in the vertex shader and sent to the fragment shader, as is done, for example, in the conversion of central angles to coordinate pairs for the curve's endpoints, shown in equations 4 and 5, because these are high cost operations.

All clipping operations are done in the fragment shader. The proposed clipping technique needs the exact position of a point on the splat for each pixel. Thus, the splat is rasterized using the method of ray casting [BHZK05]. After ray casting, the 3D point is available, but all clipping information and methods are in splat coordinates. This 3D point can be converted to the splat's local coordinate system as follow:

$$\begin{aligned} P_{q_x} &= (\mathbf{P}-\mathbf{C})\cdot\mathbf{u_n}, \\ P_{q_y} &= (\mathbf{P}-\mathbf{C})\cdot\mathbf{v_n}, \end{aligned} \tag{9}$$

where $\mathbf{u_n}$ and $\mathbf{v_n}$ are unit vectors in the same direction as the splat's axes $\mathbf{u}$ and $\mathbf{v}$.

Detecting if a point is in the internal or in the external region of the clipping curve is a high cost operation and must be avoided if possible. Some optimizations are:

- if the splat's size in image space is too small, the clipping can be done by a simple line connecting the endpoints.
- if the weight is equal or near to zero, the clipping can be done by a simple line connecting the endpoints. If the weight is too high, the curve is almost equal to the control polygon, thus, the clipping is done using in a simpler way using the control polygon;
- if the splat is too big in image space and/or the the control polygon points are almost collinear, a big number of points can be discarded using only the control polygon. Thus, the points outside the triangular control polygon can be checked for clipping using only the control polygon.

The clipping operation by the control polygon is simpler that performed by the curve. If the clipping region is concave, the fragment is discarded if the point is on the right side of $\overrightarrow{\mathbf{P_1P_2}}$ or $\overrightarrow{\mathbf{P_2P_3}}$. If the clipping region is convex, the fragment is discarded if the point is on the right side of $\overrightarrow{\mathbf{P_1P_2}}$ and $\overrightarrow{\mathbf{P_2P_3}}$.

Usually, sending splats info to the GPU is a bottleneck in rendering time. Performing the curve clipping adds five floating point numbers to be sent to GPU: two angles representing the curve endpoints, two coordinates of the control polygon's midpoint and the curve's weight. A possible optimization is to detach the splats with clipping from the others. Most of the splats are sent to shaders of regular splats and only the splats with clipping info are sent to a shader that is specialized in curve clipping.

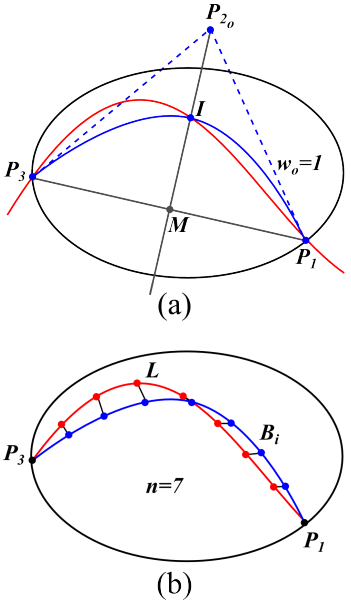

Figure 6: *Approximation of the clipping curve. (a) Initial curve set as a symetrical parabola whose midpoint is the intersection point with the ridge line. (b) The error function is defined as the average distance between equally spaced points over the clipping curve and the ridge line.*

4. Clipping Curve's Approximation

Equation 3 represents a family of curves that are specified by three control points and a scalar parameter, w. When a splat is intersected by a ridge line of the model, appropriate values for the three control points and for the parameter w have to be defined in order to have the best approximation of the ridge line inside that splat. To let the user specify those appropriate values would be highly impracticable. Therefore, in this section, we discuss how to determine the curve attributes automatically. The ridge lines to be approximated can be defined as polylines or NURBS, and can be either computed using an edge detection algorithm or set explicitly by the user. The parameters of the rational Bézier curve (Equation 3) that best fits the ridge line within a splat are determined through an optimization procedure that starts from an initial approximation (see Figure 3a and Section 4.1) and minimizes an appropriate error function (see Section 4.2).

4.1. Initial Clipping Curve

The definition of the initial approximation curve for the optimization process starts with the computation of the intersection of the ridge line with the splat (points $\mathbf{P_1}$ and $\mathbf{P_3}$). Those two points will remain fix throughout optimization. The proposed initial curve is the parabola passing through the endpoints $\mathbf{P_1}$ and $\mathbf{P_3}$ and through the intersection point between the perpendicular bisector of line segment $\overline{\mathbf{P_1P_3}}$ and

the ridge line **L** (Figure 6a). The definition of the initial values of $\mathbf{P_2}$ and w is simple by exploiting the following properties of a Bézier curve:

- When all the weights of a rational Bézier curve are equal, the curve is equal to the classical Bézier curve;
- A classical Bézier curve with three control points forms a parabola. If the control polygon forms an isosceles triangle where whose base is the segment connecting the two endpoints of the curve, the perpendicular bisector of the base is also the axis of symmetry of the parabola;
- If the curve is symmetrical, then its midpoint belongs to the perpendicular bisector of the base of the isosceles triangle used as control polygon. That point is also the midpoint of the height of the isosceles triangle.

Therefore, the weight and the central control point of the cut curve are defined as:

$$w_o = 1.0 \;\; ; \;\; \mathbf{P_{2_o}} = 2\mathbf{I} + \mathbf{M}, \tag{10}$$

where **M** is the midpoint of $\overline{\mathbf{P_1P_3}}$ and **I** is the intersection of the bisector of $\overline{\mathbf{P_1P_3}}$ and the ridge line **L**.

4.2. Error function

Let **L** be a part of the ridge line that intersects the splat and is bounded by the points $\mathbf{P_1}$ and $\mathbf{P_3}$. An error function is defined in order to account for how close the clipping curve **B** is to the ridge line **L** so that it can be used in the minimization procedure. If it were possible to find a coordinate axis so that both curves, **L** and **B**, could be converted to functions $l(x)$ and $b(x)$, respectively, the error function would be defined as:

$$e = \frac{1}{x_2 - x_1} \int_{x_1}^{x_2} |l(x) - b(x)| dx \tag{11}$$

because it is the sum of the distances between each point on function **B** and the corresponding point on line **L**. However, it is not always possible to find that common coordinate system, then a discrete approach is used. First, a set of n evenly spaced sampling points is marked on both curves, **L** and **B**. Next, the error function is defined as the arithmetic average of the distances between corresponding points:

$$e = \frac{1}{n} \sum_{i=1}^{n} |\mathbf{L}_i - \mathbf{B}_i| \tag{12}$$

The points on the clipping curve **B** cannot be spaced using the parameter of the curve, then an arc length parameterization is computed. This curve has no inflection points, then a simple numerical method table that relates parameter value to arc legth can be used to find equally spaced points on **B**. To uniformly sample the ridge line **L** depends on how the line is represented.

5. Representing a surface with Clip Splats

The ridge lines needed for clipping the splats adjacent to them can be created or detected in several ways. The splats

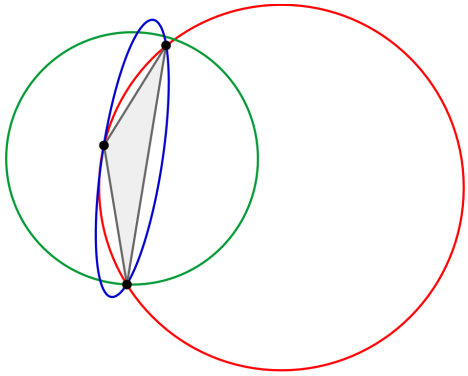

Figure 7: *Splats computed using the approach of circle centered at centroid (in green) or the approach of circumscribed circle (in red) have a lot of external area in relation to the triangle entered when it is narrow. The blue ellipse is created by the proposed approach.*

can be generated by simple conversion of a point cloud or by sampling a surface that is represented as: point clouds, meshes, implicit surfaces, etc. That sampling can be adapted according to the surface's curvature and to the proximity to the ridges. In this paper, the technique is demonstrated through a conversion from triangular meshes to splats with clipping curves. To a faithful adaptation of the clipping curves to the sharp features, the initial model must be dense to provide a good representation of the object. The following overview describes the procedure:

1. The sharp features of the mesh are detected using a simple technique described in [KBSS01]. If the normal vector's deviation between neighboring elements is too large, then the edge between them belongs to a sharp feature, which is represented by a set of polylines.
2. Each triangle of the mesh is replaced with an elliptical splat described in Section 5.1.
3. A polyline formed by the segments present in a sharp feature is assigned to each splat. The intersection of that polyline with the splat is addapted by a smooth clipping curve (see Section 4).
4. Section 4 describes the process of approximation of the polyline by the curve defined in Section 3.1 and the error between this curve and the polyline. If the error is lower than a certain threshold, then the mesh is simplified locally returning to step 1. Otherwise the process stops and the splat is created by adding the curve found as its curve clipping.

5.1. Adjusting an elliptical splat to a triangle

The most common way of converting a triangle into a splat is using its centroid and adjusting the radius of a circle to fully cover the triangle. Another common way is to use the circumscribed circle. However, those approaches do not

work well for narrow triangles, which are very common near edges and regions of high curvatures, building splats larger than needed. Tha splat that replaces a triangle must cover all triangle's area, but with the minimun extra area as possible to minimize the number of fragments to be rasterized and to avoid possible artifacts. Figure 7 shows different ways of fitting a triangle.

The circumscribed circle is the best approach when the triangle is equilateral. The proposed approach is to find the matrices needed to transform an equilateral triangle of side length equal to one into the desired triangle and apply those matrices to the circumscribed circle associated with the unit side equilateral triangle, in order to obtain the desired best fitting ellipse.

Let a triangle be formed by the points $\mathbf{P_1}$, $\mathbf{P_2}$ and $\mathbf{P_3}$. A local coordinate system is defined by the vectors:

$$\mathbf{t_x} = \frac{\mathbf{P_2} - \mathbf{P_1}}{|\mathbf{P_2} - \mathbf{P_1}|} \quad ; \quad \mathbf{t_y} = \mathbf{N} \times \mathbf{t_x} \tag{13}$$

where $\mathbf{N}$ is the unit vector, normal to the triangle with origin at point $\mathbf{P_1}$. Thus, the vertices of the triangle in local coordinates are: $\mathbf{P'_1} = \mathbf{O}$ located at the origin; $\mathbf{P'_2} = (x'_2, 0)$ located on the x-axis and $\mathbf{P'_3} = (x'_3, y'_3)$, where y'_3 is certainly positive.

Let $\Delta E_1E_2E_3$ be an equilateral triangle with side length equal to 1, where $\mathbf{E_1} = \mathbf{O}$ and $\mathbf{E_2}$ is on the x-axis. The circumscribed circle is defined by:

$$\begin{array}{rcl} \mathbf{C_e} & = & \frac{1}{3}(\mathbf{E_1} + \mathbf{E_2} + \mathbf{E_3}) \\ r_e & = & \frac{\sqrt{3}}{3} \\ \mathbf{u_e} & = & (r_e, 0) \\ \mathbf{v_e} & = & (0, r_e) \end{array} \tag{14}$$

where $\mathbf{C_e}$ is the circle's center, r_e is its radius and $\mathbf{u_e}$ and $\mathbf{v_e}$ are orthogonal axis of an ellipse that defined the same circle.

Two affine transformations are needed to transform $\Delta E_1E_2E_3$ in $\Delta P1'P2'P3'$: a scale, $\mathbf{S}$, and a shear $\mathbf{Sh}$ (Figure 8a). The scale matrix is equal to:

$$\mathbf{S} = \begin{bmatrix} s_x & 0 \\ 0 & s_y \end{bmatrix} \; ; \; s_x = x'_2 \; ; \; s_y = \frac{y'_3}{\sqrt{3}/2} \tag{15}$$

and the shear matrix is equal to:

$$\mathbf{Sh} = \begin{bmatrix} 1 & sh_x \\ 0 & 1 \end{bmatrix} \; ; \; sh_x = \frac{x'_3 - \frac{x'_2}{2}}{y'_3}. \tag{16}$$

The combination of these matrices $\mathbf{T} = \mathbf{Sh} \cdot \mathbf{S}$ is applied in points of the circumscribed circle. This will result in an ellipse passing through the three triangle vertices with the same quality of approximation as the initial circle in relation to the equilateral triangle (Figure 8b). Let $\mathbf{u''} = \mathbf{T} \cdot \mathbf{u_e}$ and $\mathbf{v''} = \mathbf{T} \cdot \mathbf{v_e}$ be the ellipse axis after the transformation. When the triangle $P_1P_2P_3$ is isosceles with base P_1P_2, the shear value is zero, then those vectors are already orthogonal and nothing is done. However, usually, $\mathbf{u''}$ and $\mathbf{v''}$ are not

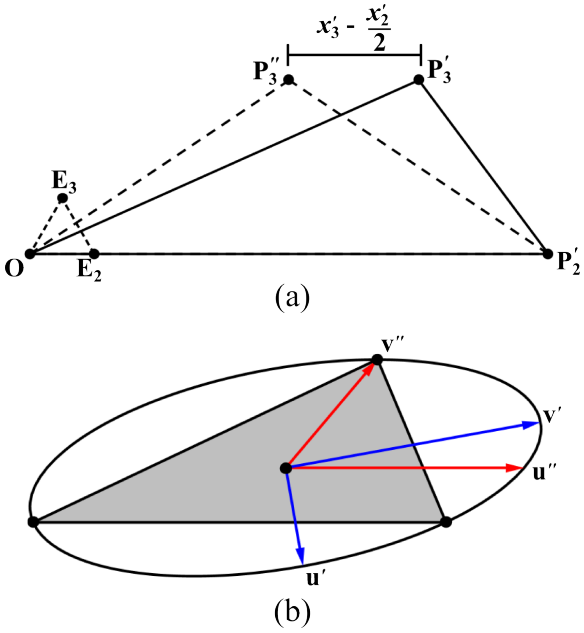

Figure 8: *Procedure of fitting a splat to a triangle. (a) $\Delta E_1E_2E_3$ is scaled to the triangle $\Delta OP'_2P''_3$ and then sheared resulting in the desired triangle $\Delta P'_1P'_2P'_3$. (b) The initial axes are transformed by the same matrix used to transform the equilateral triangle. Usually, the axes are not orthogonal after the transformation, then* PCA *is used to compute orthogonal axes to the objective ellipse.*

orthogonal. One way to find two orthogonal vectors that define the same ellipse is using *Principle Component Analysis* (PCA) on the ellipse. The covariance matrix of the ellipse is equal to:

$$\mathbf{Cov} = \begin{bmatrix} a & b \\ b & c \end{bmatrix}, \; where \; \begin{cases} a = s_x^2 + s_y^2 sh_x^2 \\ b = s_y^2 sh_x \\ c = s_y^2 \end{cases} \tag{17}$$

Let λ_1 and λ_2 be the eigenvalues of $\mathbf{Cov}$ and $\mathbf{w_1}$ and $\mathbf{w_2}$ be the corresponding unit length eigenvectors. The eigenvectors are necessarily orthogonal and the semi-axes of the splat are equal to:

$$\mathbf{q_1} = r_e\sqrt{\lambda_1}\mathbf{w_1} \; ; \; \mathbf{q_2} = r_e\sqrt{\lambda_2}\mathbf{w_2} \tag{18}$$

The first eigenvalue is always greater than the second and the eigenvectors are always in the half-plane of positive values of x. The cross product $\mathbf{q}_1 \times \mathbf{q}_2$ the same direction as the triangle's normal. Thus, the new splat attributes are:

$$\begin{array}{rcl} \mathbf{C'} & = & \mathbf{Sh} \cdot \mathbf{S} \cdot \mathbf{C_e} \\ \mathbf{u'} & = & \mathbf{q_i} \; ; \; i \in \{1,2\} \text{ and } y_{q_i} < 0 \\ \mathbf{v'} & = & \mathbf{q_i} \; ; \; i \in \{1,2\} \text{ and } y_{q_i} > 0 \end{array} \tag{19}$$

In the end, the center $\mathbf{C'}$ and the vectors $\mathbf{u'}$ and $\mathbf{v'}$ are

transformed from the triangle's local coordinates system to global coordinate system.

5.2. Sampling of splats by Edge-Collapse Mesh Simpflification

In this section, we are converting a mesh-based model in a splat-based model using clipping curves. Firstly, we find all the edges of the mesh that will define the clipping curve. Those are either the edges on the border of the mesh or the edges for angle between the normal vectors of the two adjacent faces is greater than a user-given threshold as in [KBSS01].

An edge collapse consists of merging two vertices that are connected by an edge of the mesh. This operation causes the number of faces to decrease by the number of faces that are adjacent to the collapsed edge (two if the edge is not on the border, one if it is). The choices of which edge to collapse determines the final quality of the simplified mesh. Tipically one wants to preserve the topology of the mesh, so a feasibility test is run on each candidate collapse to check whether performing the collapse would change the topology of the mesh, in which case the collapse is said to be unfeasible and it is not performed. Among the feasible collapses, a priority is determined on the base of how well the edge collapse would modify the appearance of the mesh. Although there are many possible criteria (often combined) such as variation of the normal, variation of the volume, variation of the quality of the triangles etc. a *de facto* dominating criterion is the quadric error metric [GH97]. Edge collapse simplification is adapted for producing a description by splats with clipping curves in two simple steps.

Feasiblity

Not all the triangulations of a surface can be turned into a splat-based representation with the clipping curve proposed in Section 3.1, because the clipping curve may intersect the splats in a way that cannot be approximated (for example because the intersection consists of more disconnected components). This tipically happens if the mesh is coarse, so that big triangles (and hence big splats) intersect the clipping line mutiple times. Therefore we always start from a dense mesh such that all splats can be successfully created.

Then, we add to the feasibility test the check that the splats associated with the new faces resulting from the collapse can be created. Thus, we are sure that the simplification will always return a mesh that can be turned into a splat-based representation.

Priority of collapses

We add to the approximation error associated with a collapse the approximation error of the curve fitting explained in Section 4.2, so that we can guarantee that no curve will be approximated with an error greater than a user-given value in the final model.

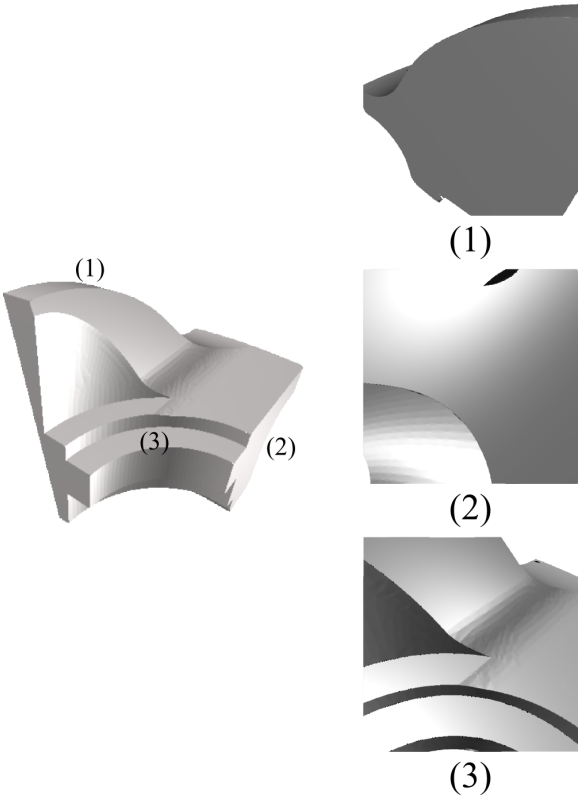

Figure 9: *Results for Fandisk Model. Initially, the model has 54,193 elements and it is reduced to 2,500 elements, but the quality of the rendering is kept. The images on the right show highlights of curved edges and the treatment technique.*

6. Results

The main goal of this work is to maintain good quality rendering of models with edges and corners even in low sampling models and regardless of the distance from the observer to the model. That goal was demonstrated by carring out the conversion of a polygonal mesh into a cloud of splats (Section 5).

Figure 9 shows the results of rendering Fandisk model reduced from 54,192 samples to 2,500. Despite the black pixels that appear because of the incomplete attaching of the clipping curves and the splats across the ridge, the approach is more faithful to the model than if straight lines are used to approximate the curved edges of the model.

Table 1 shows the rendering times for the same model, but using different amounts of splats. All the rendering tests used images with resolution of 800×800 in which the model occupies nearly 40% of the image. The tests were made in a computer equipped with an Intel®Core 2 Duo 2.88 GHz processor with a Graphics Processor Nvidia®GeForce310M. When the number of splats de-

Number of splats	Frames/sec	Splats/sec
2,500	114.65	0.143M
5,000	97.82	0.489M
54,192	31.35	2.782M

Table 1: *Rendering times of Fandisk model in different samplings.*

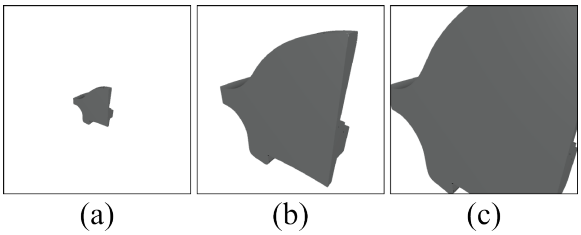

Figure 10: *Results for Fandisk Model with 2500 splats and in different zooms. (a) Model covering 3.1% of the image. (b) Model covering 37.5% of the image. (c) Model covering 80.7% of the image.*

creases, the splats of the model should be larger in order to fill the empty spaces between the samples. Bigger splats cause a greater number of pixels to be rasterized in fragment shader and a better treatment in clipping splats near ridges. However, because of the increased number of pixels sent to the fragment shader the rate of rendered splats per second becomes lower and lower. Nevertheless, it is noticed that the number of samples of a model has as stronger influence on the rate of frames per second. Therefore, our technique allows a model with less number of splats to have a rendering quality as good as that obtained with techniques that require larger number of splats, and to present lower randering times.

Occupancy Image (%)	Frames/sec	Splats/sec
3.1	298.75	0.746M
37.5	196.18	0.490M
80.7	115.57	0.288M

Table 2: *Rendering times of Fandisk model in different zooms.*

Table 2 shows rendering times for the fandisk model with $2,500$ samples in different zooms as shown in Figure 10. All the rendering tests used images with resolution 800×800. The distance from the model to the viewer influences the rendering time. This is because more fragments are rasterized and must be blended and clipped if needed. However, the frame rate is still acceptable.

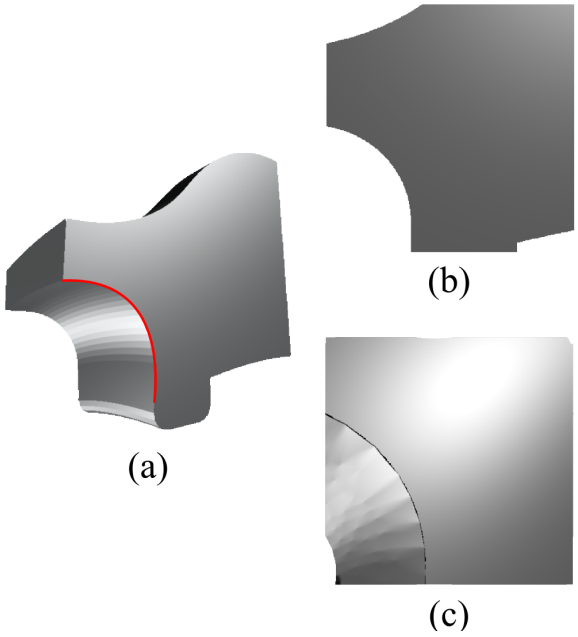

Figure 11: *Curve clipping analysis. (a) Let the curved ridge highlighted in red. (b) The curved clipping faithfully adapts the splats to the curved ridge on the surface. (c) However, by observing the two surfaces connected by the edge, artifacts can be noticed due to overlapping splats on opposite sides of the edge, because splats are planes representing a curved surface, thus, linear approximations of the curved ridge.*

7. Conclusion

Usually, modern graphics cards can render huge point cloud models very fast. However, as with any other rendering primitive, the processing costs are still proportional to the number of primitives used to represent a given object. Thus, complexity reduction for splat-sampled geometry is as important as it is for triangle meshes. For this reason, this work aims at achieving high quality renderings of models with sharp features, but with low density of samples.

In this paper, a proper way of clipping splats was presented and discussed. It was pointed out that, when the sampling rate is low, the sizes of the splats should be large in order to avoid the appearing of holes making artifacts more visible. Our approach uses local rational Bézier curves to clip splats near sharp features. The consistent data structure of the curves allows an easy implementation in GPU, allowing the rendering of splat-based models to be processed within good frame rates.

Although our curved clipping technique is able to faithfully adapt the splats to the curved ridge on the surface, there are still some improvements to be made. Splats across the ridge line are still planes adapting a curved surface. Thus,

this junction between a set of curves applied to the splats on a surface with less curvature in one side of the ridge line and a set of planes of splats on a surfaces with greater curvature on the other side of the ridge line is not perfect, showing some artifacts (Figure 11). Another problem is caused by the local approximation of the ridge line as the clipping curve of a splat. Thus, although the clipping curves of two adjacent splats with respect to the ridge line are similar, they are not unique, because of numerical errors. Therefore, holes or wrongly overlaps around the ridge line may appear.

As future works, we aim to use a sampling method over the original surface, making the splat and curve fitting procedure independent of the input. Other possible improvement can be using curved splats, instead of flat ones, with some procedure in geometric shader. The local adaptation of a curve for each splat can allow some artifacts by numeric errors. An idea to solve that is to compute a general curve for the model and use that curve to compute the local ones without a numeric approximation.

References

[AA06] ADAMSON A., ALEXA M.: Point-sampled cell complexes. In *ACM SIGGRAPH 2006 Papers* (New York, NY, USA, 2006), SIGGRAPH '06, ACM, pp. 671–680. URL: http://doi.acm.org/10.1145/1179352.1141940, doi:http://doi.acm.org/10.1145/1179352.1141940. 1

[ABCO*01] ALEXA M., BEHR J., COHEN-OR D., FLEISHMAN S., SILVA C.: Point set surfaces. In *Proceedings of IEEE Visualization 01* (2001), pp. 21–28. 1

[AD03] ADAMS B., DUTRÉ P.: Interactive boolean operations on surfel-bounded solids. In *ACM SIGGRAPH 2003 Papers* (2003), SIGGRAPH '03, ACM, pp. 651–656. 1, 2

[BHZK05] BOTSCH M., HORNUNG A., ZWICKER M., KOBBELT L.: High-quality surface splatting on today's gpus. *Proceedings Eurographics/IEEE VGTC Symposium Point-Based Graphics 0* (2005), 17–141. 2, 5

[BSK04] BOTSCH M., SPERNAT M., KOBBELT L.: Phong splatting. In *Proceedings of Symposium on Point-Based Graphics 2004* (2004), pp. 25–32. 2

[DOHS08] DANIELS J. I., OCHOTTA T., HA L. K., SILVA C. T.: Spline-based feature curves from point-sampled geometry. *The Visual Computer 24* (2008), 449–462. 2

[FCOAS03] FLEISHMAN S., COHEN-OR D., ALEXA M., SILVA C. T.: Progressive point set surfaces. *ACM Transactions on Graphics 22 4* (2003). 1

[GD98] GROSSMAN J. P., DALLY W. J.: Point sample rendering. In *Proceedings of Eurographics Workshop on Rendering 98* (1998), pp. 181–192. 1

[GH97] GARLAND M., HECKBERT P. S.: Surface simplification using quadric error metrics. In *SIGGRAPH 97 Conferece Proceedings* (1997), pp. 209–216. 8

[GP07] GROSS M., PFISTER H.: *Point-Based Graphics*. Morgan Kaufmann Publishers, 2007. 1

[GWM01] GUMHOLD S., WANG X., MACLEOD R.: Feature extraction from point clouds. In *In Proceedings of the 10 th International Meshing Roundtable* (2001), pp. 293–305. 2

[KB04] KOBBELT L., BOTSCH M.: A survey of point-based techniques in computer graphics. *Computers & Graphics 28*, 6 (2004), 801 – 814. 1

[KBSS01] KOBBELT L. P., BOTSCH M., SCHWANECKE U., SEIDEL H.-P.: Feature sensitive surface extraction from volume data. In *Proceedings of the 28th annual conference on Computer graphics and interactive techniques* (2001), SIGGRAPH '01, ACM, pp. 57–66. 2, 6, 8

[LW85] LEVOY M., WHITTED T.: *The Use of Points As Display Primitive*. Tech. rep., Computer Science Department, University of North Carolina at Chapel Hill, 1985. 1

[PKA*05] PAULY M., KEISER R., ADAMS B., DUTRÉ P., GROSS M., GUIBAS L. J.: Meshless animation of fracturing solids. In *ACM SIGGRAPH 2005 Papers* (2005), SIGGRAPH '05, ACM, pp. 957–964. 1

[PKG03] PAULY M., KEISER R., GROSS M.: Multi-scale feature extraction on point-sampled surfaces. *Computer Graphics Forum 22* (September 2003), 281–289. 2

[PKKG03] PAULY M., KEISER R., KOBBELT L. P., GROSS M.: Shape modeling with point-sampled geometry. In *ACM SIGGRAPH 2003 Papers* (2003), SIGGRAPH '03, ACM, pp. 641–650. 1, 2

[PZvBG00] PFISTER H., ZWICKER M., VAN BAAR J., GROSS M.: Surfels: Surface elements as rendering primitives. In *Proceedings of SIGGRAPH 00* (2000), pp. 335–342. 1

[WTG04] WICKE M., TESCHNER M., GROSS M.: Csg tree rendering for point-sampled objects. In *Proceedings of the Computer Graphics and Applications, 12th Pacific Conference* (2004), PG '04, IEEE Computer Society, pp. 160–168. 1, 2

[ZK07] ZHANG H., KAUFMAN A. E.: Point-and-edge model for edge-preserving splatting. *Vis. Comput. 23*, 6 (May 2007), 397–408. URL: http://dx.doi.org/10.1007/s00371-007-0098-7, doi:10.1007/s00371-007-0098-7. 2

[ZPvBG01] ZWICKER M., PFISTER H., VAN BAAR J., GROSS M.: Surface splatting. In *SIGGRAPH '01: Proceedings of the 28th annual conference on Computer graphics and interactive techniques* (2001), ACM, pp. 371–378. 1, 2

[ZRB*04] ZWICKER M., RASANEN J., BOTSCH M., DACHSBACHER C., PAULY M.: Perspective accurate splatting. In *Proceedings of Graphics Interface 2004* (2004), GI '04, Canadian Human-Computer Communications Society, pp. 247–254. 2, 4