In *Proceedings IASTED Computer Graphics and Imaging*, 2003.

Efficient Level-of-Details for Point Based Rendering

Renato Pajarola[1]

Computer Graphics Lab
Information & Computer Science
University of California Irvine

Abstract

In this paper we present techniques for the efficient generation of a level-of-detail (LOD) data structure for large scale point-based surface representation and rendering. Our approach generates a spatial partitioning hierarchy of irregular point samples in 3D space, and we provide an efficient point-octree LOD generation algorithm. Using the concept of transformation-invariant homogeneous covariance matrices we show how bounding ellipsoid information can efficiently be computed for all LODs. Furthermore, we present an efficient data structure for the representation of the LOD hierarchy.

Keywords: point-based rendering, multiresolution modeling, level-of-detail

1. Introduction

Points as rendering primitives have been discussed first in [12] and have only recently been rediscovered as viable graphics rendering primitives [10]. In particular they offer efficient level-of-detail (LOD) representations as demonstrated in [20], [2], [19] and [16] for rendering large point-sampled surfaces such as the digitally scanned statues of Michelangelo [11]. One of the main advantages of point based surface representations over polygonal meshes is that no mesh connectivity information must be generated via surface extraction methods nor maintained during LOD-based rendering. Recent efforts in point-based rendering (PBR) have not only resulted in efficient LOD representations but also in high-quality rendering algorithms such as the EWA splatting method [18,24,19] or the object-space point blending in [16].

Further research using point primitives has considered issues such as the combination of point and triangle primitives in LOD-based rendering approaches [5,3,8,4] or simplification of large point sets [1,17].

The challenge of generating an efficient LOD representations lies in the efficient processing of large point sample data sets. In this paper we address the issue of how to generate a spatial partitioning LOD hierarchy and the necessary attributes associated with each LOD node in that hierarchy. In particular, the efficient computation of bounding volume information as well as the splat-size determination are important in multiresolution PBR merhods. Figure 1 shows an example rendering result from our multiresolution data structure of David's head model, at 16 pixels screen-space error threshold only one forth of the points are rendered while providing an extremely high display quality.

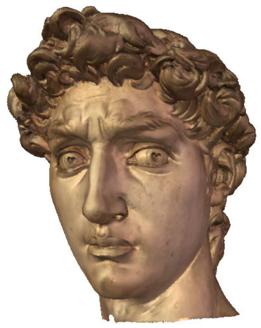

FIGURE 1. The head of Michelangelo's David statue rendered with τ =16 pixels screen tolerance, at 1/4 of the full resolution (510827 out of 2000606 points).

2. Surface Representation

We consider the LOD generation of objects represented as dense sets of irregularly distributed point-samples. We assume that the discrete input point set sufficiently samples the object's geometry and color texture (i.e. satisfies the Nyquist sampling criteria). The input data set consists of surface samples s with coordinates $\boldsymbol{p}$, normal orientation $\boldsymbol{n}$ and surface color $\boldsymbol{c}$. Each sample s also has information about its spatial extent in object-space which is given by the parameters of an elliptical disk e centered at $\boldsymbol{p}$ and orthogonal to $\boldsymbol{n}$. An elliptical disk e consists of major and minor axis directions $\boldsymbol{e}_1$ and $\boldsymbol{e}_2$ and their lengths. Together with the surface normal $\boldsymbol{n}$, the axis directions $\boldsymbol{e}_1$ and $\boldsymbol{e}_2$ define the local tangential coordinate system of that sample. The dense set of surface samples must cover the represented object without holes and thus overlap each other in object-space as shown in Figure 2

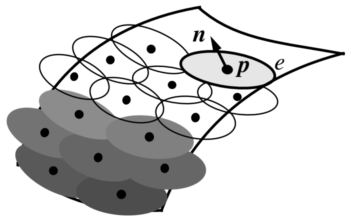

FIGURE 2. Elliptical surface elements covering a smooth and curved 3D surface.

In the remainder of this paper we assume that the input surface samples already contain information about their elliptical splat size and we focus on the LOD hierarchy generation. Initial splat ellipses can be derived using various approaches (see for example [7], [8] or [17]) and can

1. 444 Computer Science, Irvine CA 92697, pajarola@acm.org

also be obtained using our presented covariance-based method over k-nearest neighbors.

3. Space-partitioning hierarchy

The multiresolution point representations considered in this paper are hierarchical space-partitioning data structures [13,22]. Each cell or node c of such a hierarchy H, containing a set of k point surface samples $S_c = \{s_1 \ldots s_k\}$ has a representative sample s_c with average coordinates $\bar{\boldsymbol{p}}_c = k^{-1} \cdot \sum_{i=1}^{k} \boldsymbol{p}_i$, as well as average normal $\bar{\boldsymbol{n}}_c = k^{-1} \sum_{i=1}^{k} \boldsymbol{n}_i$ and color $\bar{\boldsymbol{c}}_c = k^{-1} \sum_{i=1}^{k} \boldsymbol{c}_i$ information. Furthermore, for efficient view-frustum and back-face culling each cell $c \in H$ also includes the sphere radius r_c and normal-cone [23] semi-angle θ_c parameters bounding all points in S_c. Several conforming space-partitioning multiresolution hierarchies have been proposed for point-based rendering [20,2,17]. In our work we want to focus on a *point-octree* [21,22] hierarchy which partitions the space adaptively to the sample distribution (data-driven) rather than regularly in space (space-driven) like *region-octrees* [21,22] which have been used more commonly.

Point-Octree Generation: Given N input point samples $s_1 \ldots s_N$, a point-octree data structure can efficiently be generated by a single depth-first traversal in $O(N \log N)$ time. Starting with the root, at each node c of the current hierarchy H its corresponding input points have to be distributed among the eight child nodes. In point-octrees, the one-to-eight subdivision is determined by the average of the points in each node. Therefore, given the k point samples $S_c = \{s_1 \ldots s_k\}$ of a node $c \in H$ and the average position $\bar{\boldsymbol{p}}_c$ of the samples S_c, the set S_c is partitioned into sets S_1 to S_8 according to the eight octants with respect to the split coordinate $\bar{\boldsymbol{p}}_c$.

As shown in Figure 3, during the recursive top-down construction of the octree we assume that the input data to each node not only consists of the set of points S_c but also includes its average coordinate $\bar{\boldsymbol{p}}_c$, normal $\bar{\boldsymbol{n}}_c$, and color $\bar{\boldsymbol{c}}_c$. Thus these average values are computed in the parent node. In one linear traversal of the samples $s_i \in S_c$ the current node c does the following:

1. divide S_c into the subsets S_i with $s \in S_i$ in octant i with respect to split coordinate $\bar{\boldsymbol{p}}_c$,
2. accumulate averages $\bar{\boldsymbol{p}}_i$, $\bar{\boldsymbol{n}}_i$, and $\bar{\boldsymbol{c}}_i$,
3. compute bounding sphere radius r_c of points $\boldsymbol{p}_i$ with respect to center $\bar{\boldsymbol{p}}_c$, and
4. compute bounding normal cone semi angle θ_c of normals $\boldsymbol{n}_i$ with respect to center normal $\bar{\boldsymbol{n}}_c$.

This process is repeated recursively passing the information $(S_i, \bar{\boldsymbol{p}}_i, \bar{\boldsymbol{n}}_i, \bar{\boldsymbol{c}}_i)$ to the eight child nodes c_i.

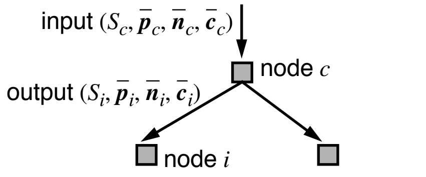
input $(S_c, \bar{\boldsymbol{p}}_c, \bar{\boldsymbol{n}}_c, \bar{\boldsymbol{c}}_c)$
node c
output $(S_i, \bar{\boldsymbol{p}}_i, \bar{\boldsymbol{n}}_i, \bar{\boldsymbol{c}}_i)$
node i

FIGURE 3. Top-down point-octree data flow.

As outlined in the previous section, each LOD node c in the point-octree hierarchy needs to store its spatial extent in form of an elliptical disk e_c. To compute these elliptical disks efficiently, we use a novel technique based on the *generic homogeneous covariance matrix* concept outlined in the following section. This allows us to propagate covariance information in form of a homogeneous 4x4 matrix efficiently bottom-up in the LOD hierarchy as illustrated in Figure 4. To compute the elliptical disk e_c, the child nodes of c return their generic homogeneous covariance matrices $\overline{M}_i$ to calculate $\overline{M}_c$. The following section explains in more detail the concept of a *generic homogeneous covariance matrix*, how it can efficiently be applied during the construction of a multiresolution hierarchy, and how to compute a bounding elliptical disk of a set of points.

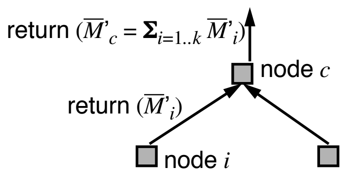
return $(\overline{M}'_c = \boldsymbol{\Sigma}_{i=1..k} \overline{M}'_i)$
node c
return $(\overline{M}'_i)$
node i

FIGURE 4. Bottom-up point-octree data flow.

4. Generic homogeneous covariance

Since the smallest bounding ellipsoid of a set of points is hard to compute, covariance analysis – basically a singular value decomposition approach – is often used to determine an elliptical Gaussian distribution fitting the given point set [9].

The covariance matrix of a set of n points $\boldsymbol{p}_1, \ldots, \boldsymbol{p}_n \in \mathbf{R}^3$ and their average $\bar{\boldsymbol{p}} = n^{-1} \sum_{i=1}^{n} \boldsymbol{p}_i$ is defined by

$$M = n^{-1} \sum_{i=1}^{n} (\boldsymbol{p}_i - \bar{\boldsymbol{p}}) \cdot (\boldsymbol{p}_i - \bar{\boldsymbol{p}})^{\mathrm{T}} . \qquad \textbf{(EQ 1)}$$

The mean $\bar{\boldsymbol{p}}$ denotes the center of the ellipsoid, and the unit-length eigenvectors $\boldsymbol{v}_1$, $\boldsymbol{v}_2$ and $\boldsymbol{v}_3$ of the covariance matrix M denote the axis of the ellipsoid. The ratios of the axis lengths are given by the eigenvalues λ_1, λ_2 and λ_3. In the local coordinate system with center $\bar{\boldsymbol{p}}$, and x-, y- and z-axis aligned with eigenvectors $\boldsymbol{v}_1$, $\boldsymbol{v}_2$ and $\boldsymbol{v}_3$, the ellipsoid distribution is then given by $x^2/a^2 + y^2/b^2 + z^2/c^2 = f^2$ with $a = \lambda_1$, $b = \lambda_2$ and $c = \lambda_3$. To scale the ellipsoid such that it encloses exactly all points, we have to find $f_{\max}$ over all $\boldsymbol{p}_i$ and scale the ellipsoid parameters to $a' = f_{\max} a$, $b' = f_{\max} b$ and $c' = f_{\max} c$. The so defined ellipsoid is a close approximation of the smallest bounding ellipsoid of the given point set, feasible to compute and widely used in practice.

The main problem with this formulation is the dependency of Equation 1 on the mean $\bar{\boldsymbol{p}}$. Therefore, if the covariance matrices M_O and M_Q of two point sets $O = \{\boldsymbol{o}_1, \ldots, \boldsymbol{o}_n\}$ and $Q = \{\boldsymbol{q}_1, \ldots, \boldsymbol{q}_m\}$ as well as the covariance $M_{O \cup Q}$ of their union has to be calculated, Equation 1 has to be computed for point sets O and Q, as well as for their union $O \cup Q$. Therefore, the outer product $(\boldsymbol{p}_i - \bar{\boldsymbol{p}}) \cdot (\boldsymbol{p}_i - \bar{\boldsymbol{p}})^{\mathrm{T}}$ in Equation 1 is evaluated $|O| + |Q| + |O \cup Q| = 2 \cdot (n + m)$ times. In a multiresolution hierarchy over N points this leads to a cost of $O(N \log N)$ outer product calculations. Despite the fact that the cost of generating a multiresolution hierarchy is in the order of $O(N \log N)$, see also Section 3, it is desired to avoid excess numerical calculations such as the expensive outer product of two vectors.

Homogeneous Covariance: In a homogeneous coordinate system with points $\boldsymbol{p}'^{\mathrm{T}}_i = (\boldsymbol{p}^{\mathrm{T}}_i, 1)$ we can rewrite the outer product $(\boldsymbol{p}_i - \bar{\boldsymbol{p}}) \cdot (\boldsymbol{p}_i - \bar{\boldsymbol{p}})^{\mathrm{T}}$ to $(T \cdot \boldsymbol{p}'_i) \cdot (T \cdot \boldsymbol{p}'_i)^{\mathrm{T}}$ with the transformation matrix T denoting the translation by the mean $-\bar{\boldsymbol{p}}$. Thus we can revise Equation 1 to

$$\begin{aligned} M_{\mathrm{h}} &= n^{-1}\textstyle\sum_{i=1}^{n}(T \cdot \boldsymbol{p}'_i) \cdot (T \cdot \boldsymbol{p}'_i)^{\mathrm{T}} \\ &= n^{-1}\textstyle\sum_{i=1}^{n}(T \cdot \boldsymbol{p}'_i \cdot \boldsymbol{p}'^{\mathrm{T}}_i \cdot T^{\mathrm{T}}) \\ &= n^{-1}T \cdot (\textstyle\sum_{i=1}^{n}\boldsymbol{p}'_i \cdot \boldsymbol{p}'^{\mathrm{T}}_i) \cdot T^{\mathrm{T}} = n^{-1}T \cdot \overline{M}_{\mathrm{h}} \cdot T^{\mathrm{T}} \end{aligned} \qquad \textbf{(EQ 2)}$$

with $\overline{M}_{\mathrm{h}} = \sum_{i=1}^{n}\boldsymbol{p}'_i \cdot \boldsymbol{p}'^{\mathrm{T}}_i$ denoting the new *generic homogeneous covariance matrix* of points $\boldsymbol{p}_1 \ldots \boldsymbol{p}_n$. This matrix $\overline{M}_{\mathrm{h}}$ expresses the non-normalized[1] covariance of a point set with respect to the origin of the coordinate system. Therefore, we can now express the homogeneous covariance M_{h} of a set of points with respect to any arbitrary center $\boldsymbol{o}$ given by a translation T with parameters $(-o_x, -o_y, -o_z)$ by $M_{\mathrm{h}} = n^{-1}T \cdot \overline{M}_{\mathrm{h}} \cdot T^{\mathrm{T}}$. The corresponding covariance matrix M in Cartesian space is given by the upper-left 3x3 sub-matrix of M_{h}.

In fact, we can now transform the covariance into any coordinate system, thus not only including a translation T to a new center of origin but also involving a rotation R of the coordinate axis. Therefore, the homogeneous covariance matrix in any local coordinate system is given by

$$M_{\mathrm{h}} = n^{-1}R \cdot T \cdot \overline{M}_{\mathrm{h}} \cdot T^{\mathrm{T}} \cdot R^{\mathrm{T}}. \qquad \textbf{(EQ 3)}$$

From the 4D homogeneous covariance matrix M_{h} the 3D Cartesian-space covariance matrix M is given by dropping the fourth row and column. This corresponds to an orthogonal projection from 4D into 3D along the homogeneous axis. Similarly we can express the covariance in any lower-dimensional sub-space by orthogonal projection. Therefore, we get the covariance $M^{x,y}$ of the points projected into the x,y-plane by dropping the homogeneous and z-axis rows and columns from M_{h}.

Since the introduced generic homogeneous covariance matrix is invariant to transformations, the union $O \cup Q$ of two point sets O and Q can efficiently be handled: Given their generic homogeneous covariances $\overline{M}_O = \sum \boldsymbol{o}'_i \cdot \boldsymbol{o}'^{\mathrm{T}}_i$ and $\overline{M}_Q = \sum \boldsymbol{q}'_i \cdot \boldsymbol{q}'^{\mathrm{T}}_i$, the combined generic covariance matrix $\overline{M}_{O \cup Q}$ of the union $O \cup Q$ is then given by

$$\overline{M}_{O \cup Q} = \overline{M}_O + \overline{M}_Q. \qquad \textbf{(EQ 4)}$$

Therefore, in a multiresolution hierarchy the expensive outer product sums of points only have to be calculated once on the leaf-level of the hierarchy, thus only O(N) outer products are computed. All non-leaf nodes in the hierarchy can compute their generic homogeneous covariance matrix from their child nodes by simple addition as in Equation 4.

Splat-Size Determination: To determine the splat size of an elliptical disks e_c of a node $c \in H$ in the LOD hierarchy H we must compute a tangential bounding ellipse of the points associated with c. The basic principle is to project the set of points $\boldsymbol{p}_1 \ldots \boldsymbol{p}_k$ of a node c into the tangent plane κ_c given by the equation $\bar{\boldsymbol{n}}_c \bullet (\boldsymbol{x} - \bar{\boldsymbol{p}}_c) = 0$ which is defined by the node location $\bar{\boldsymbol{p}}_c$ and normal orientation $\bar{\boldsymbol{n}}_c$. The bounding ellipse e_c is then computed in the tangent plane κ_c as illustrated in Figure 5.

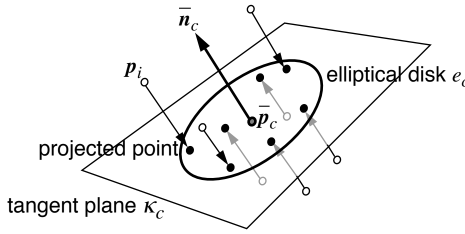

FIGURE 5. Projection of points onto tangent plane κ_c at position $\bar{\boldsymbol{p}}_c$ and with normal $\bar{\boldsymbol{n}}_c$.

Using the generic homogeneous covariance matrix $\overline{M}_c$ of node c we first express the covariance in the 2D sub-space of the tangent plane κ_c, then compute the elliptical distribution of points in κ_c and lastly we adjust the ellipse to bound all projected points in κ_c.

Let us first outline how we get the ellipse axis and its axis-ratio within the tangent plane κ_c. For a node c and its matrix $\overline{M}_c$ denoting the covariance of all points $\boldsymbol{p}_i$ represented by c, we apply a coordinate system transformation RT according to Equation 3 to a local tangential coordinate system of c. Thus the translation matrix T has the last column being $(-\bar{p}_x, -\bar{p}_y, -\bar{p}_z, 1)^{\mathrm{T}}$ from the node's location $\bar{\boldsymbol{p}}_c$ and the rotation matrix R has the row vectors $R_x = R_y \times R_z$, $R_y = (0, -\bar{n}_z, \bar{n}_y, 0)$ and $R_z = (\bar{n}_x, \bar{n}_y, \bar{n}_z, 0)$ given by $\bar{\boldsymbol{n}}_c$. This transforms the covariance into M_c given in the local tangent-space coordinate system. Moreover, the covariance of the points projected into the 2D tangent sub-space κ_c is given by the upper-left 2x2 sub-matrix of M_c, denoted by M_{κ_c}. The axis-ratio of the elliptical distribution in κ_c is then given by the eigenvalue decomposition of M_{κ_c}, and we obtain the eigenvalues λ_1 and λ_2 from solving the quadratic equation

$$\lambda^2 + \mathrm{trace}(M_{\kappa_c}) \cdot \lambda + \det(M_{\kappa_c}) = 0. \qquad \textbf{(EQ 5)}$$

Furthermore, we obtain the major and minor axis directions of the bounding ellipse in the tangent plane κ_c by solving

$$M_{\kappa_c} \cdot \boldsymbol{v}_i = \lambda_i \cdot \boldsymbol{v}_i \qquad \textbf{(EQ 6)}$$

for the eigenvectors $\boldsymbol{v}_1$ and $\boldsymbol{v}_2$. Note that $\boldsymbol{v}_1$ and $\boldsymbol{v}_2$ are in $\mathbf{R}^2$, given in the tangent plane κ_c. However, with respect to the local tangential coordinate system with z-axis perpendicular to κ_c we get their 3D vectors by $\boldsymbol{v}'^{\mathrm{T}}_i = (\boldsymbol{v}^{\mathrm{T}}_i, 0)$. The world-coordinate system ellipse axis $\boldsymbol{e}_1$ and $\boldsymbol{e}_2$ are then obtained by applying the inverse rotation R^{-1} of the coordinate system transform and normalization to unit length $\boldsymbol{e}_i = R^{-1} \cdot \boldsymbol{v}'_i / |\boldsymbol{v}'_i|$. Now we have defined a planar elliptical disk e_c in the world coordinate system with center $\bar{\boldsymbol{p}}_c$, axis directions $\boldsymbol{e}_1$, $\boldsymbol{e}_2$ perpendicular to $\bar{\boldsymbol{n}}_c$ as well as major axis length $a' = \lambda_1$ and minor axis length $b' = \lambda_2$.[2]

The so defined elliptical disk does not yet exactly bound all points $\boldsymbol{p}_i$ projected onto plane κ_c and its size must be scaled to $a = fa'$ and $b = fb'$. We obtain the necessary maximal scale factor f by evaluating the ellipse equation $f^2 = x^2/a^2 + y^2/b^2$ in the tangent plane κ_c spanned

1. no division by n

2. assuming λ_1 bigger than λ_2

by $\boldsymbol{e}_1$ and $\boldsymbol{e}_2$ for all points $\boldsymbol{p}_i$, with $x_i = (\boldsymbol{p}_i - \bar{\boldsymbol{p}}_c) \bullet \boldsymbol{e}_1$ and $y_i = (\boldsymbol{p}_i - \bar{\boldsymbol{p}}_c) \bullet \boldsymbol{e}_2$. However, since every surface element s_i represents an elliptical disk e_i and not just a single point $\boldsymbol{p}_i$ we generate bounding ellipses that not only include $\boldsymbol{p}_i$ but cover the entire disks e_i, approximated by oriented bounding boxes as illustrated in Figure 6. The bounding boxes are derived by the major and minor axis of e_i. Without this conservative measure a coarse LOD would not cover the surface without holes. For non-leaf nodes the bounding ellipse covers its child node ellipses.

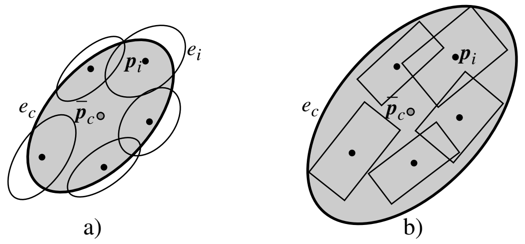

FIGURE 6. a) Ellipse e_c bounding only the points $\boldsymbol{p}_i$ and b) conservatively bounding the disks e_i.

The outlined generation of elliptical disks cannot only be used to compute bounding elliptical disks of nodes c of the multiresolution hierarchy H, but could also be applied in a similar way to obtain elliptical disks of the initial input point set. For this, one would compute the k-nearest neighbors of each point, calculate the average normal if necessary, compute the covariance matrix of this neighborhood and get a bounding ellipse of the k-nearest neighbors as outlined above.

5. Data structure

The data structure of our point-octree LOD hierarchy is fairly simple. As shown in Figure 7 each octree node has a counter denoting how many data points this node references (zero for non-leaf nodes), contains a point data structure to hold the split point information, and includes an array of pointers to child nodes (for non-leaf nodes) or sample data points (for leaf nodes).

```
struct Octree {
   unsigned char np;     // number of points
   MyPoint split;        // split point

   // pointers to child nodes (np==0)
   // or if leaf (np>0) to sample points
   union {
      Octree *(*child)[8];
      MyPoint *points;
   };
};
```

FIGURE 7. Octree node data structure.

The data structure for sample points is given in Figure 8 and basically includes the information outlined in Sections 2 and 3 to specify a sample splat point. Note that for storage efficiency the normal information is not encoded as a three-dimensional vector but is represented by an index into a pre-computed table of quantized normals as illustrated in Figure 9.

```
struct MyPoint {
   Vector3f p;     // coordinates
   Color3u c;      // color

   float size;     // bounding sphere size
   float theta;    // normal cone semi-angle

   float ell;      // major axis length
   float ratio;    // minor/major axis ratio

   NIndex nIndex; // normal
   NIndex e1Index;// major ellipse axis
   NIndex e2Index;// minor ellipse axis
};
```

FIGURE 8. Point sample data structure.

As shown in Figure 9, the normal-space is quantized to q bits and a discrete number 2^q of normal directions is maintained in a look-up table similar to [6]. Three bits are required to denote the octant of the normal and the remaining q-3 bits can be used to uniformly subdivide the unit sphere in the first octant. The first octant is subdivided as shown on the right in Figure 9, the index i starts at the z-axis pole and the latitude is subdivided into k values. The longitude subdivision j varies according to the latitude at latitude i, the longitude is subdivided into i segments. Thus the table stores $(k^2+k)/2$ quantized normals. Section 7 provides results on the efficiency of this data structure and examples of the normal table.

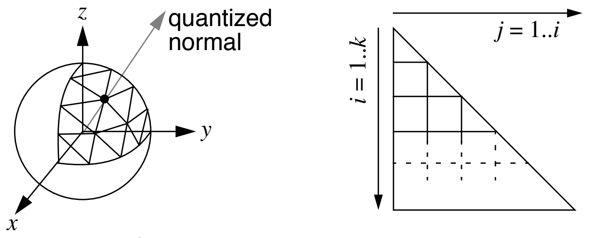

FIGURE 9. Quantized normal representation.

6. Rendering

In this paper we will briefly outline the LOD rendering aspects of our point blending and splatting algorithm proposed in [16]. The overall rendering process includes view-dependent LOD selection of points which is explained in this section, followed by a visibility splatting, blending and rendering step that is explained in [16].

View-Dependent LOD Selection: The LOD selection takes three view-dependent selection criteria into account: *view-frustum culling*, *back-face culling* and *screen projection tolerance*. These criteria allow efficient back-tracking during the recursive traversal of the point-octree hierarchy H. If for a node $c \in H$ the bounding sphere with radius r_c does not intersect the view frustum (approximated by a viewing cone) or if the bounding normal-cone [23] with semi-angle θ_c indicates a completely back-facing surface region, recursive LOD selection can be stopped. Only a few floating-point operations are used to compute the view-frustum and back-face culling criteria as proposed in [14, 15]. We assume that the viewpoint $\boldsymbol{e}$, the viewing direction $\boldsymbol{w}$ and semi-angle ω (as well as its sine, cosine and tangens values) of the viewing cone are given for each frame.

As shown in Figure 10, view-frustum culling is performed if $\gamma - \alpha > \omega$ and back-face culling is done if

$\beta + \theta < 90°$. Both criteria can be computed without any expensive trigonometric functions as shown in [14, 15].

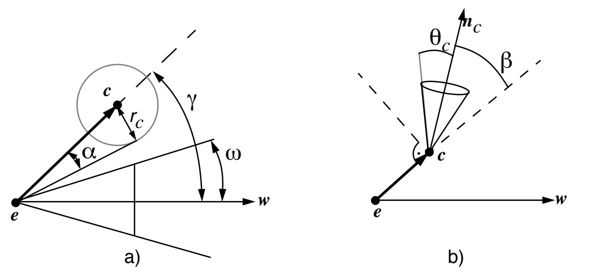

FIGURE 10. a) View-frustum culling if $\gamma - \alpha > \omega$ and b) back-face culling if $\beta + \theta < 90°$.

Additionally, a screen projection error tolerance is used for LOD selection. A node c is selected if its elliptical disk e_c projected on screen is less than a threshold τ. Given the area A_{e_c} of the ellipse e_c, the normalized viewing direction $\boldsymbol{w}$ and the focal length d of the viewing plane as shown in Figure 11, the projected area on screen that is compared to τ for LOD selection is $A_{screen} = f(A_{e_c} \cdot d^2 / |(\boldsymbol{c} - \boldsymbol{e}) \cdot \boldsymbol{w}|^2)$. The factor $f = \cos(\gamma - \theta_c)$ takes the *tilting* of the ellipse with respect to the normal-cone into account, see also Figure 11 and [14]. With respect to a given viewpoint e, the visible area is maximal if $\gamma = 0° \Rightarrow \cos\gamma = 1$, and minimal if $\gamma = 90° \Rightarrow \cos\gamma = 0$. However, due to the normal variation bounded by θ_c, the maximal visible area can already occur when $\cos(\gamma - \theta)$ is 1.

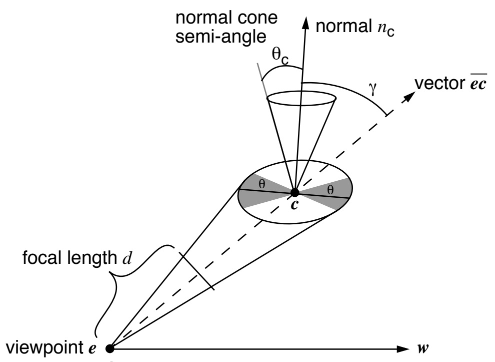

FIGURE 11. Screen projection of elliptical disk.

7. Experimental results

The data-driven space-partitioning point-octree hierarchy is illustrated in Figure 12 by transparent boxes. In comparison to more widely used region-octree hierarchies (i.e. in [20,2]), the point-octree is more adaptive to the spatial distribution of the points. This can also be seen from Table 1 which reports much fewer nodes for the octree hierarchy than reported in [20] for a region-octree (i.e. for the David head model).

In Table 1 we report LOD hierarchy construction times for several models performed on a 1.4GHz Pentium4 CPU. Reported are the number of point samples of each object, the number of nodes generated in the point-octree hierarchy and the CPU time cost that not only includes the spatial partitioning of the points but also the covariance matrix calculations and determination of elliptical splat sizes for all points. We can see that our multiresolution model and splat size generation achieves a performance of processing about 100,000 input points per second. Even multi-million point models can efficiently be processed by our approach, and the splat size generation using the homogeneous covariance computation is very efficient.

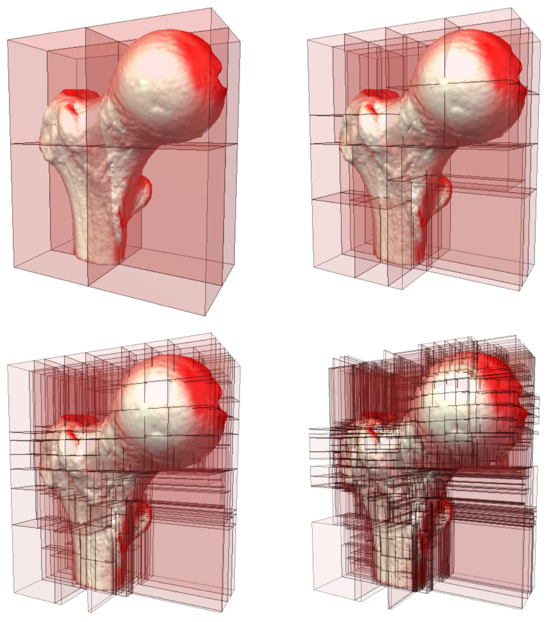

FIGURE 12. Point-octree space partitioning levels.

Model	#Points	#Nodes	Time
David	2000606	802371	20.6s
Female	302948	121439	2.97s
Balljoint	137062	54992	1.37s

TABLE 1. Multiresolution point hierarchy construction and splat generation times.

Table 2 shows the memory cost of our multiresolution point-octree data structure. The PLY file size shows a reference indexed triangle mesh file of the same data set. Our point-octree data structure requires about 46 bytes per input vertex, including all the LOD hierarchy information and per-point attributes (coordinates, normal, color, ellipse information). This compares favorably also with highly optimized multiresolution triangle mesh formats such as [14] which reported about 60 bytes/vertex on disk and 106 bytes/vertex in main memory to perform similar view-dependent LOD rendering.

Model	#Points	#Triangles	PLY File	#Points & Nodes	Octree File
David	2000606	2000606	165MB	2802977	89MB
Female	302948	302948	24MB	424387	13MB
Balljoint	137062	137062	11MB	192054	6MB

TABLE 2. Multiresolution point hierarchy file sizes.

The efficiency of the normal-space quantization is shown in Figure 13. Already a low quantization of subdividing the latitude into 31 discrete angles (and the longitude accordingly, see Section 5) that results in only $(31^2+31)/2 = 496$ normals that can be indexed by 9 bits only (thus $q = 9+3 = 12$ bits) shows a very dense distribution of normals and corresponding oriented triangles in Figure 13. Going to $q = 16$ bits leads to an extremely well sampled normal space in practice.

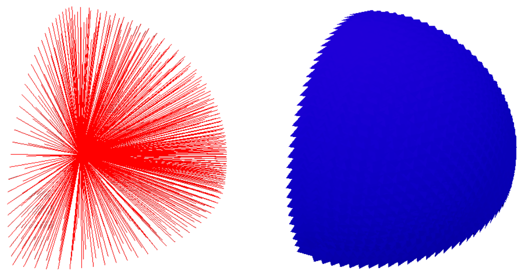

FIGURE 13. Quantization of normal-space into 9 bits.

Example renderings of some test models are given in Figure 14 to show the efficiency of our point-octree LOD hierarchy. Despite a very high screen-space error tolerance of 0.02%=246pixesl, the quality of the rendering is extremely good. The point-octree hierarchy allows very fast LOD-based point selection and rendering as reported in [16].

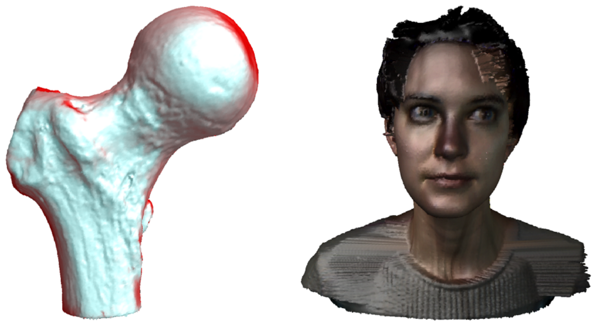

FIGURE 14. Balljoint (left, rendered 42076 out of 137062 points), Female (right, rendered 77227 out of 302948 points) at τ=0.02%. (screen-space tolerance τ given in percentage of viewport size)

8. Conclusions

We have presented efficient LOD generation techniques for point-based surface representations based on a data-driven point-octree LOD hierarchy and efficient covariance matrix computation. The experiments show the efficiency of our approach for multiresolution representation of large point data sets.

While not optimized on the bit-level for storage cost, our approach achieves excellent results compared to optimized multiresolution triangle mesh formats. Future work includes optimizing bit-level storage efficiency for octree nodes and point representations comparable to [20] and [2].

Acknowledgements

We would like to thank the Stanford *3D Scanning Repository* and *Digital Michelangelo* projects as well as *Cyberware* for freely providing geometric models to the research community.

References

[1] M. Alexa, J. Behr, D. Cohen-Or, S. Fleishman, D. Levin, and C. T. Silva. Point set surfaces. In *Proceedings IEEE Visualization 2001*, pages 21–28. Computer Society Press, 2001.

[2] M. Botsch, A. Wiratanaya, and L. Kobbelt. Efficient high quality rendering of point sampled geometry. In *Proceedings Eurographics Workshop on Rendering*, pages –, 2002.

[3] B. Chen and M. X. Nguyen. POP: A hybrid point and polygon rendering system for large data. In *Proceedings IEEE Visualization 2001*, pages 45–52, 2001.

[4] L. Coconu and H.-C. Hege. Hardware-oriented point-based rendering of complex scenes. In *Proceedings Eurographics Workshop on Rendering*, pages 43–52, 2002.

[5] J. D. Cohen, D. G. Aliaga, and W. Zhang. Hybrid simplification: Combining multi-resolution polygon and point rendering. In *Proceedings IEEE Visualization 2001*, pages 37–44, 2001.

[6] M. Deering. Geometry compression. In *Proceedings SIGGRAPH 95*, pages 13–20. ACM SIGGRAPH, 1995.

[7] T. K. Dey, J. Giesen, and J. Hudson. A delaunay based shape reconstruction from larga data. In *Proceedings IEEE Symposium in Parallel and Large Data Visualization and Graphics*, pages 19–27, 2001.

[8] T. K. Dey and J. Hudson. PMR: Point to mesh rendeering, a feature-based approach. In *Proceedings IEEE Visualization 2002*, pages 155–162. Computer Society Press, 2002.

[9] D. H. Eberly. *3D Game Engine Design*. Morgan Kaufmann Publishers, San Francisco, California, 2001.

[10] J. Grossman and W. J. Dally. Point sample rendering. In *Proceedings Eurographics Rendering Workshop 98*, pages 181–192. Eurographics, 1998.

[11] M. Levoy, K. Pulli, B. Curless, S. Rusinkiewicz, D. Koller, L. Pereira, M. Ginzton, S. Anderson, J. Davis, J. Ginsberg, J. Shade, and D. Fulk. The digital Michelangelo project: 3D scanning of large statues. In *Proceedings SIGGRAPH 2000*, pages 131–144. ACM SIGGRAPH, 2000.

[12] M. Levoy and T. Whitted. The use of points as display primitives. Technical Report TR 85-022, Department of Computer Science, University of North Carolina at Chapel Hill, 1985.

[13] J. Nievergelt. 7±2 criteria for assessing and comparing spatial data structures. In *Proceedings of the 1st Symposium on the Design and Implementation of Large Spatial Databases*, volume 409 of *Lecture Notes in Computer Science*, pages 3–27. Springer-Verlag, 1989.

[14] R. Pajarola. Fastmesh: Efficient view-dependent meshing. In *Proceedings Pacific Graphics 2001*, pages 22–30. IEEE, Computer Society Press, 2001.

[15] R. Pajarola, M. Antonijuan, and R. Lario. QuadTIN: Quadtree based triangulated irregular networks. In *Proceedings IEEE Visualization 2002*, pages 395–402. Computer Society Press, 2002.

[16] R. Pajarola, M. Sainz, and P. Guidotti. Object-space blending and splatting of points. Technical Report UCI-ICS-03-01, The School of Information and Computer Science, University of California Irvine, 2003. submitted for publication.

[17] M. Pauly, M. Gross, and L. P. Kobbelt. Efficient simplification of point-sampled surfaces. In *Proceedings IEEE Visualization 2002*, pages 163–170. Computer Society Press, 2002.

[18] H. Pfister, M. Zwicker, J. van Baar, and M. Gross. Surfels: Surface elements as rendering primitives. In *Proceedings SIGGRAPH 2000*, pages 335–342. ACM SIGGRAPH, 2000.

[19] L. Ren, H. Pfister, and M. Zwicker. Object space EWA surface splatting: A hardware accelerated approach to high quality point rendering. In *Proceedings EUROGRAPHICS 2002*, pages –, 2002. also in Computer Graphics Forum 21(3).

[20] S. Rusinkiewicz and M. Levoy. Qsplat: A multiresolution point rendering system for large meshes. In *Proceedings SIGGRAPH 2000*, pages 343–352. ACM SIGGRAPH, 2000.

[21] H. Samet. The quadtree and related hierarchical data structures. *Computing Surveys*, 16(2):187–260, June 1984.

[22] H. Samet. *The Design and Analysis of Spatial Data Structures*. Addison Wesley, Reading, Massachusetts, 1989.

[23] L. A. Shirman and S. S. Abi-Ezzi. The cone of normals technique for fast processing of curved patches. In *Proceedings EUROGRAPHICS 93*, pages 261–272, 1993. also in Computer Graphics Forum 12(3).

[24] M. Zwicker, H. Pfister, J. van Baar, and M. Gross. Surface splatting. In *Proceedings SIGGRAPH 2001*, pages 371–378. ACM SIGGRAPH, 2001.