

A quasi-Monte Carlo method for computing areas of point-sampled surfaces

Yu-Shen Liu, Jun-Hai Yong, Hui Zhang, Dong-Ming Yan, Jia-Guang Sun

▶ **To cite this version:**

Yu-Shen Liu, Jun-Hai Yong, Hui Zhang, Dong-Ming Yan, Jia-Guang Sun. A quasi-Monte Carlo method for computing areas of point-sampled surfaces. Computer-Aided Design, 2006. inria-00518328

HAL Id: inria-00518328

https://inria.hal.science/inria-00518328v1

Submitted on 17 Sep 2010

HAL is a multi-disciplinary open access archive for the deposit and dissemination of scientific research documents, whether they are published or not. The documents may come from teaching and research institutions in France or abroad, or from public or private research centers.

L'archive ouverte pluridisciplinaire **HAL**, est destinée au dépôt et à la diffusion de documents scientifiques de niveau recherche, publiés ou non, émanant des établissements d'enseignement et de recherche français ou étrangers, des laboratoires publics ou privés.



HAL Authorization

A quasi-Monte Carlo method for computing areas of point-sampled surfaces

Yu-Shen Liu[a,b,*], Jun-Hai Yong[a], Hui Zhang[a], Dong-Ming Yan[c], Jia-Guang Sun[a,b]

[a]*School of Software, Tsinghua University, Beijing 100084, People's Republic of China*
[b]*Department of Computer Science and Technology, Tsinghua University, Beijing 100084, People's Republic of China*
[c]*The University of Hong Kong, Hong Kong, China*

Received 10 April 2005; received in revised form 15 July 2005; accepted 18 July 2005

Abstract

A novel and efficient quasi-Monte Carlo method for computing the area of a point-sampled surface with associated surface normal for each point is presented. Our method operates directly on the point cloud without any surface reconstruction procedure. Using the Cauchy–Crofton formula, the area of the point-sampled surface is calculated by counting the number of intersection points between the point cloud and a set of uniformly distributed lines generated with low-discrepancy sequences. Based on a clustering technique, we also propose an effective algorithm for computing the intersection points of a line with the point-sampled surface. By testing on a number of point-based models, experiments suggest that our method is more robust and more efficient than those conventional approaches based on surface reconstruction.

Keywords: Point-sampled surfaces; Area; Quasi-Monte Carlo methods; Intersection

1. Introduction

In recent years, point sets have received a growing amount of attentions as an alternative surface representation method [11,19]. Although many researchers have proposed various methods for rendering and processing point sets [1, 11,19,21], little attention has been paid on how to directly compute the integral properties of point-sampled surfaces, such as area, volume, and moment. Surface area is an important property of a 3D model. It may be frequently used in many applications. For instance, the area of a geometric model should be kept unchanged before and after iteratively smoothing [19,26]. Another class of applications is in 3D shape recognition and matching which focus on the area of geometric measurements [18]. Furthermore, it seems important to compute the area of 3D point cloud data for obtaining mass properties in the reverse engineering and finite element analysis. In this paper, we study the problem of computing the area of a point-sampled surface.

Traditionally, there are many methods for computing the integral properties, such as the area, of solid models with boundary representation (B-rep) or constructive solid geometry (CSG) representation [5,12–14,27]. A review of many available methods for computing surface areas of solid models is beyond the scope of this paper. The reader may consult Ref. [14] for detailed expositions. The above existing methods require a parametric or implicit representation for boundary surfaces of solid models. However, an unorganized point set does not contain any extra information of surface, except the geometric position. One possible solution is to reconstruct a surface from the point set [3,8,17] and then uses the common techniques to compute the area of the reconstructed surface. There are several limitations on computing areas of point sets based on surface reconstruction. First, it is a non-trivial task to build a surface representation that is faithful to point sets, and the errors of the area approximation would be introduced by various surface reconstruction methods. Second, it often fails to reconstruct surfaces from large and complex point sets acquired by 3D scanning devices or sampled from multi-surfaces or non-manifold such that

* Corresponding author. Address: School of Software, Tsinghua University, Beijing 100084, People's Republic of China. Tel.: +86 13681126650.

E-mail addresses: liuyushen00@mails.tsinghua.edu.cn (Y.-S. Liu), yongjh@mail.tsinghua.edu.cn (J.-H. Yong), huizhang@mail.tsinghua.edu.cn (H. Zhang), dyman@cs.hku.hk (D.-M. Yan).

the area computation is unsuccessful. Furthermore, the surface reconstruction requires the expenditure of large amounts of time and space if the number of points is gigantic. To our knowledge, there is no published literature on computing areas of point-sampled surfaces directly.

The work most related to ours is Ref. [14], in which a quasi-Monte Carlo method for computing the surface area of a CSG model is presented by Li et al. Their method is based on the Cauchy–Crofton formula, and performs the area computation by counting the number of intersection points between the boundary surface of the CSG model and a set of uniformly distributed lines. However, since no parametric or implicit surface is given for an unorganized point set, their method cannot be applied directly to compute the area of the point-sampled surface. This paper mainly discusses how to apply the quasi-Monte Carlo method to an unorganized point set for quickly and efficiently computing the area of the point-sampled surface. The main difficulty is to determine all intersection points between a line and the point set. Traditionally, there are two approaches to compute the intersection. One indirect approach is based on surface reconstruction techniques, e.g. Refs. [3,8,17], and then uses general line/surface intersection methods. To speed up the intersection computation, Adamson et al. [2] define a point set surface approximated locally by polynomial, and then compute the intersection of a ray and the point set surface. The other approach described by Schaufler et al. [25] computes the intersection between a ray and the point set directly by placing a disk at each point of the point set without reconstructing any surface, but their method can only find one of intersection points.

In this paper, we present a novel method for computing the area of a point-sampled surface. The new method operates directly on the given point set without reconstructing any surface. Thus, it avoids the additional storage overhead and an approximation error that would be introduced by a polygonal reconstruction. Our method is based on the Cauchy–Crofton formula used in Ref. [14]. It computes the area of the point-sampled surface by counting the number of intersection points between the point set and a set of uniformly distributed lines generated using low-discrepancy sequences. For this reason, our method is also called a quasi-Monte Carlo area (QMCA) method for a point-sampled surface. To determine all intersection points between the point set and the lines, we present a new algorithm of a line and point sets intersecting (LPSI), which is based on a clustering technique. In some sense, our LPSI algorithm can be considered an extension of the ray tracing routine proposed by Schaufler et al. [25]. In addition, the approximation influences of sampling density and noise on the area computation of point sets are discussed. The major contributions of our work are as follows.

- Propose the formula for area computation of the point-sampled surface. A novel QMCA method is presented for directly computing the area of a point-sampled surface without any surface reconstruction procedure. This method can be applied to large and complex point sets acquired by 3D scanning devices or sampled from multi-surfaces (or non-manifold).
- Extend the ray tracing routine [25]. A new LPSI algorithm is designed for computing all approximated intersection points between a line and a point set.
- Investigate the effect of quasi-Monte Carlo integration on the discontinuous problem. The influences of sampling density and noise on the QMCA method are discussed.

The remainder of this paper is organized as follows. In Section 2, the procedure of the QMCA method is presented. In Section 3, the LPSI algorithm is described. In Section 4, some experimental results are introduced. In Section 5, the influences of sampling density and noise on the QMCA method are discussed, and the comparisons with the methods based on surface reconstruction are presented. Finally, we give conclusions in Section 6.

2. Computing the area of a point-sampled surface

2.1. Definition of the area of a point-sampled surface

Let $\wp = \{\mathbf{p}_i | 1 \le i \le M\}$ be a set of points sampled from an unknown surface $S \in \mathbb{R}^3$ with or without boundary. Because, there is no connectivity information for point sets, these local neighbors of $\mathbf{p}_i \in \wp$ are usually constructed using k-nearest neighbors determined by gathering the k points of $\wp$ nearest to $\mathbf{p}_i$ [8,11,19]; this set is denoted by $N_k(\mathbf{p}_i)$. For finding k-nearest neighbors, we use the ANN library written in C++. The ANN library approximates nearest neighbor searching based on kd-trees and box-decomposition trees, which can be found at: http://www.cs.umd.edu/~mount/ANN/. We assume that each point $\mathbf{p}_i$ of $\wp$ is equipped with a unit normal $\mathbf{n}_i$. In practice, these normals can be estimated either from scanners or by local neighboring positions [8,17]. Here, the area of a point-sampled surface, which consists of a point set $\wp$ sampled from a regular surface $S \in \mathbb{R}^3$ with or without boundary, is defined by the area of S. In this paper, our goal is to approximate the area of the point-sampled surface consisting of $\wp$ without reconstructing the underlying surface S.

2.2. Formula for surface area computation

Consider a surface in $\mathbb{R}^3$. For a line L intersecting the surface, let dL be the density of all lines intersecting the surface. Then, let m be the number of intersection points of all lines with the surface, and s the area of the surface.

The Cauchy–Crofton formula is described by Li et al. [14]:

$$\int m \, \mathrm{d}L = \pi s, \tag{1}$$

which relates the area of a surface to the number of intersection that the surface has with all lines. The reader may see Ref. [14] for more detailed derivation of the formula. Li et al. [14] use the Cauchy–Crofton formula to compute the surface area of a given 3D body (e.g. a CSG model) by Monte Carlo integration approximation. The method relates the surface area of a 3D body B to the number of intersection points between the boundary surface of B and a set of random lines in $\mathbb{R}^3$. Suppose that S_B is the boundary surface of B whose area is s. Let S_{B_1} be the boundary surface of a reference object B_1, which contains B. The area s_1 of B_1 is known. Consider a set $\mathcal{L}$ of N lines that are randomly sampled from the set of lines that intersect B_1, as shown in Fig. 1. Let n_B and n_{B_1} be the total numbers of intersection points of the lines in $\mathcal{L}$ with respect to S_B and S_{B_1}, respectively. According to Eq. (1), by the integration approximation, Li et al. [14] get

$$\frac{n_B}{N} \approx c\pi s \quad \text{and} \quad \frac{n_{B_1}}{N} \approx c\pi s_1,$$

where c is a constant of proportionality. Then, the formula for computing the surface area s of B can be given by

$$s \approx \frac{n_B}{n_{B_1}} s_1. \tag{2}$$

Based on the formula (2), Li et al.'s method is essentially a Monte Carlo method for numerical integration, and it can be applied to CSG representation, boundary representation, or surface models. However, since point sets acquired by 3D scanning devices do not give explicitly surfaces and invariably contain noise and irregular samples, we cannot compute the intersection points of a line and point sets directly.

2.3. A description of the QMCA method

We have just sketched the basis for computing the surface area of a given 3D body. Now, we apply the formula (2) to the area computation of a point-sampled surface. Suppose that a point set $\wp$ describes some underlying surface S. Analogously, the above formula (2) can be extended to the point set $\wp$. Let the area of the underlying surface S of $\wp$ be s. Suppose that S_1 is the boundary surface of a reference object B_1 containing $\wp$. The area s_1 of B_1 is known. According to the formula (2), we can give the formula for computing the surface area s of $\wp$ as follows:

$$s \approx \frac{n}{n_1} s_1, \tag{3}$$

where n and n_1 are the total number of intersection points of the lines in $\mathcal{L}$ with respect to S and S_1. To summarize, our new method for computing the area of the point set $\wp$, i.e. the so-called QMCA method, can be described as follows.

1. Generate the reference object B_1 containing the point set $\wp$ (Section 2.4).
2. Generate a set $\mathcal{L}$ of N random lines that sample the set of all lines intersecting the reference object B_1 (Section 2.5).
3. Compute the number of intersection points of the lines in $\mathcal{L}$ with respect to the reference surface S_1 and the underlying surface S of the point set $\wp$ (Section 3). Let n_1 and n denote those two numbers of intersection points, respectively.
4. Approximate the area s of S, i.e. the area of $\wp$, by $s \approx (n/n_1)s_1$.

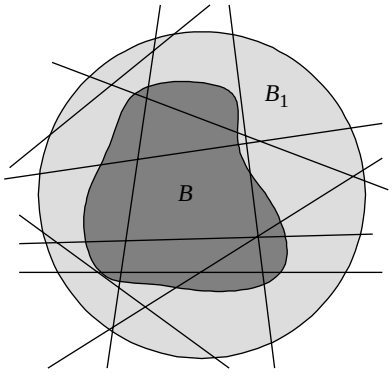


Fig. 1. Compute the area of a body B contained within the reference body B_1 using the Cauchy–Crofton formula. Random lines are chosen within B_1.

2.4. The smallest enclosing ball of point sets

In the first step of out method, there are two requirements for generating a reference body B_1, which contains the point set $\wp$. One requirement is that B_1 should been chosen as a simple object to simplify the intersection of a line with B_1. The other requirement is that the generated reference body B_1 should enclose the original object, i.e. the point set $\wp$, as closely as possible such that the approximation error derived from the Monte Carlo method is decreased [23]. In practice, the reference body B_1 is chosen as a sphere to simplify the intersection of a line with B_1 [14], where the sphere is denoted by S_R. In this paper, we use Gärtner's *miniball* method [7] to choose the smallest enclosing ball of $\wp$ as S_R. The method is very fast in low dimensions, e.g. three dimensions. An alternative might be to consider a convex bounding box, such as an axis-parallel bounding box of $\wp$, as B_1, and the correlative sampling problem is discussed by Castro et al. [4].

2.5. Generating uniformly distributed lines using low-discrepancy sequences

In this phase, we attempt to find a set $\mathcal{L}$ of N uniformly distributed lines that sample all lines intersecting the reference object B_1. Castro et al. [4] generate a uniform density of lines on a bounding sphere or a convex bounding

box, where the convex bounding box might be to consider an axis-parallel bounding box of point sets. Li et al. [14] use two models, called the *chord model* and *tangent model*, for sampling on a sphere. The chord model can be described as follows: a random line is defined by a line passing through two independent uniformly distributed points on a sphere S_R of radius R in $\mathbb{R}^3$. Let S_R be parameterized by

$$Q(\beta,\alpha) = (R\sin\beta\cos\alpha, R\sin\beta\sin\alpha, R\cos\beta) \tag{4}$$

where $\beta\in[0,\ \pi]$ and $\alpha\in[0,\ 2\pi)$ are shown in Fig. 2. According to the definition of the chord model, the key problem of generating uniformly distributed lines is how to obtain uniformly distributed points on the sphere. Since the area element of the sphere S_R is a function (i.e. $\sin\beta \mathrm{d}\alpha\mathrm{d}\beta$) of β, it is incorrect to select spherical coordinates β and α from uniform distributions $\beta\in[0,\ \pi]$ and $\alpha\in[0,\ 2\pi)$ for obtaining uniformly distributed points on the sphere. The more details of this result can be found at: http://mathworld.wolfram.com/SpherePointPicking.html. To obtain points such that any small area on the sphere is expected to contain the same number of points, in general there are four methods for doing this [24]. Here, we use one of these methods, which are described as follows. By setting $u=\cos\beta$, Eq. (4) can be rewritten by

$$Q(u,\alpha) = (R\sqrt{1-u^2}\cos\alpha, R\sqrt{1-u^2}\sin\alpha, Ru) \tag{5}$$

with $\alpha\in[0,\ 2\pi)$ and $u\in[-1,1]$. Using the improved parameterization given in Eq. (5), we obtain the uniformly distributed points on the sphere S_R.

The method of computing the area of a point-sampled surface based on the formula (3) is essentially a Monte Carlo method for numerical integration, with the domain of integration being the 4D parameter space. It is known that uniformly distributed random points are not distributed as evenly as so-called *low-discrepancy sequences* of points for the purpose of accurate numerical integration [14–16]. The reader may consult Refs. [15,16] for the concept of low-discrepancy sequences. One can also use rejection sampling for integration problem [28]. In this paper, we use low-discrepancy sequences, instead of pseudo-random number generators, to generate the set $\mathcal{L}$ of evenly distributed lines. For this reason, our method is also called a quasi-Monte Carlo method. Some quasi-Monte Carlo methods have already been used for computing the volume [5] and area [14] in CSG modeling. Castro et al. [4] also apply such methods to the multi-path method for radiosity. Let N be the number of sampling for approximating numerical integration. Using low-discrepancy sequences has theoretical error bound $O(N^{-1}\ 1\log^s N)$, which is much faster than the probabilistic error bound $O(N^{-1/2})$ of Monte Carlo methods using pseudo-random sequences in our present setting [5,14–16], where discrepancy is defined in a s-dimensional space.

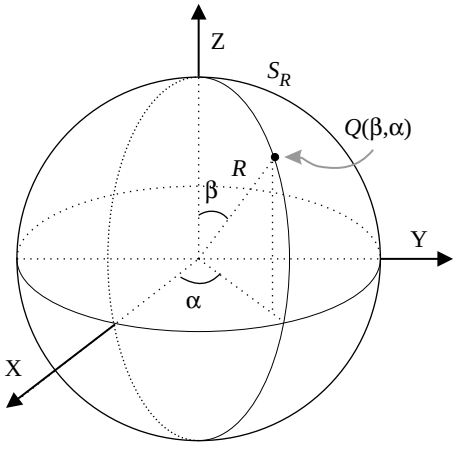


Fig. 2. The parameterization of a sphere S_R.

There exists a lot of different low-discrepancy sequences: Halton, Sobol, Niederreiter, and others [16,23]. In our case, we use Niederreiter's 4D low-discrepancy sequences of points for Eq. (5). The chord model is performed in the 4D space $(u_0, \alpha_0, u_1, \alpha_1)$, where (u_0, α_0) and (u_1, α_1) are two independent pairs of parameters for generating two independent points on the sphere S_R and the line passes through $Q(u_0, \alpha_0)$ and $Q(u_1, \alpha_1)$.

3. The LPSI algorithm

Suppose that a set $\mathcal{L}$ of uniformly distributed lines on a sphere S_R has been generated using Niederreiter's 4D low-discrepancy sequences for Eq. (5). The emphasis of our method for computing the area of the underlying surface S of the point set $\wp$ is to determine the number of intersection points between each line in $\mathcal{L}$ and $\wp$. In this section, we introduce a new algorithm of a line and point sets intersecting, called LPSI. Schaufler et al. [25] compute directly the intersection of a ray and a point set by placing a disk at each point of the point set without reconstructing any surface. Their method first intersects a cylinder around the ray with those disks. Then, the intersection is computed as a weighted average of disks whose centers are inside the cylinder. However, their method can only find one of intersection points. We extend Schaufler et al.'s approach based on a clustering technique. In contrast to previous intersection approaches [2,25], our LPSI algorithm does not reconstruct or approximate a surface from a point set while finding all intersection points of a line with the point set. The main steps of the LPSI algorithm can be described as follows.

1. Detect whether an intersection has occurred between a cylinder around a line and a point set, and collect the points inside the cylinder (Section 3.1).
2. Cluster the collected points (Section 3.2).
   2.1. Build initial clusters by projecting the collected points onto the line.
   2.2. Classify the built clusters (possibly remove some clusters or add some new clusters).

3. Return the number of intersection points, which is equal to the number of the resultant clusters. (Optional) approximate the intersection points for all resultant clusters.

We will explain each step of the LPSI algorithm in the following sections.

3.1. Collecting inclusion points

We assume that the maximum size $d_{\max}$ of a gap in the samples for a point set $\wp$ is known [25], and we will describe how to choose $d_{\max}$ in Section 4.1. Let $\mathbf{l} \in \mathcal{L}$ be a line with the parametric representation: $\mathbf{l}(t) = \mathbf{o} + t\mathbf{n}$ for some $t \in \mathbb{R}$, where $\mathbf{o}$ and $\mathbf{n}$ is an origin and a unit direction of the line $\mathbf{l}$, respectively. We can use octrees described by Adams et al. [1] or BSP trees, to accelerate the intersection point searching speed. The points of $\wp$ are considered as surfels with zero radius. Suppose that we have constructed an octree for $\wp$, and have classified the cells of the octree as two types: boundary cells containing points of $\wp$, and empty cells containing no point of $\wp$ [1]. In some point-based processing, such as Boolean operations [1] and surface reconstruction [17], the octree can be utilized. Thus, it is not an additional price for the point set.

The first step of our LPSI algorithm is similar to *intersection detection* described in Ref. [25]. The line $\mathbf{l}$ is surrounded by a cylinder of radius r, where r is set slightly larger than the largest hole $d_{\max}$ in $\wp$. An intersection is reported if the cylinder contains some points of $\wp$. We call these points inside the cylinder *inclusion points*. Let $\mathcal{C}_b = \{C_i\}$ be a set of boundary cells of the octree, where C_i is the ith boundary cell and contains part of point sets. If $\mathbf{l}$ does not intersect with C_i, we continue looking for another boundary cell until the intersection yields. All points of $\mathcal{C}_b$ within the cylinder are collected as inclusion points of $\mathbf{l}$ with $\wp$. In Fig. 3, the black points are inclusion points of a point set relative to a line $\mathbf{l}$ surrounded by a cylinder of radius r.

3.2. Clustering inclusion points

In this section, we present a projection-based clustering algorithm. The algorithm consists of two steps. The first step builds initial clusters by collecting the projection of inclusion points onto $\mathbf{l}$. In the second step, we classify these clusters for counting the number of intersection points and approximating all intersection points.

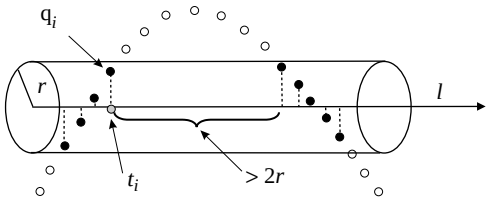


Fig. 3. Collecting initial clusters by projection.

3.2.1. Building initial clusters by projection

Clustering methods are widely used in Computer Graphics to reduce the complexity of 3D objects. For instance, Pauly et al. use region-growing and hierarchical clustering methods to simplify point-sampled surfaces [20]. Unlike Pauly et al.'s method, our clustering method maps three dimensional points into one dimensional coordinates. Suppose that $\{\mathbf{q}_i\}$ is a set of inclusion points of $\mathbf{l}$ with $\wp$. Firstly, we project each point $\mathbf{q}_i$ onto $\mathbf{l}$, and get one corresponding coordinate $t_i \in \mathbb{R}$ by the parametric representation: $\mathbf{l}(t) = \mathbf{o} + t\mathbf{n}$. So we also obtain a set $\{t_i\}$ of coordinates. Secondly, the set $\{t_i\}$ is sorted in increasing order. Thus we suppose below that $\{t_i\}$ has already been sorted. Finally, we build initial clusters by $\{t_i\}$ as follows. Starting from the minimal coordinate of $\{t_i\}$, a cluster Q_0, which is a set of some inclusion points in $\{\mathbf{q}_i\}$, is built by comparing the distance of adjacent coordinates. This cluster is terminated when the distance of two adjacent coordinates is larger than a maximum bound (we typically choose $2r$ as the bound). Then, starting from the terminated coordinate, the next cluster Q_1 is built repetitively. Clustering is terminated until the maximal coordinate is reached. Fig. 3 shows the procedure of building initial clusters.

3.2.2. Classifying clusters

Assume that $Q = \{\mathbf{q}_i \in \wp \,|\, 1 \le i \le m\}$ is one of initial clusters of $\mathbf{l}$ with $\wp$, where m is the number of inclusion points of Q. Consider that Q is sorted in increasing order of coordinates. Each inclusion point $\mathbf{q}_i$ is equipped with a unit normal $\mathbf{n}_{\mathbf{q}_i}$. We shall classify Q into four cases as follows:

Case 1: Q contains no intersection point.
Case 2: Q contains only one touching point.
Case 3: Q contains only one intersection point.
Case 4: Q contains two intersection points.

We use the following algorithm to classify Q.

1. Project each inclusion point $\mathbf{q}_i$ onto $\mathbf{l}$ and get the corresponding projection point $\mathbf{q}_i'$. If $\mathbf{n}_{\mathbf{q}_i} \cdot (\mathbf{q}_i' - \mathbf{q}_i) > 0$, is classified as *outside*; otherwise $\mathbf{q}_i'$ is classified as *inside*, where $\cdot$ denotes dot product. If all corresponding projection points of Q are outside, the cluster Q is classified as Case 1 (see Fig. 4(a)). Stop.
2. If there is no outside point and all inside points of Q are on the line $\mathbf{l}$, the cluster Q is classified as Case 2 (see Fig. 4(b)). Stop.
3. Map the unit direction $\mathbf{n}$ of $\mathbf{l}$ and each unit normal $\mathbf{n}_{\mathbf{q}_i}$ into a unit sphere. Let S_P be a plane with the normal $\mathbf{n}$ and through the sphere center (see Fig. 5). If all $\mathbf{n}_{\mathbf{q}_i}$ are at the same side of S_P, the cluster Q is classified as Case 3 (see Fig. 4(c)). Stop.
4. If all $\mathbf{n}_{\mathbf{q}_i}$ are not at the same side of S_P, the cluster Q is classified as Case 4 (see Fig. 4(d)). Stop.

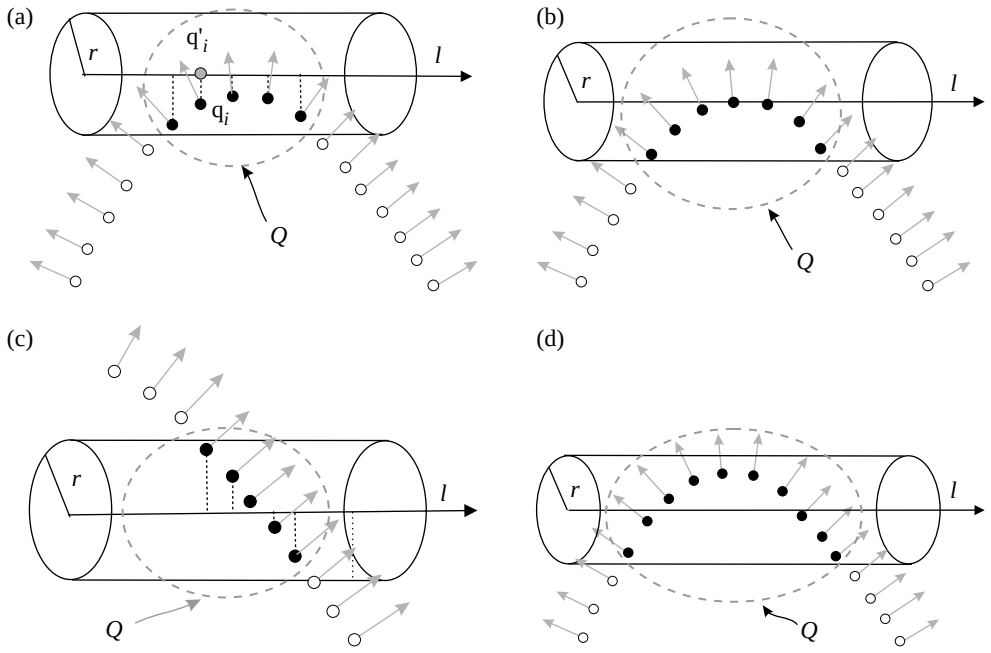


Fig. 4. Analyzing clustering. (a) Q contains no intersection point. (b) Q contains only one touching point. (b) Q contains only one intersection point. (d) Q contains two intersection points.

In general, compared with the point set $\wp$, the radius r of the cylinder surrounding the line $\mathbf{l}$ is too small. For this reason, we can consider the cluster Q as a sampling on a local convex (concave, or flat) patch on the underlying S. For Case 4, we split the cluster Q into two new clusters at such one inclusion point, whose normal and $\mathbf{n}_{\mathbf{q}1}$ are not at the same side of S_P. In practice, the case that Q contains multiple convex (or concave) patches generally derives from noise, and we will discuss the influence of noise in Section 5.2.

After classifying clusters, we choose the number of resultant clusters as the number of intersection points of the line $\mathbf{l}$ with the point set $\wp$. Although only the number of intersection points of $\mathbf{l}$ with $\wp$ is relative to the area computation of $\wp$, it is still necessary to determine the locations of the intersection points of $\mathbf{l}$ with $\wp$ for other applications, such as ray tracing point-sampled geometry [25]. Suppose that $Q=\{\mathbf{q}_i \in \wp \,|\, 1 \le i \le m\}$ is one of intersection clusters of $\mathbf{l}$ with $\wp$. The intersection point of one cluster Q is given by the average of projecting the points $\mathbf{q}_i$ onto the line $\mathbf{l}$. Let $\mathbf{y}$ be a point on the line $\mathbf{l}$. The intersection point of one cluster Q can be written by $\mathbf{y}=\mathbf{o}+t\mathbf{n}$, where t is given by

$$t = \frac{1}{m}\int_{i=1}^{m} (\mathbf{q}_i - \mathbf{o})\cdot\mathbf{n}. \tag{6}$$

4. Implementation and experiments

We have applied our QMCA method to different point sets, which are sampled from some simple objects (see Fig. 6) or are obtained by 3D scanning devices (see Fig. 11). The algorithms described above are implemented in C++. The execution time is given in seconds on a Pentium IV 1.70 GHz processor with 512M RAM excluding the time of loading point sets.

4.1. Implementation details

Our QMCA algorithm involves two parameters: the number N of lines used and the radius r of the cylinder surrounding the intersection line. N is set by the user. One may choose a large N, which yields the better approximation result at the expense of computation time. We define r to be proportional to the maximum gap size $d_{\max}$ described in Section 3.1:

$$r = \lambda d_{\max}. \tag{7}$$

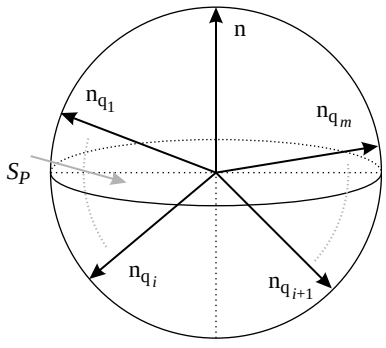


Fig. 5. Map the unit direction $\mathbf{n}$ of $\mathbf{l}$ and inclusion point's unit normals into a unit sphere.

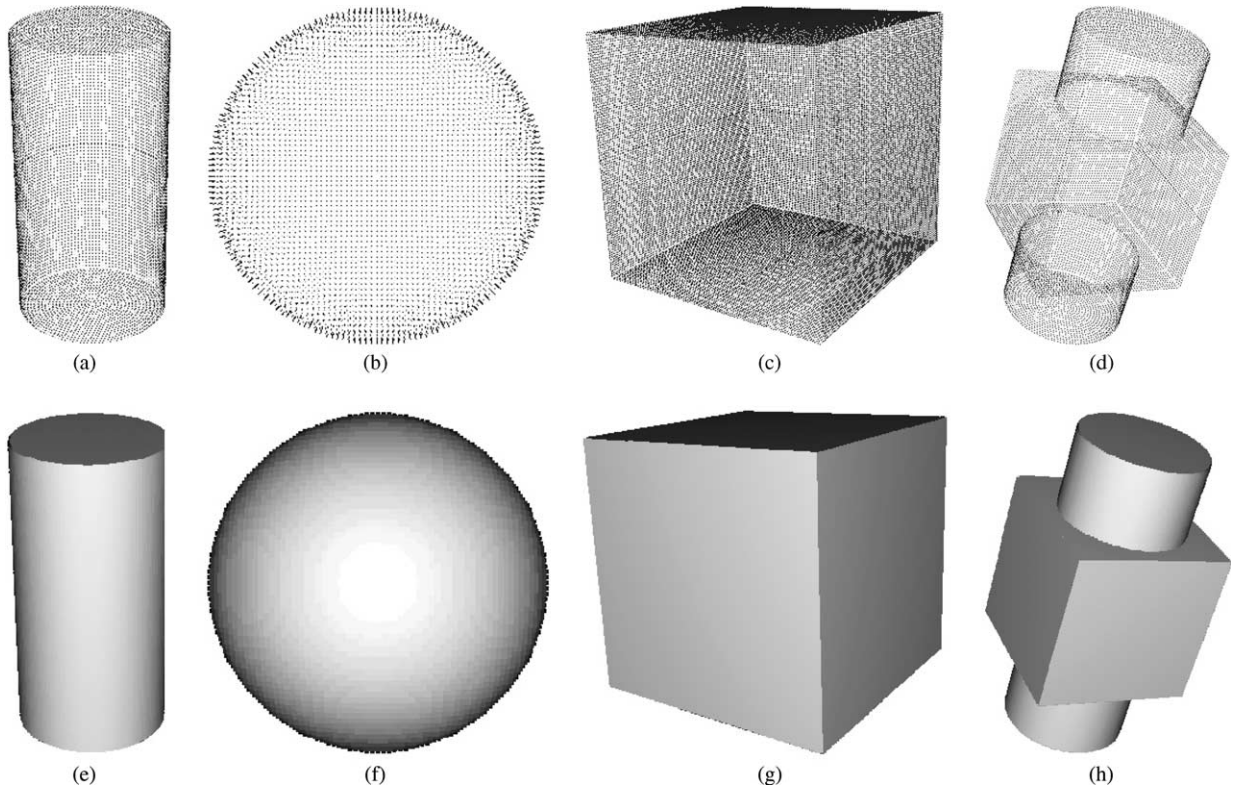


Fig. 6. Point sets sampled from some simple objects. (a) A sampling on a cylinder of radius 0.2 and height 0.8. (b) A sampling on a sphere of radius 0.4. (c) A sampling on a cube of side-length 1.0. (d) A sampling on a CSG difference between a cube of side-length 0.5 and a cylinder of radius 0.2 and height 1.0. (e)–(h) show the rendering model corresponding to (a)–(d). Note that the maximum surfel radius of each model is chosen as its $d_{\max}$.

One may choose $d_{\max}$ by two methods. For a point set with surfel representation, $d_{\max}$ is set to the maximum one of the surfel radii. For a general point set acquired by a 3D scanning device, $d_{\max}$ is set to the *mean radius* of the particular point set, where the mean radius is defined by the average over the sum of the distance between each point of the point set and its nearest neighborhood. We typically choose $\lambda = 1.5$. A larger value for λ yields the expense of computation time and increases the approximation error.

For point sets used in this section, the enclosing reference body B_1 defined in Section 2.4 is a sphere with radius R. R is chosen to be slightly larger than the radius of the smallest enclosing ball of the given point set contained by B_1. To accelerate the computation, we exploit the octree described in Section 3.1. In the preprocessing step, an axis-aligned octree of depth d (typically $d = 4$) [1] is constructed and the smallest enclosing ball of a point set is computed. In this section, we demonstrate the rendering models, represented by surfels, using the program *sview* developed by Bart Adams. For more details of this program, see the website (http://www.cs.kuleuven.ac.be/~barta/). All errors measured and given below are relative errors.

The efficiency of the quasi-Monte Carlo method for computing surface area has been described in Ref. [14]. The comparison suggests that the low-discrepancy sequences lead to smaller approximation errors than pseudo-random numbers. In this section, we do not repeat the comparison between the standard Monte Carlo method using pseudo-random numbers and the quasi-Monte Carlo method using low-discrepancy sequences. Niederreiter's sequences of points in the 4D space of $(u_0, \alpha_0, u_1, \alpha_1)$ are generated for the tangent model, as described in Section 2.5. For timing the performance and measuring the quality, a set of models with varying complexities (with respect to the number of points) have been used. In the next section, we first give some experimental results to demonstrate the convergence of the estimated areas of three point sets sampled from a cylinder, a sphere, and a cube, as shown in Fig. 6(a)–(c). For convenience, these point sets used in Fig. 6(a)–(c) are called the cylinder, the sphere, and the cube, respectively. Fig. 6(d) shows an example of CSG object, which will be referred to as the 'cube + cylinder'. The true areas of the four models are 1.2566, 2.0106, 6.0, and 2.1283, respectively.

4.2. Approximation errors

The approximation error arises from using the LPSI method for intersecting and using low discrepancy sequences for generating lines, so the real convergence rate is determined by the LPSI's error and the quasi-Monte Carlo error. For the LPSI method, it is difficult to analyze the theoretical error bound because of the complex cases of intersection between lines and point sets.

Fig. 7 shows the curves of relative approximation errors generated by our QMCA method for the cylinder, the sphere, the cube, and the cube + cylinder, in Fig. 6(a)–(d), respectively. In Fig. 7, the number of lines is specified from 10^2 to 10^6. The reference solid lines marked with $-(1/2)$ and $-(2/3)$ in Fig. 7 are the graphs of the functions $N^{-1/2}$ and $N^{-2/3}$, respectively, for revealing the trend of the error curves, where N is the number of lines used. The standard Monte Carlo method approximates the error $O(N^{-1/2})$ and the quasi-Monte Carlo method using low-discrepancy sequences approximates the error $O(N^{-2/3})$ [14]. Fig. 7 shows that the error curves approximate the domain between $N^{-1/2}$ and $N^{-2/3}$. Those error curves in Fig. 7 suggest that the QMCA method leads to small approximation errors. In general, the more lines are chosen, the higher accuracies would be obtained. When $N = 5000$ lines are generated using Niederreiter's low-discrepancy sequences, it is expected to approximate the relative error $O(N^{-2/3})(\approx 0.0034)$ for the quasi-Monte Carlo method [14]. In practice, we also find that 5000 lines for the QMCA method can lead to both small errors and little computation time.

4.3. Execution time

Before computing the area of a point set, a preprocessing step consists of constructing an octree and computing the smallest enclosing ball. The preprocessing step can be performed in a short time. Table 1 gives the time in seconds for the preprocessing step of some point sets referred to in

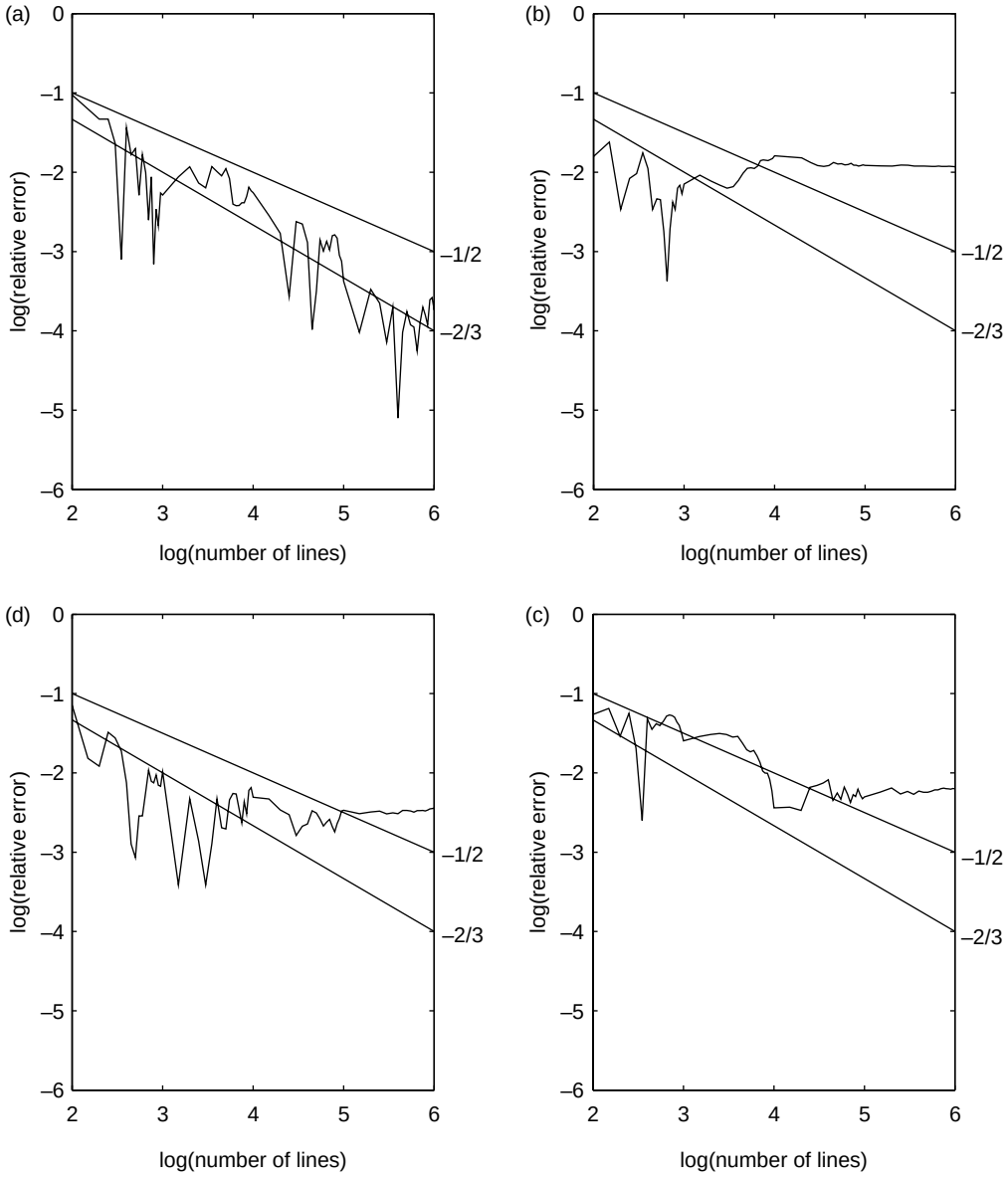


Fig. 7. (a) Approximation error for the area of the cylinder. (b) Approximation error for the area of the sphere. (c) Approximation error for the area of the cube. (d) Approximation error for the area of the cube + cylinder. Note that the dashed curves correspond to relative errors.

Table 1
Time for preprocessing

Model	Fig.	Number of points	Time1[a] (s)	Time2[b] (s)
Cylinder	6(a)	12,516	0.015375	0.015625
Sphere	6(b)	30,096	0.032250	0.062500
Cube	6(c)	65,000	0.016500	0.125000
Cube+cylinder	6(d)	21,941	0.015105	0.046875
Bunny	11(a)	35,947	0.046999	0.062500
Dragon	11(b)	437,645	0.469000	0.921875
Buddha	11(c)	543,652	0.515999	1.156250

[a] Time 1 is the time of constructing an octree at depth 4.
[b] Time 2 is the time of computing the smallest enclosing ball.

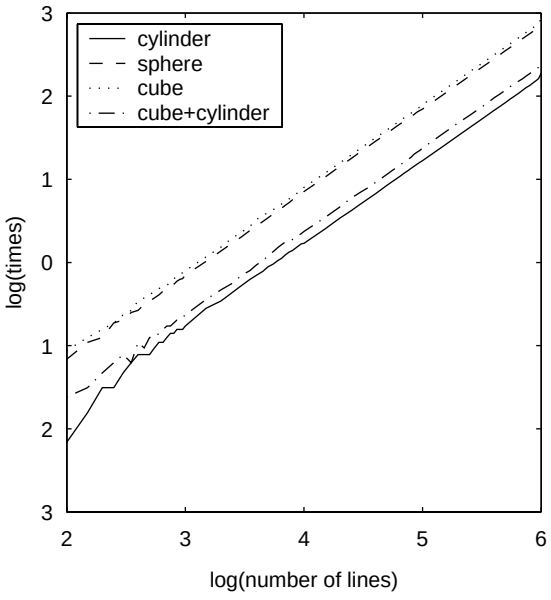


Fig. 8. Time comparison of the estimated areas of four point sets, i.e. the cylinder, the sphere, the cube, and the cube+cylinder, respectively, in Fig. 6.

this paper. The execution time shown in the following test does not include the preprocessing time.

Fig. 8 shows four curves of time, which are relative to the number of lines during computing the areas of four models: the cylinder, the sphere, the cube, and the cube+cylinder, respectively, in Fig. 6. In Fig. 8, each curve shows that the more lines are chosen, the larger computation time is required. The computation time increases approximately linearly with the number of lines used. The time also depends on the number of points of the models.

5. Discussion

Since point sets acquired by 3D scanning devices invariably contain noise and irregular samples, we have to consider their influences on the errors of the area computation for point sets.

5.1. Sampling density

From an approximation point of view, when a point set $\wp$ sampled from an underlying surface S is sufficiently dense, $\wp$ is a piecewise constant approximation of S. It is impossible to approximate the area of S at higher accuracy when insufficient sampling has occurred. We will discuss the influence of sampling density on the QMCA method in this section.

To choose the radius of the cylinder surrounding the intersection line $\mathbf{l}$, we assume that the maximum size $d_{\max}$ of a gap in samples is known in Section 3. We use *ρ-dense* proposed by Hoppe et al. [8] to define the sampling density, where *ρ-dense* is related to $d_{\max}$. The point set $\wp$ is said to be *ρ-dense* if any sphere with radius ρ and center in S contains at least one sampled point in $\wp$ [8], where ρ is called the sampling radius. For sampling on a parameter or implicit surface, ρ can be specified. For the surfel representation, ρ is the maximum one of the surfel radii. For a general point cloud, ρ can be obtained by computing the sampling density as described in Ref. [22]. In particular, if the radius $\mathbf{r}$ of the cylinder surrounding the intersection line $\mathbf{l}$ is less than ρ, it is impossible to test reliably the intersection between $\mathbf{l}$ and $\wp$. In contrast, if $\mathbf{r}$ is larger than ρ, it will spend more time on computing intersection. When $\wp$ approximates S sufficiently, less relative approximation errors will occur by our QMCA method. Fig. 9 shows the error curves with respect to the various sampling radii ρ. Fig. 9(a) shows the curves of relative approximation errors for point sets that are uniform sampling on a sphere of radius 0.4 by different sampling radii. Fig. 9(b) shows the curves of relative approximation errors for point sets that are a uniform sampling on a cylinder of radius 0.2 and height 0.8 by different sampling radii.

Highly irregularly sampled point sets are uncommon in scanned data sets. If the sampled points are not uniformly distributed over the underlying surface, the QMCA method may lead to large approximation errors. We present two simple methods to eliminate the influence of uneven sampling. One direct method is to use the resampling technique to insert new sampling points [19]. However, this method has one drawback: the resampled point set may contain too many points such that it will spend more time on preprocessing and intersecting. The other method is based on computing intersection between an *uneven cylinder* and the original point set. We use the local sampling density ρ_i described in Ref. [22] to define an uneven cylinder surrounding the line $\mathbf{l}$, where ρ_i is the *ρ-dense* of the ith boundary cell of the octree of the original point set. The uneven cylinder is defined by the intersection line $\mathbf{l}$ with different radius ρ_i when it intersects with some boundary cells in the octree.

5.2. Noise

Models created from 3D scanners usually contain noise [9,10]. Noise tends to increase errors in the LPSI algorithm. Next, we give two experimental results demonstrating the approximation influences of noise on the area computation

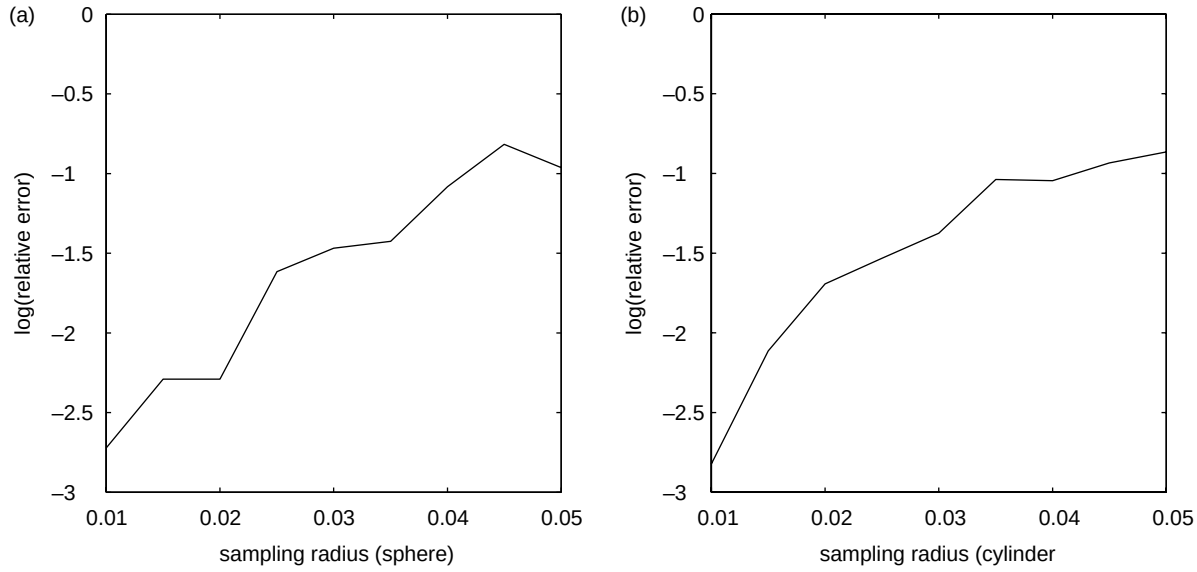


Fig. 9. (a) Errors for a sphere of radius 0.4 with different sampling radius ρ. (b) Errors for a cylinder of radius 0.2 and height 0.8 with different sampling radius ρ.

of point sets. Fig. 10 shows two point sets sampled from a cylinder of radius 0.2 and height 0.8, and a sphere of radius 0.4, with Gaussian noise added to positions and normals with 0.001% variances. The true areas for the two models are 1.2566 and 2.0106, respectively. Table 2 shows the error comparison for these two noisy models in Fig. 10(a) and (b) using the QMCA method with the same lines used. Table 2 suggests that the relative errors of approximating noisy models are larger than the relative errors for the noiseless models. To eliminate the influence of noise, one possible scheme is to smooth the point set in preprocessing. In practice, simple, fast, and feature-preserving bilateral filtering [9,10], which removes noise from models, is used to reduce the error of area computation.

5.3. Comparison with methods based on surface reconstruction

The area of a point set can also be estimated indirectly by surface reconstruction. The MPU method, described by Ohtake et al. [17], is an implicit shape reconstruction technique with a high speed. In this section, we choose 10^{-4} as the MPU approximation accuracy, where 10^{-4} is quite sufficient for the reconstruction of fine features [17]. Table 3 shows the comparison of relative approximation errors generated by the QMCA method and the MPU method. Note that the relative errors generated by the QMCA method are always smaller than the MPU method in Table 3. Table 3 also gives the time comparison generated by the QMCA method using 5000 lines and the MPU method. Note that the processing time for the cube using the MPU method is faster than the QMCA method. This is because reconstruction methods, which also include the MPU method, depend on not only the size but also the geometric complexity of a given point set. The QMCA method depends both the size of the point set and the number of

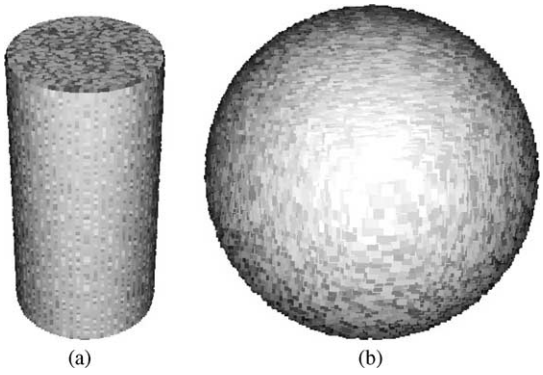


Fig. 10. Two noisy point sets sampled from some simple objects. (a) The noisy cylinder. (b) The noisy sphere.

Table 2
Error comparison with the noisy models for the QMCA method

Models	Fig.	Number of elements	Relative error noiseless models	Relative error noisy models
Cylinder	10(a)	5000 lines	0.0015	0.0054
Sphere	10(b)	5000 lines	0.0019	0.0113

Table 3
Comparison with the MPU method for simple models

Model	Method	Time(s)	Number of elements	Relative error
Cylinder	QMCA	1.015625	5000 lines	0.001506
	MPU	10.281000	32,180 triangles	0.005816
Sphere	QMCA	3.609375	5000 lines	0.001900
	MPU	26.672000	31,372 triangles	0.013713
Cube	QMCA	4.250000	5000 lines	0.004009
	MPU	3.782000	40,372 triangles	0.016475
Cube + cylinder	QMCA	2.921875	5000 lines	0.003615
	MPU	7.218999	31,776 triangles	0.014159

Table 4
Time and error comparisons for large complex point sets

Model	Method	Time(s)	Number of elements	Area	Relative error[a]
Bunny	QMCA	1.875000	5000 lines	0.057626	0.008699
	Mesh	–	69,451 triangles	0.057129	–
Dragon	QMCA	7.109375	5000 lines	0.073941	0.018527
	Mesh	–	871,414 triangles	0.072596	–
Buddha	QMCA	7.828125	5000 lines	0.053063	0.052362
	Mesh	–	1,087,716 triangles	0.055995	–

[a] Relative error is relative to the area of the corresponding mesh.

lines. Therefore, the QMCA method is suitable for large complex point sets. Furthermore, the QMCA method is suitable for open surfaces but the MPU method cannot reconstruct open surfaces [17]. The interpolation technique [3] based on the three-dimensional Voronoi diagram can efficiently reconstruct surfaces with boundary, but it requires the expenditure of large amounts of time and space when point sets are gigantic. For instance, the running time for the reconstruction of a point set sampled on a square of side-length 1.0 is about 708.359375 s using Amenta et al.'s method [3], where the point set contains about 10,000 points. The area relative error generated using the QMCA method with 5000 lines is about 1.7856% and its running time is about 0.390625 s.

The QMCA method is suitable for large complex point sets. Table 4 compares the errors for three reconstructed triangle meshes: the Bunny, the Dragon, and the Buddha, from the Stanford 3D scanning repository. Those reconstructed models can be found at: http://graphics.stanford.edu/data/3Dscanrep/. Using Desbrun's method [6] for

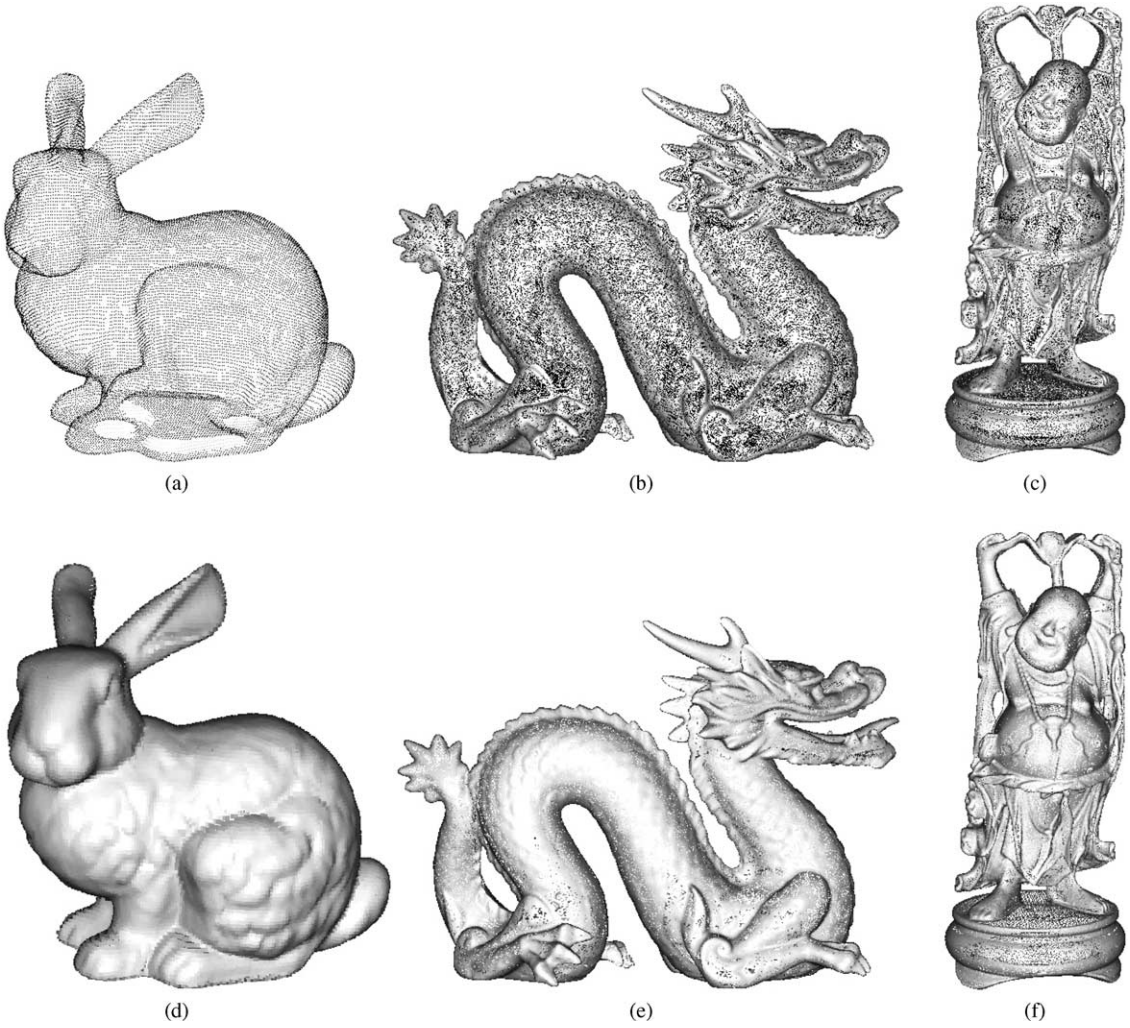


Fig. 11. Three large complex point sets from the Stanford 3D scanning repository. (a) The Bunny model. (b) The Dragon model. (c) The Buddha model. (d)–(f) are the rendering models corresponding to (a)–(c).

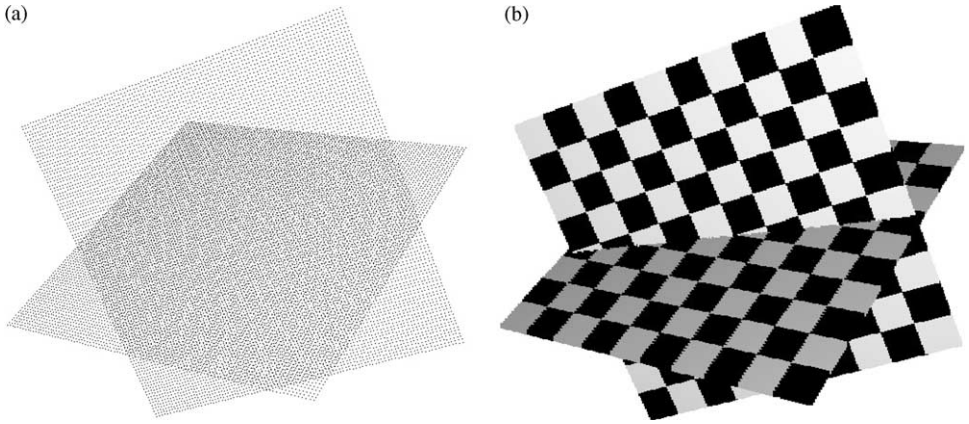


Fig. 12. (a) A point set sampled from two squares. (b) The rendering model corresponding to (a) with checkerboard texture.

computing the areas of triangle meshes, we obtain three *true areas*: 0.057129, 0.072596 and 0.055995, of the above reconstructed models. Now, we extract three point sets from these triangle meshes as follows: point positions are acquired by vertices from the triangle meshes; the normal at a point is computed as the weighted average (by the area of the triangles) of the normals to the triangles in the 1-ring neighborhood of the vertex. Fig. 11(a)–(c) shows those extracted point sets. Table 4 shows time and error comparisons for these complex point sets. In Table 4, the 'Mesh' item of each model is the reconstructed Stanford triangle mesh, and the 'Relative error' item is defined by the error relative to the corresponding true area (i.e. the area of the reconstructed mesh). In general, point sets acquired by 3D scanning devices typically have uneven sampling density, resulting in larger approximation errors. Fig. 11(d)–(f) show the rendering models using surfels with the nonzero radius that is equal to the mean radius of the point set. Some small black holes can be found in the Dragon and the Buddha because of the uneven sampling density. Table 4 shows that the relative errors of the Dragon and the Buddha are larger than that of the Bunny, where $\lambda = 3.0$ of Eq. (7) is selected for the QMCA method.

The Cauchy–Crofton formula can also be used directly for multi-surfaces and non-manifold. If multi-surfaces are considered as a collection Σ of regular surface patches, Eq. (3) is suitable for point sets sampled from Σ. In Fig. 12, we show a point set sampled from two squares of side-length 1.0. The true area is 2.0 for those two squares. We approximate the area of the point set as 2.002553 using the QMCA method with 5000 lines, which demonstrates high accuracy of area computation for point sets sampled from the multi-surfaces.

Reconstruction methods depend on the geometric complexity of given point sets, and they may fail or form holes for some large complex point sets. Therefore, for these cases, reconstruction methods would fail on the area computation of these point sets. The failure of computing areas of point sets does not occur to the QMCA method. In some cases, e.g. 3D shape recognition and matching, one only focuses on the area of geometric measurements [18] without requiring reconstruction. Thus, reconstruction becomes an additional price. In addition, the QMCA method should be well suited to a parallel implementation due to the independent nature of lines used.

6. Conclusions

We have presented a quasi-Monte Carlo method for computing areas of point-sampled surfaces, called the QMCA method. The method is based on the Monte Carlo integration and counts the number of intersection points between point sets and a set of straight lines. We have also introduced a new algorithm: LPSI, for intersecting a line with a point set based on the clustering technique. Our experiments show that the QMCA method is fast, robust and obtains the high accuracy without requiring a reconstruction of the underlying surface from point sets. We believe that the QMCA method presented in this paper can be a good help to many point-based processing applications, such as property computation, area-preserving smoothing, and shape recognition and matching.

The major drawback in our current implementation is that the results of point classification in the LPSI algorithm are dependent on the orientation of the normal vector, i.e. whether points are inside or outside. However, surface area should not be dependent on normal orientation; besides, one orientation cannot be favored over another for an open surface. This may lead to large approximation errors. In the future, we plan to improve this step to make it sensible. Furthermore, it is an interesting topic to calculate volumes and areas of point sets with large noise.

Acknowledgements

We would like to thank Wenping Wang for many helpful discussions, and Piqiang Yu and Jean-Claude Paul for some valuable comments during our work. The authors appreciate

the comments and suggestions of the anonymous reviewers. The research was supported by Chinese 973 Program (2004CB719400), and the National Science Foundation of China (60403047). The second author was supported by the project sponsored by a Foundation for the Author of National Excellent Doctoral Dissertation of PR China (200342), and SRF for ROCS, SEM (041501004).

References

[1] Adams B., Dutré P. Interactive boolean operations on surfel-bounded solids. In: Proceedings of SIGGRAPH'03; 2003. p. 651-656.
[2] Adamson A., Alexa M. Ray tracing point set surfaces. In: Proceedings of Shape Modeling International 2003; 2003. p. 272-279.
[3] Amenta N., Bern M., Kamvysselis M. A new Voronoi-based surface reconstruction algorithm. In: Proceedings of SIGGRAPH'98; 1998. p. 415-421.
[4] Castro F., Sbert M. Application of quasi-Monte Carlo sampling to the multi-path method for radiosity. Proceedings of the Third International Conference on Monte Carlo and quasi Monte Carlo methods in scientific computing, Springer series Lecture Notes in Computational Science and Engineering, Springer Berlin, 1998.
[5] Davies T., Martin R.R. Low-discrepancy sequences for volume properties in solid modelling, Proceedings of CSG, 98 (1998) p. 13.
[6] Desbrun M., Meyer M., Schröder P., Barr A.H. Implicit fairing of irregular meshes using diffusion and curvature flow. In: Proceedings of SIGGRAPH'99; 1999. p. 317-324.
[7] Gärtner B. Fast and robust smallest enclosing balls. In: Proc. 7th Annual European Symposium on Algorithms (ESA). Volume 1643 of Lecture Notes in Computer Science, Springer-Verlag (1999), p. 325-338, 1999.
[8] Hoppe H., DeRose T., Duchamp T., McDonald J., Stuetzle W. Surface reconstruction from unorganized point. In: Proceedings of SIGGRAPH'92; 1992. p. 71-78.
[9] Jones T., Durand F., Desbrun M. Non-iterative, feature-preserving mesh smoothing. In: Proceedings of SIGGRAPH' 03; 2003. p 943-949.
[10] Jones T, Durand F, Zwicker M. Normal improvement for point rendering. IEEE Comput Graph Appl 2004;53–6.
[11] Kobbelt L, Botsch M. A survey of point-based techniques in computer graphics. Comput Graph 2004;28(6):801–14.
[12] Lee YT, Requicha AAG. Algorithms for computing the volume and other integral properties of solids, Part I. Commun ACM 1982;25(9): 635–41.
[13] Lee YT, Requicha AAG. Algorithms for computing the volume and other integral properties of solids, Part II. Commun ACM 1982;25(9): 642–50.
[14] Li X, Wang W, Martin RR, Bowyer A. Using low-discrepancy sequences and the Crofton formula to compute surface areas of geometric models. Comput Aided Design 2003;35(9):771–82.
[15] Niederreiter H. Quasi-Monte Carlo methods and pseudo-random numbers. Bull Am Math Soc 1978;84(6):957–1041.
[16] Niederreiter H. Random number generation and quasi-Monte Carlo methods. Philadelphia: SIAM Philadelphia; 1992.
[17] Ohtake Y., Belyaev A., Alexa M., Turk G., Seidel H.P. Multi-level partition of unity implicits. In: Proceedings of SIGGRAPH'03; 2003. p. 463-470.
[18] Osada R, Funkhouser T, Chazelle B, Dobkin D. Shape distributions. ACM Transact Graph 2002;21(4):807–32.
[19] Pauly M., Kobbelt L., Gross M. Multiresolution modeling of point-sampled geometry. ETH Zurich Technical Report, #378, September 16, 2002. http://graphics.stanford.edu/~mapauly/Pdfs/Multires Modeling.pdf.
[20] Pauly M., Gross M., Kobbelt L. Efficient simplification of point-sampled surfaces. In: Proceedings of IEEE Visualization'02; 2002. p. 163-170.
[21] Pauly M., Keiser R., Kobbelt L., Gross M. Shape modeling with point-sampled geometry. In: Proceedings of SIGGRAPH'03; 2003. p. 641-650.
[22] Pauly M. Point primitives for interactive modeling and processing of 3D geometry. Ph. D thesis, ETH Zurich, Switzerland, 2003.
[23] Press WH, Teukolsky SA, Vetterling WT, Flannery BP. Numerical recipes in C. 2nd ed.: Cambridge University Press; 1992.
[24] Rusin D. N-dim spherical random number drawing. In the Mathematical Atlas. http://www.math.niu.edu/~rusin/known-math/96/sph.rand.
[25] Schaufler G., Jensen H.W. Ray tracing point sampled geometry, In Proceedings of the 11th Eurographics Workshop on Rendering, (2000) 319–328
[26] Taubin G. A signal processing approach to fair surface design. In: Proceedings of SIGGRAPH'95; 1995. p. 351-358.
[27] Timmer HG, Stern JM. Computation of global properties of solid objects. Comput Aided Design 1980;12(6):301–4.
[28] Wang X. Improving the rejection sampling method in quasi-Monte Carlo methods. J Comput Appl Math 2000;114(2):231–46.



Yu-Shen Liu is a PhD student in the Department of Computer Science and Technology at Tsinghua University, China. He received his BSc in mathematics from Jilin University of China in 2000. His research interests are computer-aided design and computer graphics.



Jun-Hai Yong is an associate professor in School of Software at Tsinghua University, China. He received his BSc and PhD in computer science from the Tsinghua University, China, in 1996 and 2001, respectively. He held a visiting researcher position in the Department of Computer Science at Hong Kong University of Science and Technology in 2000. He was a postdoctoral fellow in the Department of Computer Science at the University of Kentucky from 2000 to 2002. His research interests include computer-aided design, computer graphics, computer animation, and software engineering.



Hui Zhang is an assistant professor in School of Software at Tsinghua University, China. She received her BSc and PhD in computer science from the Tsinghua University of China in 1997 and 2003, respectively. Her research interests are computer-aided design and computer graphics.



Dong-Ming Yan is a PhD student of Computer Science at the University of Hong Kong. He received his BSc and Master degrees in Computer Science from Tsinghua University, China, in 2002 and 2005, respectively. His current research interests include geometric modeling and computer-aided design.



Jia-Guang Sun is a professor in the Department of Computer Science and Technology at Tsinghua University, China. His research interests are computer graphics, computer aided design, computer-aided manufacturing, product data management and software engineering.