Direct Visibility of Point Sets

Sagi Katz*
Technion – Israel Inst. of Technology

Ayellet Tal†
Technion – Israel Inst. of Technology

Ronen Basri‡
Weizmann Inst. of Science

Abstract

This paper proposes a simple and fast operator, the *"Hidden" Point Removal* operator, which determines the visible points in a point cloud, as viewed from a given viewpoint. Visibility is determined without reconstructing a surface or estimating normals. It is shown that extracting the points that reside on the convex hull of a transformed point cloud, amounts to determining the visible points. This operator is general – it can be applied to point clouds at various dimensions, on both sparse and dense point clouds, and on viewpoints internal as well as external to the cloud. It is demonstrated that the operator is useful in visualizing point clouds, in view-dependent reconstruction and in shadow casting.

Keywords: Point-based graphics, visibility, visualizing point sets

1 Introduction

In the last decade, an alternative to meshes, in the form of a point-based representation (*a point cloud*), has gained increasing popularity [Rusinkiewicz and Levoy 2000; Pauly and Gross 2001; Zwicker et al. 2002; Alexa et al. 2003; Fleishman et al. 2003; Alexa et al. 2004; Kobbelt and Botsch 2004]. Point clouds are 3D positions, possibly associated with additional information, such as colors and normals, and can be considered a sampling of a continuous surface. This representation is extremely simple and flexible. Moreover, it offers the additional advantage of avoiding connectivity information and topological consistency.

This paper investigates visibility of point clouds. One way to compute visibility of a point cloud is to reconstruct the surface [Hoppe et al. 1992; Bernardini et al. 1999; Curless and Levoy 1996; Carr et al. 2001; Amenta et al. 2001; Amenta et al. 2002; Amenta and Kil 2004; Fleishman et al. 2005] and determine visibility on the reconstructed triangular mesh. Reconstruction, however, is a difficult problem, both theoretically and implementation-wise, which often requires additional information, such as normals and sufficiently dense input.

The key question that this paper attempts to answer is how the visibility information can be directly extracted from a point cloud. Evidently, points cannot occlude one another (unless they accidentally fall along the same ray from the viewpoint), and therefore no point is actually hidden. However, once a surface is reconstructed from the points, it is certainly possible to determine which of the points are visible. This implies that a point cloud inherently contains information from which it is possible to extract the visibility of the points. The challenge is to skip the full reconstruction.

*e-mail: sagikatz@techunix.technion.ac.il
†e-mail: ayellet@ee.technion.ac.il
‡e-mail: ronen.basri@weizmann.ac.il

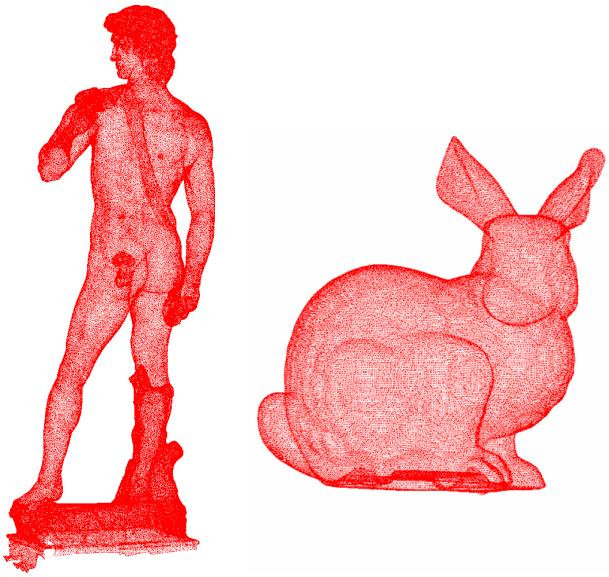


Figure 1: Input to the operator – Are the objects looking forwards or backwards?

Suppose that we are given a point cloud depicting an object, such as the statue of David or the bunny in Figure 1. If all the points are drawn, it is difficult to determine whether these objects are looking forwards or backwards. This paper describes an operator that computes the visibility directly from a point cloud. For instance, after applying the operator on the point clouds of Figure 1, it can be seen in Figure 2 that David (/bunny) is looking backwards. The points need not be associated with normal information and need not be sampled densely.

We show that the operator proposed in this paper is simple and fast. It can be described in just a handful of Matlab lines and its asymptotic complexity is $O(n \log n)$, where n is the number of points in the point cloud. Moreover, it can calculate visibility for dense as well as sparse point clouds, for which reconstruction or other methods, might be difficult. In addition, the correctness of the operator is proved in the limit and theoretical guarantees are provided for finite sampling.

Other benefits of the operator are that it does not depend on the screen resolution (since it operates in *object space*); a change in camera rotation or field of view does not require re-calculation of visibility; it works in various dimensions; and the viewpoint can be positioned either within or outside the point cloud.

Calculating visibility directly from a point cloud is an interesting problem in its own right. However, it can be utilized in a variety of applications. We show that it can be used to visualize point sets. Moreover, without additional cost, it can produce a view-dependent "quick-and-dirty" reconstruction online. Finally, realistic shadow casting can be achieved in object space in interactive time.

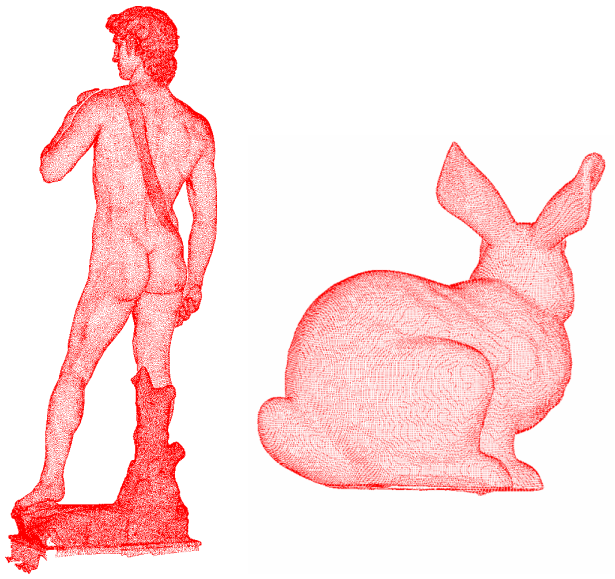


Figure 2: Output of the operator - They are heading backwards!

The contribution made in this paper is thus twofold: First, the paper presents a general, fast, and simple operator for determining visibility of points in various dimensions and proves some theoretical guarantees. Second, the paper demonstrates the utility of the operator in visualizing point clouds, in view-dependent reconstruction, and in shadow casting.

The rest of the paper is structured as follows. Section 2 briefly discusses related work. Section 3 describes the proposed operator. Section 4 proves some properties of the operator. Section 5 discusses implementation and demonstrates results and applications. Section 6 concludes the paper.

2 Related work

Visibility determination has been a basic problem in computer graphics from its early days [Appel 1968; Sutherland et al. 1974; Funkhouser et al. 1992; Greene et al. 1993; Bittner and Wonka 2003; Cohen-Or et al. 2003; Leyvand et al. 2003]. It is important in a variety of applications, including rendering, animation and simulation, security, and robotics. However, while most of the work in computer graphics determines the visibility between polygons, the purpose here is quite different. Our operator intends to find the points that would be visible, if the surface they are sampled from, existed.

Though computing the correct visibility is useful in various applications, in point-based representations it has been addressed mainly within rendering, where it is considered a major challenge [Sainz et al. 2004]. Here, visibility computation is usually performed during ray tracing [Wald and Seidel 2005]. Since both rays and points are singular primitives, this requires the algorithm to either trace "thick" rays [Schaufler and Jensen 2000] or use "finite-area" points [Rusinkiewicz and Levoy 2000; Dutré et al. 2000; Zwicker et al. 2001; Wu and Kobbelt 2004; Guennebaud et al. 2004; Guennebaud et al. 2004]. The most common approaches splat the points into a Z-buffer [Sainz and Pajarola 2004; Dachsbacher et al. 2003].

This paper attempts to solve the problem of visibility regardless of rendering. Moreover, the aim is to avoid the two assumptions that are made in most rendering papers – that the points satisfy sampling criteria, such as the Nyquist condition, and that the points are associated with normals (or that the normals can be estimated). In a couple of recent works, it becomes evident that it is important to be able to handle point clouds that are not well sampled and are not interpreted in any way (such as by meshing or estimating normals) [Wimmer and Scheiblauer 2006; Co 2006]. Finally, we wish to support visibility calculation by rigorous theoretical guarantees.

Another related, yet distinct problem is surface reconstruction from point clouds, which has received considerable attention. Different approaches have been proposed, such as representations by implicit functions [Hoppe et al. 1992; Carr et al. 2001; Ohtake et al. 2003], by Moving Least Squares [Alexa et al. 2003; Fleishman et al. 2005], employing Voronoi/Delaunay techniques [Amenta et al. 2002; Amenta et al. 2001; Mederos et al. 2005], and others. Some of the methods are supported by theoretical results, e.g. [Amenta et al. 2001], while others sacrifice theory and instead, optimize for high speed, e.g. [Ohtake et al. 2003].

The current paper attempts to determine visibility, while skipping the reconstruction phase. Nevertheless, we will show that our algorithm can be used for view-dependent reconstruction. In this case, both speed and theoretical support can be achieved.

3 The HPR operator

Given a set of points $P = \{p_i | 1 \leq i \leq n\} \subset \Re^D$, which is considered a sampling of a continuous surface S, and a viewpoint (camera position) C, our goal is to determine $\forall p_i \in P$ whether p_i is visible from C.

Straightforward solutions are bound to fail. Calculating the line-of-sight from C to p_i is not helpful, because, except for degenerate cases, a point is always visible. We therefore need to define when a point is considered visible. Obviously, a sensible criterion of visibility must relate to the density of the sampling. Suppose that P is a ρ-sample of S, i.e., if we surround each sample point $p_i \in P$ by an open ball of radius ρ, the surface S will be completely contained within the union of these balls. A simple definition of visibility then implies that p_i is visible if it does not become occluded by another point when we perturb its position anywhere within the ball. While this definition works well for surfaces that are perpendicular to the line of sight, it fails when the surface is oblique, since in this case, a small perturbation could make a point occluded by another point from the same surface (e.g., when the surface is planar). We could overcome this if we knew the normal to the surface at each point, but we want to avoid estimating the normal.

We seek an operator that has the following properties:

1. Correctness: in the limit, as the density $\rho \to 0$, a point p_i on S should be marked visible by the operator, if and only if it is indeed visible.
2. The operator should handle oblique surfaces, while avoiding to compute the surface normals locally.
3. The asymptotic complexity and the running time should be reasonable, even in software.

This section introduces an operator, denoted as the *hidden point removal (HPR)* operator, which satisfies the above requirements (proved in the next section). The operator consists of two steps: inversion and convex hull construction, discussed below.

1. Inversion: Given P and C, we associate with P a coordinate system, in which the viewpoint C is placed at the origin. We seek a function that maps a point $p_i \in P$ along the ray from C to p_i and is monotonically decreasing in $||p_i||$. ($||\cdot||$ is a norm.)

There are various ways to perform inversion. Here, we focus on *spherical flipping*, which was first presented in [Katz et al. 2005] in a different context. Consider a D-dimensional sphere with radius R, centered at the origin (C), and constrained to include all the points in P. Spherical flipping reflects a point $p_i \in P$ with respect to the sphere by applying the following equation:

$$\widehat{p_i} = f(p_i) = p_i + 2(R - ||p_i||)\frac{p_i}{||p_i||}. \quad (1)$$

Intuitively, spherical flipping reflects every point p_i internal to the sphere along the ray from C to p_i to its image outside the sphere, as illustrated in Figure 3.

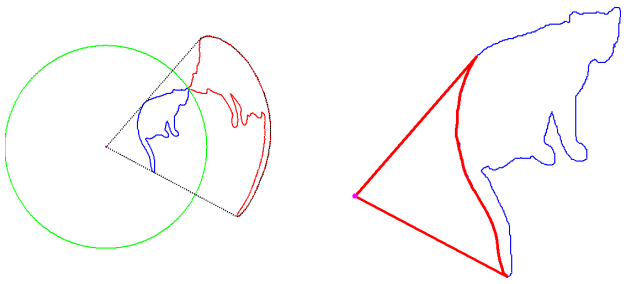


Figure 3: HPR Operator – Left: spherical flipping (in red) of a 2D curve (in blue) using a sphere (in green) centered at the view point (in magenta). Right: back projection of the convex hull. Note that this image is used only for illustration; in practice, R is much larger.

Note that there are other possible inversion functions. For instance, a function that seems to achieve roughly the same effect, is given by the following expression, where $\gamma > 1$ is a parameter and $||p_i|| < 1$:

$$\tilde{f}(p_i) = \frac{p_i}{||p_i||^{\gamma}}.$$

2. Convex hull construction: Denote by $\widehat{P}$ the transformed point cloud of P: $\widehat{P} = \{\widehat{p_i} = f(p_i) | p_i \in P\}$. Calculate the convex hull of $\widehat{P} \bigcup \{C\}$, i.e., the set that contains the transformed point cloud and the center of the sphere.

The main result of the paper is that extracting the points that reside on the convex hull of $\widehat{P} \bigcup \{C\}$ amounts to determining the visible points. (The inclusion of C is important since points on the back side of the object may otherwise lie on the convex hull, when C is external to P.) We state this as a definition and explain the intuition hereafter. In the next section we prove some properties.

Definition 3.1 *A point $p_i \in P$ is marked visible from C if its inverted point $\widehat{p_i}$ lies on the convex hull of $\widehat{P} \bigcup \{C\}$.*

The HPR operator can be applied in any dimension. However, it is best understood in 2D. Consider a point $p_i \in P$. Without loss of generality, p_i lies on the X-axis. Using a polar coordinate system (r, θ), we can write $p_i = (r_i, 0)$, where r_i is the distance of p_i from C, and the angle with the X-axis is 0. Consider the straight line $\widehat{L}$ that passes through $\widehat{p_i}$ and creates an angle β with the X-axis, as shown in Figure 4.

We wish to find the curve $L = (r(\alpha), \alpha)$, which is the source of $\widehat{L}$, i.e., the curve that is transformed to $\widehat{L}$ by spherical flipping. Using the Law of Sines we get:

$$\frac{2R - r_i}{\sin(\pi - \alpha - \beta)} = \frac{2R - r(\alpha)}{\sin\beta}. \quad (2)$$

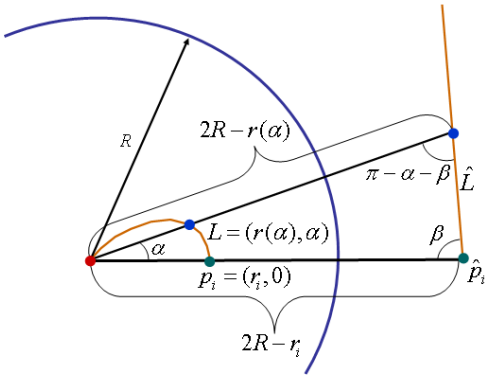


Figure 4: L is transformed to $\widehat{L}$ by spherical flipping.

Consequently,

$$L = (r(\alpha), \alpha) = (2R + \frac{(r_i - 2R)\sin\beta}{\sin(\alpha + \beta)}, \alpha). \quad (3)$$

L passes through both p_i and C. In Cartesian coordinates L is expressed by a quartic polynomial in x and y. Figure 5 illustrates how the shape of L changes as a function of angle β. The region bounded by L and the X-axis gets smaller as β gets larger.

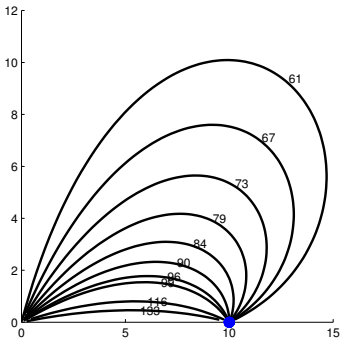


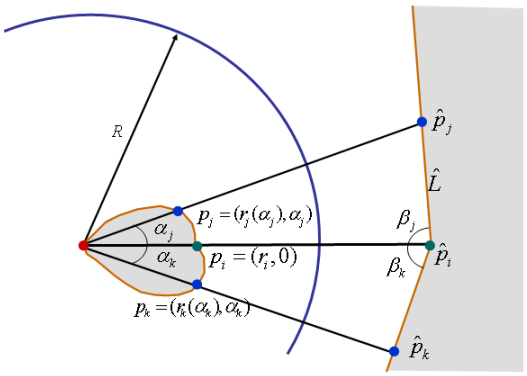
Figure 5: The shape of L for different values of β (in degrees), where $p_i = (10, 0)$, $R = 30$.

L and the X-axis define the empty region associated with p_i. "How much" p_i is visible, depends on the size of the region. The larger the size, the "more visible" p_i is. The important property of HPR is that this size is adaptively determined by p_i's neighboring points, as explained below.

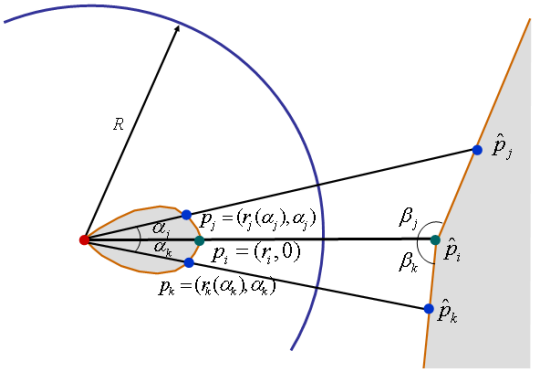
For every given point p_i, there exist two special points on either side of P, p_j and $p_k \in P$. The region bounded between the curves in Eq. 3, $L_j = (r_j(\alpha_j), \alpha_j)$ from p_i through p_j and $L_k = (r_k(\alpha_k), \alpha_k)$ from p_i through p_k, is the largest possible empty region, as demonstrated in Figure 6.

From Equation 3, it can be deduced that the largest region corresponds to the smallest β. This means that β_j and β_k that correspond to the largest possible empty region, are the smallest possible for p_i. Note that β_j and β_k can be extracted from Equation 3.

For p_i to be visible, the sum of β_j and β_k should satisfy $\beta_j + \beta_k \leq$ *const* (i.e., a large empty region is associated with p_i). In effect, this



(a) p_i is visible



(b) p_i is hidden

Figure 6: The empty gray region between L_j and L_k, as defined by the values of $\beta_j+\beta_k$.

condition defines a threshold on the size of the region for which the point is considered visible.

Setting $const = \pi$ means that computationally there is no need to find for each point the neighboring points p_j and p_k that maximize the empty region size. Instead, it suffices to calculate the convex hull of $\widehat{P\bigcup\{C\}}$. This is so because point $\widehat{p_i}$ is on the convex hull of $\widehat{P\bigcup\{C\}}$ if and only if all the points of $\widehat{P\bigcup\{C\}}$ reside to one side of the half-spaces defined by $\overline{\widehat{p_i},\widehat{p_j}}$ and $\overline{\widehat{p_i},\widehat{p_k}}$. In our case, since the region between L_j and L_k is empty, the region on the far side of $\widehat{L_j}$ and $\widehat{L_k}$ must be empty (the gray regions in Figure 6).

This observation is important computationally and is the reason why the HPR operator is so efficient. Without the computation of the convex hull, p_j and p_k would have to be found $\forall p_i \in P$, making the algorithm quadratic (and in 3D even cubic, although it can potentially be sped up by using fast nearest neighbor techniques). Instead, all that needs to be done is to compute the convex hull and consider a point p_i visible if $\widehat{p_i}$ is on the convex hull of $\widehat{P}$. Thus, the HPR operator defines both the shape and the size of the empty region, $\forall p_i \in P$.

The above explanation can be extended to 3D. In this case, the empty region between p_i and C is defined by a 3D surface enclosing a volume, rather than a curve $(L_j \cup L_k)$ enclosing an area. Also, in 3D, instead of using two neighboring points, at least three neighboring points define the surface enclosing the empty volume. However, our solution that uses the convex hull remains the same.

It is worth mentioning that even though π is a constant threshold, the threshold for visibility can be indirectly modified by changing R, the radius of the sphere. A larger R relaxes the visibility condition and more points are considered visible. Moreover, in the general case, when different families of inversion functions are used, the above explanation will remain the same, while the shape of L will change.

4 Properties of the HPR operator

This section presents and proves some properties of the operator. It addresses the theoretical guarantees of the operator, the factors that influence the choice of the parameter R, and the complexity. While we focus here on a particular inversion function spherical flipping, similar results can be derived for other, related families of inversion. The theorems and lemmas are stated and explained in the section, while the proofs can be found in the Appendix.

Throughout the section we use the notion of density, which is formulated below.

Definition 4.1 sample density (density): *A sample $P \subseteq S$ is a $\rho-$sample from surface S if* $\forall q \in S\ \exists p \in P$ *s.t.* $|q-p| < \rho$.

The first issue concerns the correctness of the operator in the case that the surface itself is given (i.e., P is a 0-sample of S). It is shown below that in this case, every point marked as visible by the operator (Definition 3.1), is indeed visible. Moreover, in the limit, when $R \to \infty$, every visible point will be marked visible.

The next issue relates R to the local curvature that permits visibility. This provides a local analysis of visibility, in which we consider occlusion by (infinitesimally) close points on the surface, while disregarding occlusions by remote points. Specifically, it is shown that given R, all the convex points, as well as concave points with sufficiently small curvature, may be marked visible by our operator. This implies in particular correctness for convex surface patches and for slanted planar surfaces at any value of R.

The third issue regards theoretical guarantees when the given set of points P is a ρ-sample of S with $\rho > 0$. Since in this case it can no longer be true that every point marked by the operator as visible is indeed visible, a more realistic visibility is defined, denoted as $\varepsilon-$visibility.

Definition 4.2 $\varepsilon-$visible: *A point $p \in P$ is $\varepsilon-$visible if* $\exists q \in \Re^D$ *s.t.* $|q-p| < \varepsilon$ *and q is visible from C. In other words, moving p in a distance shorter than ε will make it visible.*

It is proved that for every R, there exists an ε for which every point that the operator marks is $\varepsilon-$visible. Moreover, with certain restrictions, for every $\varepsilon > 0$, there exists a choice of R that guarantees $\varepsilon-$visibility.

Finally, the section discusses the choice of R and the complexity of the operator.

Correctness when $\rho = 0$: The next two lemmas assume that the input to the operator is a surface S, a viewpoint C, and a radius R. It is further assumed that there exists a gap T between the viewpoint and the object: $T = \inf\{\|p-C\| | p \in S\} > 0$. (This assumption is essential because points that are very close to C may occlude extremely large sections of the space.) Let $V \subseteq S$ be the set of visible points from C and $H_R \subseteq S$ be the set of points marked visible by the operator. The two lemmas imply that the operator is conservative and converges to the correct solution as R approaches infinity.

Lemma 4.1 *$H_R \subseteq V$, i.e., every point marked visible by the HPR operator is indeed visible from C.*

Lemma 4.2 $\lim_{R\longrightarrow\infty} H_R = V$, *i.e., assuming* $T > 0$, *when* $R \to \infty$, *the set of visible points marked by HPR is equal to the set of visible points.*

R and the local curvature: For a finite value of R, we can further analyze which points will be marked visible by the HPR operator, by considering the influence of the curvature on the results. We start again with the intuition. It is straightforward to see that oblique planar surfaces are correctly handled by the HPR operator, since spherical flipping maps such surfaces to convex structures. Handling concave sections of a surface, in contrast, is affected by the local curvature. Below, we provide a derivation of the permissible curvature as a function of the radius R, the distance r from the point p to the viewpoint C, and the orientation of the convex hull through $\widehat{p}$, β (Figure 4). The derivation is general, yet a particularly simple expression is obtained when the tangent to a point is perpendicular to the line of sight from this point.

Lemma 4.3 *Let S be an infinitesimal surface patch around p. Then* $p \in H_R$ *if and only if the curvature k at p satisfies:*

$$k < \frac{4R(2R-r)\cot^2\beta + 2Rr}{\left(4Rr - 4R^2 + \frac{(r-2R)^2}{\sin^2\beta}\right)^{3/2}}.$$

In the case that $\beta = \pi/2$, *which corresponds to the case that the tangent to the surface at p is perpendicular to the line of sight,* $k < \frac{2R}{r^2}$.

This implies in particular that convex shapes and slanted planes are correctly handled for any choice of R, and that points on concave sections of a surface are handled correctly as long as the curvature is sufficiently low (except when remote sections of the surface happen to fall close to the line of sight through those points). Note that in higher dimensions *all* sectional curvatures must not exceed the bound, i.e., this bound is on the maximal curvature. The case that a patch is perpendicular to the line of sight also demonstrates that the permissible curvature grows with R. Thus, as R increases, more points become visible, until all (truly visible) points become visible by the HPR operator.

Theoretical guarantees $\rho > 0$: In the rest of the section, it is assumed that the given set of points P is a ρ-sample of S with $\rho > 0$. Recall that a point is $\varepsilon-$visible, if moving it by ε will make it visible. Using this definition, it is possible to extend the correctness lemmas stated above to the more practical case of the given data.

Assuming that the sample is sufficiently dense, we show that for every R, there exists an ε, such that every point marked visible by the operator is $\varepsilon-$visible. Moreover, for sufficiently large ε, there exists R, such that every point marked visible by the operator is ε-visible.

Let $V_\varepsilon \subseteq P$ be the set of ε-visible points from C (points visible in S). As before, we assume that the distance of S to C is at least $T > 0$.

Theorem 4.4 *Assume that the sample is sufficiently dense, then for every R, there exists* $\varepsilon > 0$ *such that* $H_R \subseteq V_\varepsilon$.

Theorem 4.5 *Assume that the sample is sufficiently dense, then for sufficiently large* $\varepsilon > 0$, *there exists* $R > 0$ *such that* $H_R \subseteq V_\varepsilon$.

The proofs of these theorems imply that for a constant value of R, as ρ decreases, a smaller value of ε is obtained.

Choosing R: The proofs of the above theorems show the relation between the density ρ, R, and ε-visibility. In particular, these factors are essential for choosing a suitable R.

As R increases, more points pass the threshold of the convex hull and hence are marked visible. For instance, as $R \to \infty$, r_i in Eq. 3 becomes negligible, $\beta_j, \beta_k \to \pi/2$, and all the points are marked visible. This is so because they are transformed by spherical flipping to a sphere with an infinite radius and thus reside on the convex hull. Therefore, a large R is suitable for dense point clouds, while a small R is suitable for sparse clouds.

This is illustrated in Figure 7, where the percent of false positives and false negatives are plotted as a function of $\log(R)$. With small R, points visible in S may be marked non-visible by HPR, whereas with large R, non-visible points may be marked visible. This is also illustrated in Figure 8. A limitation of the algorithm is that even when using the optimal R, a few misclassified points, mostly near the silhouettes and deep concavities, might remain. However, the number of such points decreases with ρ.

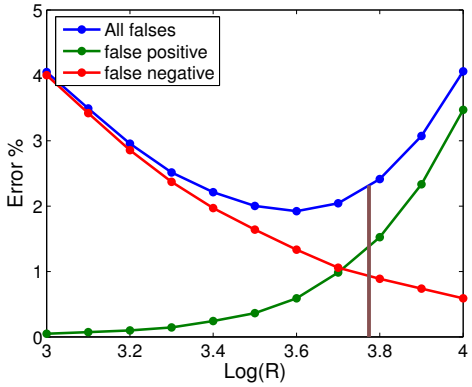


Figure 7: False positives/negatives and their sum, of a specific model (Bimba, 70K points). The automatically calculated R is shown in brown.

The permissible curvature (Lemma 4.3) can be used to determine an upper bound on R. Assume an object of thickness d is positioned at a distance r from C. Consider the parabola $y = r - (d/\rho^2)x^2$, whose apex is at the back of the object, which creates an opening of 2ρ (the expected distance between two sample points) on the frontal surface of the object. The curvature at the apex $x = 0$ is $2d/\rho^2$. Comparing this to the permissible curvature $2R/r^2$, it is concluded that R can be bounded: $R < dr^2/\rho^2$.

In our experiments, R is determined automatically as follows. An additional viewpoint, opposite to the current viewpoint on the line connecting the original viewpoint to the object's center of mass, is set. Then, R is determined by maximizing the number of disjoint points that are considered visible by both viewpoints. Gradient descent optimization is used [Forsythe et al. 1977]. The intuition is that no point should be visible simultaneously to both viewpoints. Figure 7 illustrates that the computed R is very close to the optimum (the minimum value of the blue curve).

Complexity: Finally, the operator presented is very efficient. Let n be the number of points in the point cloud. The first stage of the operator, spherical flipping, takes $O(n)$. The second stage, convex hull computation, takes $O(n\log n)$ for point sets in 2 and 3-dimensions. Therefore, the asymptotic complexity of the operator is $O(n\log n)$.

5 Implementation and applications

A major advantage of the HPR operator is that it is extremely simple to implement. Algorithm 1 shows the Matlab code that implements the HPR operator. (Matlab itself calls the Qhull algorithm [Barber

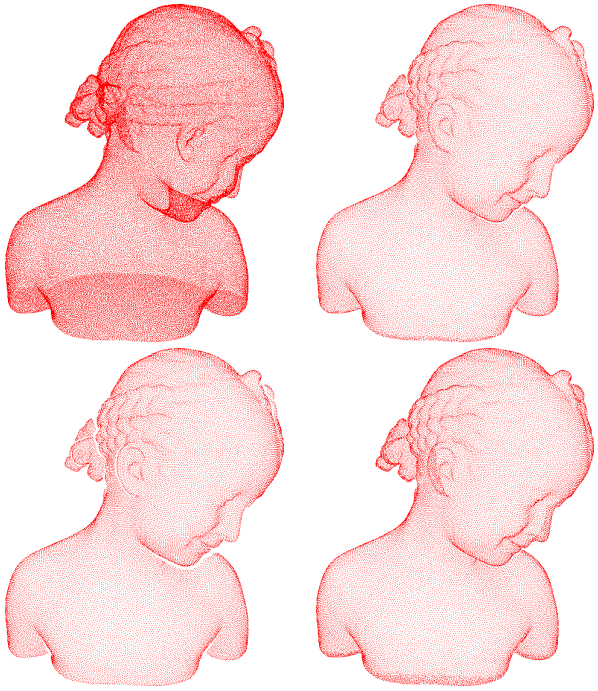


Figure 8: Top: The original point cloud (left) and the result using the automatically calculated R ($\log(R) = 3.77$) (right). Bottom: Results generated using smaller R ($\log(R) = 3$) lead to an increased number of false negatives (missing points below the chin), while results generated using larger R ($\log(R) = 4$) lead to an increased number of false positives (excessive points at the bottom of the chest).

et al. 1996].) The code is very simple and short and works for D-dimensional point clouds. The only parameter that the operator gets is used to compute radius R. This parameter can be either set by the user, or R can be computed automatically, as discussed in Section 4.

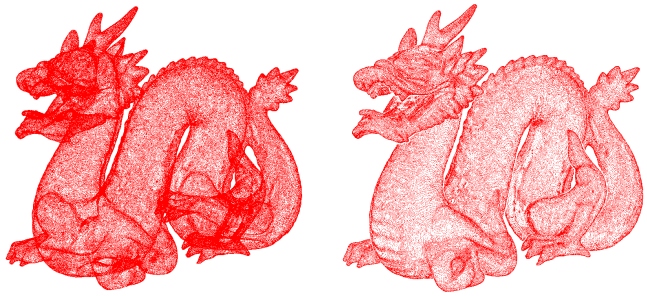
Algorithm 1 Matlab code for HPR

```
function visiblePtInds=HPR(p,C,param)
dim=size(p,2);
numPts=size(p,1);
% Move the points s.t. C is the origin
p=p-repmat(C,[numPts 1]);
% Calculate ||p||
normp=sqrt(dot(p,p,2));
% Sphere radius
R=repmat(max(normp)*(10^param),[numPts 1]);
%Spherical flipping
P=p+2*repmat(R-normp,[1 dim]).*p./repmat(normp,[1 dim]);
%convex hull
visiblePtInds=unique(convhulln([P;zeros(1,dim)]));
visiblePtInds(visiblePtInds==numPts+1)=[];
```

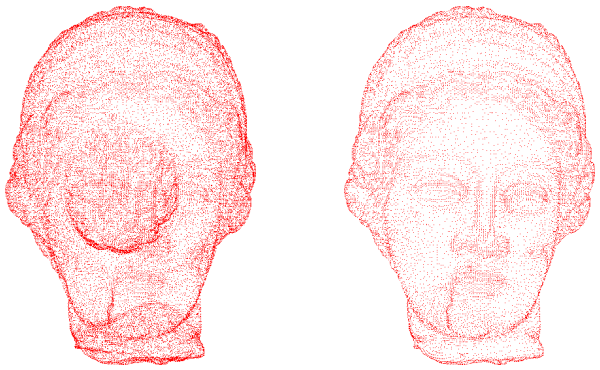
Determining the visibility of point clouds can potentially be utilized in visualization, reconstruction, shadow casting, rendering, camera placement, etc. We illustrate the usefulness of the HPR operator in three applications: visualization of point clouds, view-dependent reconstruction, and shadow casting.

5.1 Visualizing point clouds

Visualizing the raw data is important during long scanning sessions, in CAD, in simulations of scientific visualization, etc. Figures 9–10 show several results of the HPR operator, which is applied to well-known scanned point clouds. The point sets are rendered before and after applying the operator.



(a) dragon (437k points)



(c) Igea (non-uniform sampling, 33.6k points)

Figure 9: Point cloud visualization before (left) & after (right) HPR.

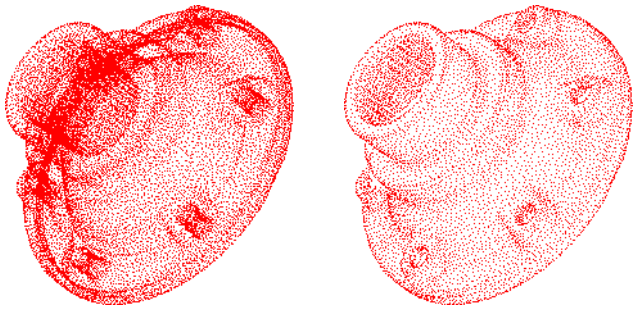
Before applying the operator, it is hard to distinguish between two possible positions that produce very similar projections – looking towards or away from the camera (up to small differences due to perspective projection). This problem is resolved using HPR. For instance, while the original point set of Igea shows both the scar and the hairdo and it is hard to say whether the statue looks forwards or away, after applying the HPR operator, only the scar shows and it is obvious that Igea is looking towards the camera.

Figure 11 shows the result of the operator, when applied to sparse point clouds. This result demonstrates well the strength of the operator, where the alternative of fully reconstructing the surface for determining visibility, might fail. Note, for instance, how only the visible subset of each ring of points on the jet fighter is left when the HPR is used.

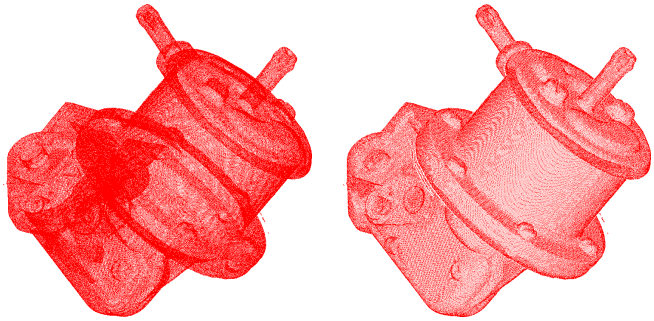
The operator takes up to a few seconds to run on large models, on Intel Core2, 2.14Ghz, 1Gb RAM. The calculation ranges between 23 milliseconds for the jet-fighter (2370 points), 1.3 seconds for David (258K points), and 3.65 seconds for the oil pump (542k points).

5.2 View-dependent reconstruction

Surface reconstruction from point clouds has received considerable attention in recent years. It is described in a variety of papers, and generates pretty results [Hoppe et al. 1992; Curless and Levoy

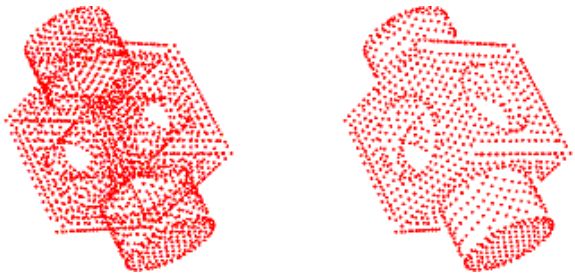


(a) carter (25k points)

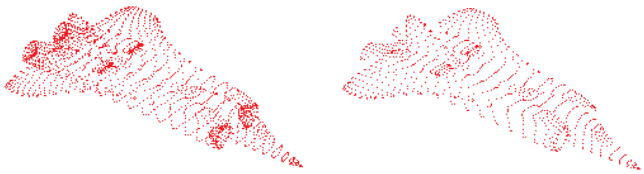


(b) oil pump (542k points)

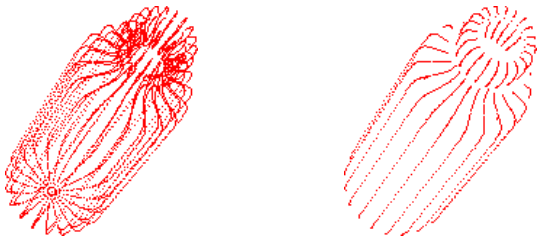
Figure 10: Point cloud visualization before (left) & after (right) HPR.



(a) sparse block (2132 points)



(a) jet-fighter (2370 points)



(b) bottle (5540 points)

Figure 11: Visualization of sparse models before & after HPR.

1996; Bernardini et al. 1999; Amenta et al. 2001; Adamson and Alexa 2003; Ohtake et al. 2003; Amenta and Kil 2004; Mederos et al. 2005; Fleishman et al. 2005; Wald and Seidel 2005]. However, the algorithms are often not simple to implement and both their running times and their asymptotic complexities might be high.

Instead of fully reconstructing the surface, we propose a view-dependent on-the-fly reconstruction, which provides a "quick-and-dirty" visualization of the surface from which the points are sampled, as illustrated in Figures 12– 13.

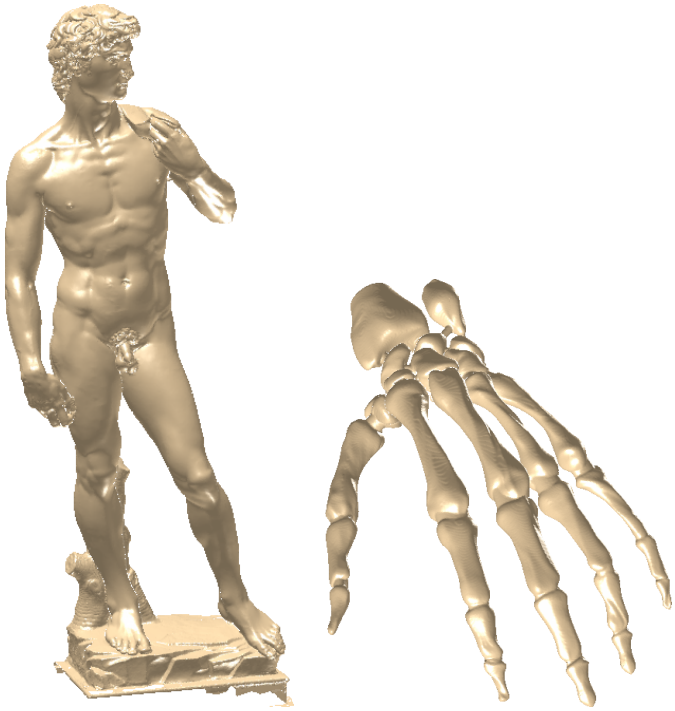


Figure 12: "Quick-and-dirty" view-dependent reconstruction of David (258K points) and the skeletal hand (327K points).

Figure 13: Two different view-dependent reconstruction of Bimba.

View-dependent reconstruction is performed by displaying not only the points residing on the convex hull of $\widehat{P} \cup C$, as described so far, but also the triangles the convex hull consists of. Long artifact triangles are eliminated using a threshold on the edge length, as illustrated in Figure 14. It is important to note that the reconstruction does not increase the complexity of the algorithm, since the convex hull is computed anyway.

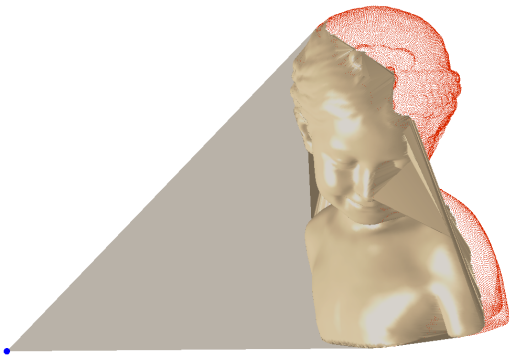


Figure 14: Reconstruction before the removal of the artifact triangles (with edges longer than 2.5% of the diameter). The blue point is the viewpoint of Figure 13(a).

Placing additional viewpoints around the original viewpoint can improve the results around the silhouettes. A point is then considered visible when it is visible from either of the viewpoints.

Though the results are not full reconstructions, they certainly suffice for quickly perceiving the surface the points represent, and they are produced very efficiently. For comparison, the algorithm of [Mederos et al. 2005] has $O(n^2)$ complexity and takes a minute to run on a 180,000 point cloud on a 2.4GHz PC, while our algorithm has $O(n \log n)$ complexity and takes less than a second to run the same size point cloud on a 2.14Ghz Intel Core2. Similarly, applying MPU interpolation [Ohtake et al. 2003] to the Buddha point cloud (543K) takes 6:53 minutes on a 1.6GHz mobile, while our operator takes 4.15 seconds.

5.3 Shadow casting

Another application of the operator is shadow casting [Woo et al. 1990; Hasenfratz et al. 2003]. Using HPR, it is possible to demonstrate realistic shadow casting for meshes, in interactive time, in Matlab. The shadow casting is calculated in object space rather than in screen space, thus it depends neither on screen resolution nor on the z-buffer accuracy.

Given a mesh, shadow casting is computed by assigning the center of the sphere C to the position of the light and applying HPR to the mesh vertices. A brightness value is assigned to each mesh vertex according to its calculated visibility. A visible vertex is given a high brightness value and a non-visible vertex is given a low brightness value. To produce soft shadows, these values are smoothed, such that the brightness value is affected by the neighboring vertices. For the final rendering, the brightness value is interpolated along the faces. (Note though that only the vertices are used for calculating the brightness values.)

Figure 15 shows a couple of results of shadow casting. In these examples, specular lighting is turned off and a single light source is used. There is no limitation, however, on the number of light sources that can be used. Note that the method works even when there are holes, where the light penetrates the holes.

Figure 16 compares the results of shadow casting, as achieved by applying the HPR operator, to those computed exactly. In the latter case (exact shadows), the intersection of a ray from the vertex to the light source is calculated, for each vertex. If the ray intersects the surface, the vertex is shadowed.

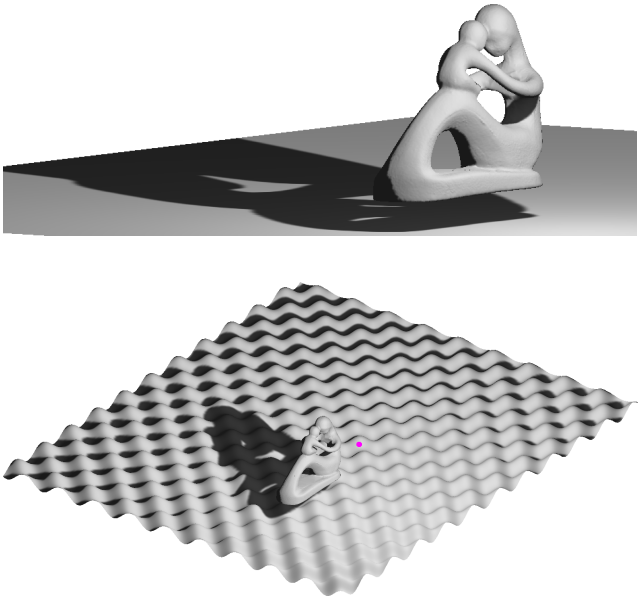


Figure 15: Shadow casting in software using HPR.

6 Conclusion

This paper proposes a simple and fast hidden point removal operator, which determines the visible points from a given viewpoint. The operator is provably correct in the limit and theoretical guarantees are given for the finite-sampling case. It can be applied to dense as well as sparse point clouds in various dimensions. The operator can be easily implemented using existing libraries. It runs in $O(n \log n)$, where n is the number of points in the point cloud.

The paper demonstrates that without additional cost, this operator can be used not only for visualizing point clouds, but also for view-dependent reconstruction and for shadow casting. We believe that other applications may also benefit from the operator.

In the future, we intend to investigate visibility under motion. The goal is to construct data structures that facilitate the computation of visibility when a subset of the points changes or when the viewpoint changes, e.g., by representing points according to their distance to the convex hull. One possible direction is the use of kinetic convex hulls [Abam and de Berg 2005].

Acknowledgments: The models of David, Bunny, and Dragon are courtesy of the Digital Michelangelo Project 3D Model Repository and the Stanford 3D Scanning Repository. The model of Igea is courtesy of Cyberware. The models of Bimba, Fertility, Gargoyle, Oil pump, and Carter are provided courtesy of INRIA, IMATI, UU, VCG-ISTI, and ISTI by the AIM@SHAPE Shape Repository. This research has been supported in part by the European Community grant IST-2002-506766 Aim@Shape. The vision group at the Weizmann Institute is supported in part by the Moross Foundation. The graphics group at the Technion is supported by the Fund for the Promotion of Research.

The technology described in this paper is patent pending.

References

ABAM, M., AND DE BERG, M. 2005. Kinetic sorting and kinetic convex hulls. In *Twenty-First Annual Symposium on Computational geometry*, 190–197.

Figure 16: Comparison between the HPR approximation (left) and the exact computation (right) of the lit vertices.

ADAMSON, A., AND ALEXA, M. 2003. Approximating and intersecting surfaces from points. In *Eurographics/ACM SIGGRAPH symposium on Geometry processing*, 230–239.

ALEXA, M., BEHR, J., COHEN-OR, D., FLEISHMAN, S., LEVIN, D., AND SILVA, C. 2003. Computing and rendering point set surfaces. *IEEE Trans. on Vis. and Computer Graphics 9*, 1, 3–15.

ALEXA, M., GROSS, M., PAULY, M., PFISTER, H., STAMMINGER, M., AND ZWICKER, M. 2004. Point-based computer graphics. In *SIGGRAPH course notes*.

AMENTA, N., AND KIL, Y. 2004. Defining point-set surfaces. *ACM Trans. Graph. 23*, 3, 264–270.

AMENTA, N., CHOI, S., AND KOLLURI, R. 2001. The power crust, unions of balls, and the medial axis transform. *Int. J. of Computational Geometry and its Applications 19*, 2-3, 127–153.

AMENTA, N., CHOI, S., DEY, T. K., AND LEEKHA, N. 2002. A simple algorithm for homeomorphic surface reconstruction. In *Int. J. Comput. Geom. Appl.*, vol. 12, 125–141.

APPEL, A. 1968. Some techniques for shading machine renderings of solids. In *AFIPS Spring Joint Computer Conf.*, vol. 32, 37–45.

BARBER, C. B., DOBKIN, D. P., AND HUHDANPAA, H. 1996. The quickhull algorithm for convex hulls. *ACM Trans. Math. Softw. 22*, 4, 469–483.

BERNARDINI, F., MITTLEMAN, J., RUSHMEIER, H., SILVA, C., AND TAUBIN, G. 1999. The ball-pivoting algorithm for surface reconstruction. *IEEE Transactions on Visualization and Computer Graphics 5*, 4.

BITTNER, J., AND WONKA, P. 2003. Visibility in computer graphics. *Environment and Planning B: Planning and Design 30*, 5, 729–756.

CARR, J. C., BEATSON, R. K., CHERRIE, J. B., MITCHELL, T. J., FRIGHT, W. R., MCCALLUM, B. C., AND EVANS, T. R. 2001. Reconstruction and representation of 3D objects with radial basis functions. In *SIGGRAPH*, 67–76.

CO, C. 2006. *Meshless Methods for Volume Visualization*. PhD thesis, University of California, Davis.

COHEN-OR, D., CHRYSANTHOU, Y., SILVA, C., AND DURAND, F. 2003. A survey of visibility for walkthrough applications. *IEEE Trans. on Vis. and Computer Graphics 9*, 3, 412–431.

CURLESS, B., AND LEVOY, M. 1996. A volumetric method for building complex models from range images. In *SIGGRAPH*, ACM Press, New York, NY, USA, 303–312.

DACHSBACHER, C., VOGELGSANG, C., AND STAMMINGER, M. 2003. Sequential point trees. *ACM Trans. Graph. 22*, 3, 657–662.

DE BERG, M., VAN KREVELD, M., OVERMARS, M., AND SCHWARZKOPF, O. 1997. *Computational geometry: algorithms and applications*. Springer-Verlag New York, Inc., NJ, USA.

DUTRÉ, P., TOLE, P., AND GREENBERG, D. 2000. Approximate visibility for illumination computations using point clouds. Tech. Rep. PCG-00-01, Cornell University, June.

FLEISHMAN, S., COHEN-OR, D., ALEXA, M., AND SILVA, C. 2003. Progressive point set surfaces. *ACM Trans. Graph. 22*, 4, 997–1011.

FLEISHMAN, S., COHEN-OR, D., AND SILVA, C. 2005. Robust moving least-squares fitting with sharp features. *ACM Trans. Graph. 24*, 3, 544–552.

FORSYTHE, G., MALCOLM, M., AND MOLER, C. 1977. *Computer Methods for Mathematical Computations*. Prentice Hall.

FUNKHOUSER, T., SQUIN, C., AND TELLER, S. 1992. Management of large amounts of data in interactive building walkthroughs. *Symposium on Interactive 3D Graphics 25*, 2, 11–20.

GREENE, N., KASS, M., AND MILLER, G. 1993. Hierarchical z-buffer visibility. In *SIGGRAPH*, 231–238.

GUENNEBAUD, G., BARTHE, L., AND PAULIN, M. 2004. Deferred splatting. *Comput. Graph. Forum 23*, 3, 653–660.

HASENFRATZ, J.-M., LAPIERRE, M., HOLZSCHUCH, N., AND SILLION, F. 2003. A survey of real-time soft shadows algorithms. In *Eurographics State-of-the-Art Reports*.

HOPPE, H., DEROSE, T., DUCHAMP, T., MCDONALD, J., AND STUETZLE, W. 1992. Surface reconstruction from unorganized points. *Computer Graphics 26*, 2, 71–78.

KATZ, S., LEIFMAN, G., AND TAL, A. 2005. Mesh segmentation using feature point and core extraction. *The Visual Computer 21*, 8-10, 865–875.

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

LEYVAND, T., SORKINE, O., AND COHEN-OR, D. 2003. Ray space factorization for from-region visibility. *ACM Transactions on Graphics (TOG) 22*, 3, 595–604.

MEDEROS, B., AMENTA, N., VEHLO, L., AND DE FIGUEIREDO, L. 2005. Surface reconstruction from noisy point clouds. In *Eurographics Symposium on Geometry Processing*, 53–62.

OHTAKE, Y., BELYAEV, A., ALEXA, M., TURK, G., AND SEIDEL, H.-P. 2003. Multi-level partition of unity implicits. *ACM Trans. Graph. 22*, 3, 463–470.

PAULY, M., AND GROSS, M. 2001. Spectral processing of point-sampled geometry. In *SIGGRAPH*, 379–386.

RUSINKIEWICZ, S., AND LEVOY, M. 2000. Qsplat: A multiresolution point rendering system for large meshes. In *SIGGRAPH*, 343–352.

SAINZ, M., AND PAJAROLA, R. 2004. Point-based rendering techniques. *Computers & Graphics 28*, 6, 869–879.

SAINZ, M., PAJAROLA, R., AND LARIO, R. 2004. Points reloaded: Point-based rendering revisited. In *Symposium on Point-Based Graphics*, 121–128.

SCHAUFLER, G., AND JENSEN, H. 2000. Ray tracing point sampled geometry. In *Eurographics Workshop on Rendering Techniques*, 319–328.

SUTHERLAND, E., SPROULL, R., AND SCHUMACKER, R. 1974. A characterization of ten hidden-surface algorithms. *ACM Comput. Surv. 6*, 1, 1–55.

WALD, I., AND SEIDEL, H.-P. 2005. Interactive ray tracing of point-based models. In *Eurographics Symposium on Point-Based Graphics*, 1–8.

WIMMER, M., AND SCHEIBLAUER, C. 2006. Instant points: Fast rendering of unprocessed point clouds. In *Proceedings Symposium on Point-Based Graphics 2006*, 129–136.

WOO, A., POULIN, P., AND FOURNIER, A. 1990. A survey of shadow algorithms. *IEEE Comput. Graph. Appl. 10*, 6, 13–32.

WU, J., AND KOBBELT, L. 2004. Optimized sub-sampling of point sets for surface splatting. *Computer Graphics Forum 23*, 643–652.

ZWICKER, M., PFISTER, H., VAN BAAR, J., AND GROSS, M. 2001. Surface splatting. In *SIGGRAPH*, 371–378.

ZWICKER, M., PAULY, M., KNOLL, O., AND GROSS, M. 2002. Pointshop 3D: an interactive system for point-based surface editing. *ACM Trans. Graph. 21*, 3, 322–329.

A Proofs of the operator's properties

This appendix provides the proofs of the properties of the operator.

Lemma 4.1 *$H_R \subseteq V$, i.e., every point marked visible by the HPR operator is indeed visible from C.*

Proof: Let $p \in H_R$. Suppose, by way of contradiction, that $p \notin V$. Then, the ray from C to p passes through some point $p' \in S$ that hides p. After inversion, p' will be farther away than p from C on this ray, since flipping is strictly monotonically decreasing along each ray from C. Thus, p is internal to the convex hull. □

Lemma 4.2 *$\lim_{R\longrightarrow\infty} H_R = V$, i.e., assuming $T = \inf\{\|p-C\| | p \in S\} > 0$, when $R \to \infty$, the set of visible points marked by HPR is equal to the set of visible points.*

Proof: One side of the equality was proved in Lemma 4.1. To prove the other side, we will show that if $p \in V$, then $p \in \lim_{R\longrightarrow\infty} H_R$. Without loss of generality, let $p = (r,0)$ (in spherical coordinates), i.e., p lies on the $X-$axis. Recall that we assume that $\forall q = (r_q, \theta) \in S$, $r, r_q \geq T$.

Then, applying spherical flipping to p and another arbitrary point $q \in S$, we get $f(p) = (2R-r, 0)$ and $f(q) = (2R-r_q, \theta)$, $(\theta \neq 0)$. To show that $p \in \lim_{R\longrightarrow\infty} H_R$, we will show that there exists R_0 such that $\forall R > R_0$, $f(q)$ is on one side of a line through $f(p)$, $\forall q \in S$. The line we choose is $x = 2R - r$, which is parallel to the $y-$axis.

Now, the x coordinate of $f(q)$ is $q_x = (2R - r_q)\cos\theta$, but since $r_q > T$, then $q_x < (2R-T)\cos\theta$. For sufficiently large R this quantity satisfies $(2R-T)\cos\theta < 2R - r$. This happens when $2R(1-\cos\theta) > r - T\cos\theta$, i.e., $R > \frac{r - T\cos\theta}{2(1-\cos\theta)}$, which holds since both the numerator and the denominator are positive. □

Lemma 4.3 *Let S be an infinitesimal surface patch around p. Then $p \in H_R$ if and only if the curvature k at p satisfies:*

$$k < \frac{4R(2R-r)\cot^2\beta + 2Rr}{\left(4Rr - 4R^2 + \frac{(r-2R)^2}{\sin^2\beta}\right)^{3/2}}.$$

In the case that $\beta = \pi/2$, which corresponds to the case that the tangent to the surface at p is perpendicular to the line of sight, $k < 2R/r^2$.

Proof: Let $\widehat{p} = f(p) = (\widehat{x}, \widehat{y})$ denote the spherical image of p and $\widehat{L}$ be the line through $\widehat{p}$ along the convex hull. WLOG, we define a coordinate system as follows (Figure 17): The vantage point C is at the origin; the Y-axis is parallel to $\widehat{L}$; and the X-axis is directed perpendicular to the Y-axis. p is given by (x,y) in Euclidean coordinates and (r,θ) in polar coordinates. Further, $\theta = \tan^{-1}(y/x)$, therefore $\theta = \beta - \pi/2$. Finally, we can relate these quantities by

$$(x,y) = (2R\cos\theta - \widehat{x}, 2R\sin\theta - \widehat{x}\tan\theta).$$

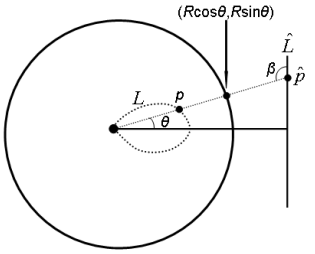


Figure 17: The figure shows line $\widehat{L}$ parallel to the Y-axis and its flipped source L through p. The curvature k_L of L at p is the maximal curvature beyond which p is marked invisible by HPR.

We are interested in curve L, which is the flip source of $\widehat{L}$ and its curvature k_L. p will be marked visible if the curvature k of S at p is smaller than k_L and marked hidden otherwise. Varying x and y along L, and taking their derivatives with respect to θ yields:

$$\begin{aligned}
\dot{x} &= -2R\sin\theta, \\
\dot{y} &= 2R\cos\theta - \frac{\widehat{x}}{\cos^2\theta}, \\
\ddot{x} &= -2R\cos\theta, \\
\ddot{y} &= -2R\sin\theta - \frac{2\widehat{x}\sin\theta}{\cos^3\theta}.
\end{aligned}$$

Using the standard formula for curvature:

$$k_L = \frac{\dot{x}\ddot{y} - \dot{y}\ddot{x}}{(\dot{x}^2+\dot{y}^2)^{3/2}} = \frac{4R^2 + \frac{4R\widehat{x}\sin^2\theta}{\cos^3\theta} - \frac{2R\widehat{x}}{\cos\theta}}{(4R^2 - \frac{4R\widehat{x}}{\cos\theta} + \frac{\widehat{x}^2}{\cos^4\theta})^{3/2}}.$$

Expressing this in terms of r and β, using the identities

$$\begin{aligned}\beta &= \frac{\pi}{2}+\theta,\\ x &= r\cos\theta = r\sin\beta,\\ y &= r\sin\theta = -r\cos\beta,\\ \widehat{x} &= 2R\cos\theta - x = (2R-r)\sin\beta,\end{aligned}$$

we obtain

$$k_L = \frac{4R(2R-r)\cot^2\beta + 2Rr}{\left(4Rr - 4R^2 + \frac{(r-2R)^2}{\sin^2\beta}\right)^{3/2}}.$$

In the special case that $\beta = \pi/2$ this simplifies to $k_L = \frac{2R}{r^2}$. □

Theorem 4.4 *Assume that the sample is sufficiently dense, then for every R, there exists $\varepsilon > 0$ such that $H_R \subseteq V_\varepsilon$.*

Proof: To prove the theorem, we should find $\varepsilon > 0$ for which if a point $p \notin V_\varepsilon$, then $p \notin H_R$. Denote the distance from p to C by r, and assume that the sample is sufficiently dense with

$$\rho < \frac{T}{2}\sqrt{\frac{r}{R}(1-\frac{r}{4R})}.$$

Assume $p \notin V_\varepsilon$. As illustrated in Figure 18, consider the two rays from C whose angle from $\overline{pC}$ is $\pm\alpha$ with $\sin\alpha = 2\rho/T$.

We first show that the region between the two rays and the $(\varepsilon-\rho)$-circle around p contains two sample points, q_1' and q_2', on either sides of the line $\overline{pC}$, and then prove that $\widehat{p}$ must lie inside the triangle $\triangle(C,\widehat{q_1},\widehat{q_2})$.

To show this, we consider two other rays from C whose angle from $\overline{pC}$ is $\pm\alpha/2$. Since $p \notin V_\varepsilon$ (i.e., p lies in a completely hidden circle), S must intersect these two rays at some points between C and the ε-circle around p. This in turn implies that we can find two sample points q_1' and q_2' within distance ρ from the two intersection points.

We next show that $\widehat{p}$ must lie inside the triangle $\triangle(C,\widehat{q_1'},\widehat{q_2'})$. Denote by q_1 and q_2 the intersection points of the two rays from C whose angle from $\overline{pC}$ is α with the $(\varepsilon-\rho)$-circle around p. It can be readily shown that if $\widehat{p}$ lies inside $\triangle(C,\widehat{q_1},\widehat{q_2})$, then it must also lie inside $\triangle(C,\widehat{q_1'}\widehat{q_2'})$ (by noticing that since the circular arc from q_1 to q_2 is concave, its flipped image must be convex).

Let K be the distance from C to q_i $(i=1,2)$. The distance from C to $\widehat{q_i}$ is thus $2R-K$, and $\widehat{p}$ lies inside $\triangle(C,\widehat{q_1},\widehat{q_2})$ if

$$(2R-K)\cos\alpha > 2R-r.$$

This implies that

$$1 = \sin^2\alpha + \cos^2\alpha > \frac{4\rho^2}{T^2} + \frac{(2R-r)^2}{(2R-K)^2},$$

from which we obtain that

$$0 < K < 2R - \frac{2R-r}{\sqrt{1-\frac{4\rho^2}{T^2}}}.$$

This relation can be used to determine ε. We compose r from two segments whose length is determined by the Pythagorean relation:

$$\sqrt{K^2(1-\frac{4\rho^2}{T^2})} + \sqrt{(\varepsilon-\rho)^2 - \frac{4\rho^2K^2}{T^2}} < r.$$

Consequently,

$$(\varepsilon-\rho)^2 > \left(r - \sqrt{K^2(1-\frac{4\rho^2}{T^2})}\right)^2 + \frac{4\rho^2K^2}{T^2}$$

and thus

$$\varepsilon > \rho + \sqrt{\left(r - \sqrt{K^2(1-\frac{4\rho^2}{T^2})}\right)^2 + \frac{4\rho^2K^2}{T^2}} \quad .\square$$

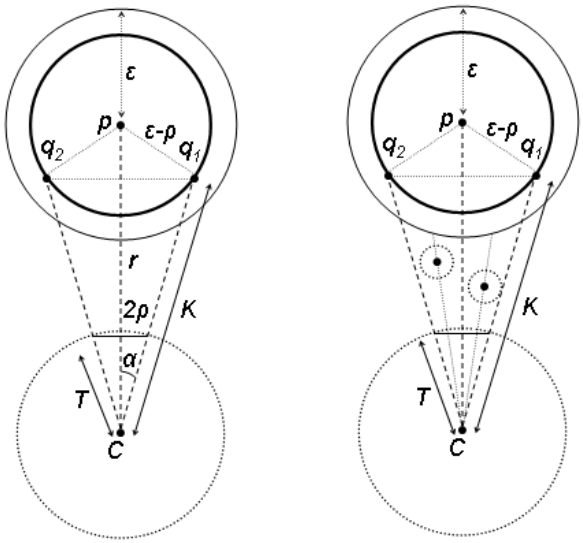


Figure 18: Geometric setup for Theorems 4.4 and 4.5. The existence of sample points q_1' and q_2' inside the triangle $\triangle(C,q_1,q_2)$ guarantees that p will belong to H_R.

Theorem 4.5 *Assume that the sample is sufficiently dense, then for sufficiently large $\varepsilon > 0$, there exists $R > 0$ such that $H_R \subseteq V_\varepsilon$.*

Proof: The proof is similar to the previous theorem. Denote by α the angle, $sin\alpha = 2\rho/T$. Consider the rays from C that form an angle $\pm\alpha$ with the line $\overline{pC}$. Denote by K the distance from C to the intersections of these rays with the $(\varepsilon-\rho)$-circle around p (q_1 and q_2). (Note that ε must be sufficiently large, or the sample be sufficiently dense, for these intersections to exist.) It is now possible to show that if $p \notin V_\varepsilon$ and we select R that satisfies

$$(2R-K)\cos\alpha > 2R-r,$$

then $p \notin H_R$. Therefore,

$$R < \frac{r - K\cos\alpha}{2(1-\cos\alpha)},$$

with $\cos\alpha = \sqrt{1-4\rho^2/T^2}$. Note also that we require ε to be sufficiently large so that $r > K(1-\cos\alpha)$. However, as we increase the density of the sample we can use smaller values of ε. □