Fall 2019

Final Project Report: Visibility of Point Cloud

Prakash Dhimal *CS 633 Computational Geometry*

Abstract

Given a point cloud $P = P_1, P_2, P_3...P_n$ and a viewpoint C, is it possible to determine all the points visible from C directly from the point cloud? Katz et al. [1] have shown that this is indeed the case and that this task can be performed without surface reconstruction and normal estimation. This paper investigates and implements an operator called the *Hidden Points Removal* operator described in [1], which computes only the visible points from a given viewpoint. The operator is simple and it consists of two steps:

1. Spherical inversion
2. Convex hull construction

During Spherical inversion we transform the points to a new domain. Once the spherical inversion is performed, the convex hull of the transformed set of points union C, the viewpoint, is computed. Points that lie on the convex hull are visible from the given viewpoint.

1 Motivation and Introduction

We formally define a point cloud P as follows:

$$P = P_1, P_2, P_3...P_n \tag{1}$$

A point cloud is simply a set of data points in space. It is a collection of points that represent a 3D shape or feature, and might contain additional information about each point P_i such as color. Point clouds are most often created by remote sensing and 3D scanning devices such as Light Detection and Ranging (LiDAR). The use of LiDAR and similar 3D scanning devices has been increasing in the past decade. Figure 1 shows a simple point cloud with Red Green Blue values associated with each point P_i.

Determining the visibility of point cloud is an interesting problem in itself. It is also useful in visualizing point clouds, in view-dependent reconstruction and in shadow casting [1]. Determining visibility can lead to a more robust registration of the points from multiple point clouds. This can also provide information about spatial relations and potential obstacles in the line of sight between two points in space, such analyses have been done in estimating the visibility of a landmark, and in finding the optimal location to place a surveillance camera.

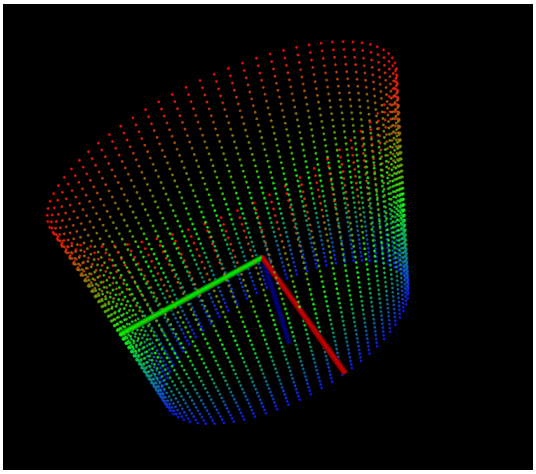

Figure 1: Sample Point Cloud

Suppose that we are given a point cloud depicting an object such as the bunny in figure 2. If all of the points are visualized, it is difficult to determine whether the bunny is facing forward or backwards. After calculating all of the points visible from a given camera location it can be seen in figure 3 that the bunny is facing forward.

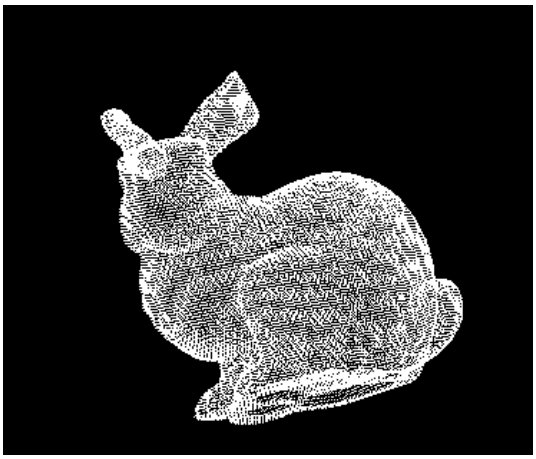

Figure 2: Is the Bunny facing forward or backward?

One way to compute visibility of a point cloud is to reconstruct the surface and determine visibility on the reconstructed surface. Surface reconstruction is difficult and often requires additional information such as normals and only tends on work with dense point cloud. In [1] it was shown that we can determine visibility without reconstructing a surface or estimating normals. This means that we can compute visibility directly from the point cloud.

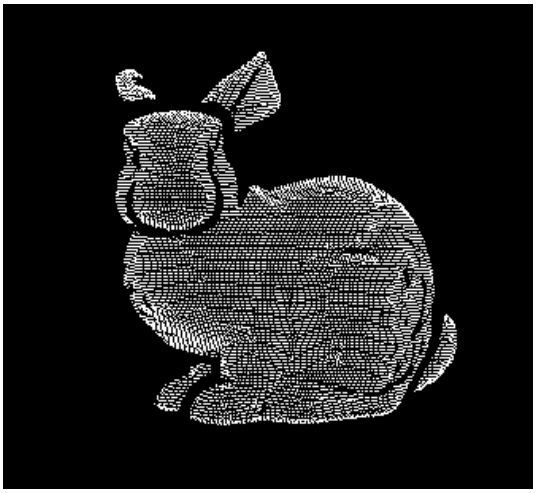

Figure 3: The Bunny is facing forward

2 Methods Used and Studied

Given a Point cloud $P = P_1, P_2, P_3...P_n$, and a viewpoint C our goal is to determine all of the points that are visible from C. To put it another way, for every point P_i in P we would like to determine if P_i is visible from C. We would like to mark a point P_i visible if and only if the point P_i is visible from C.

2.1 Hidden Point Removal Operator

Katz et al. [1] introduced an operator called the Hidden Point Removal operator that can be used to determine all of the points that are visible from C. The operator consists of two steps:

1. Spherical inversion
2. Convex hull construction

During spherical inversion we associate the points with a co-ordinate system where the viewpoint C is at the origin. We then map every point P_i along the ray from C to P_i such that this mapping is monotonically decreasing in the norm ($||P_i||$). Once we perform the spherical inversion, we compute a convex hull of the transformed set of points *union* C, the viewpoint. It is important to note that simply calculating the line of sight from the viewpoint C to the point P_i will always mark the point as visible, except in some degenerate cases.

What this operator is doing is a simple reduction of the problem of visibility to the problem of convex hull construction. This is because all we are doing is a simple transformation of the points and then computing the convex hull. The complexity of the operator is $O(nlogn)$ in two and three dimensions and can be extended into higher dimensions. The complexity of the operator is asymptotically similar to that of a convex hull construction, that is $O(nlogn)$ in two and three dimensions.

2.2 Spherical inversion

Consider a D-dimensional sphere with radius R, centered at the origin C, and includes all points in P. Spherical flipping reflects a point P_iEP with respect to the sphere by applying the following equation:

$$f(P_i) = P_i + 2(R - ||P_i||)\frac{P_i}{||P_i|}| = \widehat{P}_i \tag{2}$$

Spherical inversion is taking a point internal to some bounding sphere and projecting every point internal to the sphere to its image which is external to the using equation 2. Spherical inversion reflects every point P_i internal to the sphere along the ray from C to P_i to its image outside the sphere, as shown in figure 4.

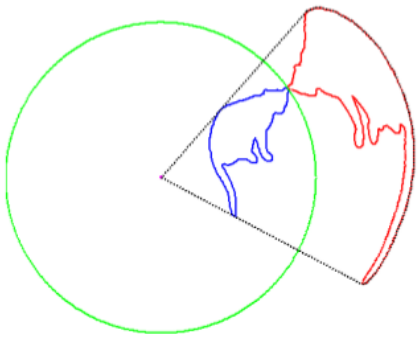

Figure 4: Spherical flipping, shown in red, of a 2D curve, shown in blue, using a sphere, shown in green

Spherical inversion is taking the blue curve, which is constrained to be bounded in some green circle, which is centered at the origin, or the viewpoint C in our case, and it maps the curve to its image that you see in red, which is the result of the spherical inversion.

2.3 Convex Hull Construction

From the spherical flipping above, we have a new point cloud that can be denoted as:

$$\widehat{P} = \hat{P}_1, \hat{P}_2, \hat{P}_3 ... \hat{P}_n \tag{3}$$

The second, and the final step in our Hidden Point Removal operator is to construct the convex hull of the new point cloud $\widehat{P}$ union C, i.e:

$$C_{hull} = \widehat{P} \cup C \tag{4}$$

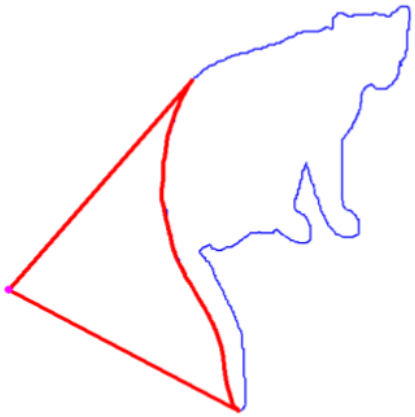

Figure 5: Projection of the convex hull

The points that lie on the convex hull C_{hull} in the equation above are the points that are visible form C. Figure 5 shows projection of the convex hull of the sphere in 4.

2.4 The data

The initial plan was to use data obtained from the *Velodyne HDL-64E LIDAR* scanner by the Motion and Shape Computing (MASC) Group at George Mason University to investigate the visibility of point clouds. While the Point Cloud Library (PCL) [2] provides capabilities to read and replay/visualize such data, it became clear that determining visibility of a moving point cloud is beyond the scope of this project. We would need more complex data structures such as kinetic data structures to handle the moving point cloud.

The data used in the scope of this project are all static models, such as the bunny from Stanford 3D Scanning Repository. The statue of Michelangelo's David was obtained form the PCL's data repository.

2.5 The implementation

Katz et al. [1] made the matlab code for this operator publicly available.

Listing 1: Matlab code

```
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)=[];
```

The goal was to implement this operator in C++ so that it can be used with the Point Cloud Library (PCL) for visualization and various other features that PCL provides. As such, the Hidden Point Operator (HPR) was implemented using PCL and various other libraries, such as Qt5, and VTK, in C++. The program takes as an input a point cloud in *PCD* or *PLY* format. Figure 6 shows the program with a split window showing the original point cloud and the visibility cloud.

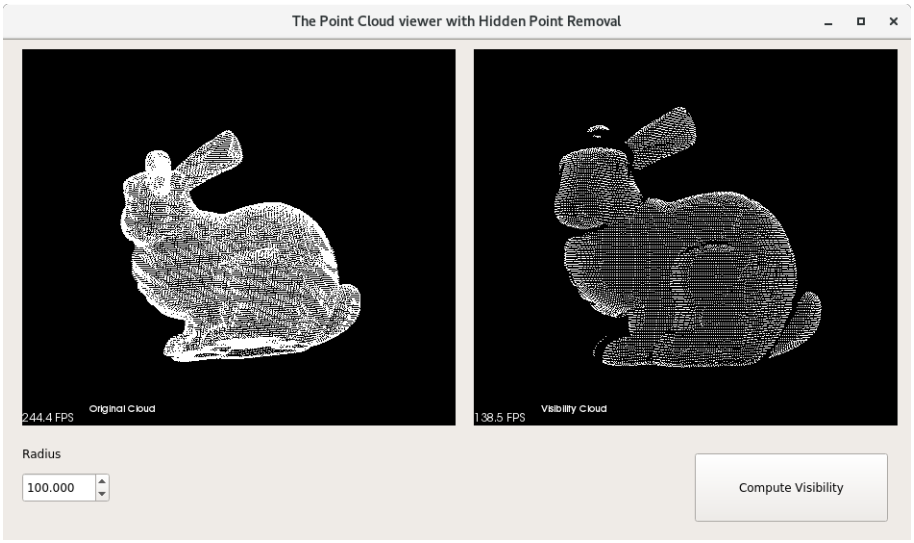

Figure 6: The Point Cloud viewer with Hidden Point Removal

This project will be made publicly available at: https://github.com/pdhimal1/HPR.git

3 Results/Findings

4 Conclusion

References

[1] Sagi Katz, Ayellet Tal, and Ronen Basri. Direct visibility of point sets. *ACM Trans. Graph.*, 26(3), July 2007. (document), 1, 1, 2.1, 2.5

[2] Radu Bogdan Rusu and Steve Cousins. 3D is here: Point Cloud Library (PCL). In *IEEE International Conference on Robotics and Automation (ICRA)*, Shanghai, China, May 9-13 2011. 2.4

List of Figures

Listings