Computing the Discrepancy with Applications to Supersampling Patterns

David P. Dobkin* David Eppstein† Don P. Mitchell*

Abstract

Patterns used for supersampling in graphics have been analyzed from statistical and signal-processing viewpoints. We present an analysis based on a type of isotropic discrepancy—how good patterns are at estimating the area in a region of defined type. We present algorithms for computing discrepancy relative to regions that are defined by rectangles, halfplanes, and higher-dimensional figures. Experimental evidence shows that popular supersampling patterns have discrepancies with better asymptotic behavior than random sampling, which is not inconsistent with theoretical bounds on discrepancy.

1 Introduction

Supersampling is one of the most general approaches to the antialiasing problem in graphics. Since synthetic images usually cannot be prefiltered, aliasing is reduced by sampling at a very high rate, and then digitally resampling to the pixel rate. In applications like ray tracing and distribution ray tracing, this is the only general solution to aliasing currently known. In the simplest form of resampling, supersamples are averaged within a square pixel area to compute a pixel value.

Supersampling can be done in a uniform pattern, but it has been shown that there are advantages to using nonuniform or stochastic sampling patterns. Uniform sampling can lead to visually conspicuous aliasing artifacts like Moiré patterns. This is worst-case behavior in the context of adaptive sampling, where a pure high-frequency signal aliased to a pure low-frequency pattern will fool schemes for deciding where to apply extra samples [28]. Randomizing the sampling pattern leads to random-noise aliasing and is more likely to avoid the worst-case scenario for adaptive sampling.

The quality of sampling patterns has been analyzed from several viewpoints. Estimating the integral of a pixel area by averaging samples can also be viewed as a statistical sampling problem, and variance-reducing techniques of experimental design (e.g., stratification) can be applied [16, 17, 24, 25]. The Central Limit Theorem implies that pixel error will decrease with the number of samples as $O(n^{-1/2})$, but this viewpoint does not describe overall image-noise characteristics.

Some researchers have taken a *signal-processing* viewpoint of the image-sampling problem [7, 9, 18] and of the distribution ray-tracing problem [19]. Here, it has been shown that

*Computer Science Dept., Princeton University, Princeton, NJ 08544.

†Dept. of Information and Computer Science, University of California, Irvine, CA 92717.

sampling patterns can be designed to drive aliasing noise into higher frequencies, where it may be removed by the pixel-resampling process and where it is less visually conspicuous. This is achieved with samples having a high-frequency spectrum ("blue-noise").

A third viewpoint which can be applied to the problem of sample-pattern analysis is the theory of discrepancy or irregularities of distribution [2]. This viewpoint was introduced to computer graphics by Shirley [26]. Niederreiter has also pointed out the possible importance of discrepancy in computer graphics [21]. This subject grew out of the study of certain *low-discrepancy* sampling patterns which have been used in quasi-Monte Carlo integration [14, 20]. What is interesting about discrepancy is that it provides a fairly direct measurement of how good a sampling pattern is at estimating certain simple integrals. From the theory of this subject has emerged the fascinating fact that some sampling patterns yield sampling errors that are asymptotically smaller than the $O(n^{-1/2})$ of uniform random sampling. Crucial to this process are efficient algorithms for actually computing the discrepancy in various models. This topic is the focus of our work here.

Our domain in all that follows will be the unit cube $[0,1]^k$. For many graphics applications, $k = 2$; the unit square then corresponds to a single pixel of image space. However, this need not be the case. Distributed ray tracing as introduced by Cook [6] involves sampling in a space including the pixel as well as time, lens area, light areas, and other affects. The goal will be to measure the quality of a point set $X \subset [0,1]^k$. Quality will be measured as the maximum discrepancy of the set with respect to families of regions $S \subset R^k$ of a particular type. We let $\mu(S)$ the Euclidean measure of $S \cap [0,1]^k$, and $\mu_X(S)$ the discrete measure $|S \cap X|/|X|$. The *discrepancy* of X at S is measured by the formula

$$d_S(X) = |\mu(S) - \mu_X(S)|.$$

This can be interpreted as a bound on the accuracy of μ_X as an approximation to μ. If F is the set of all regions of the given type, the overall discrepancy is

$$D_F(X) = \max_{S \in F} d_S(X).$$

Typically, one wishes to construct a set X with as small a discrepancy as possible, relative to some family F. A common heuristic is based on simulated annealing. A multi-dimensional local-optimization algorithm is applied repeatedly to randomly jiggled versions of a low discrepancy pattern to search for smaller local minima. The jiggling is gradually reduced as the computation proceeds. In order to do this, we need a way of computing the discrepancy of any given set. Further, since each successive set differs from the previous ones by the inclusion of relatively few points, it will be helpful to find *dynamic* discrepancy algorithms, that is, algorithms that can update the discrepancy after points are inserted to and deleted from the set X.

Surprisingly, there seems to be little work on computation of discrepancy. In this paper, we consider the families of regions defined as halfspaces and axis oriented orthants anchored at the origin. We discuss the experimental discrepancy of various point sets. We also develop algorithms for computing the discrepancy of a point set, and for maintaining the discrepancy as the set undergoes dynamic point insertions and deletions. In addition to lying at the heart of a practical problem, the data structuring issues are non-trivial extensions of known techniques.

2 Discrepancy of Line Segments

We first consider a very simplified case of discrepancy, for one-dimensional point sets (subsets of the real line). We later use our solutions to this problem as subroutines in algorithms for higher dimensional problems of more practical relevance.

We consider discrepancy relative to line segments having the origin as one endpoint. Our measure is the difference between the length of the segment and the relative number of samples in the segment. As noted above, this is a reasonable measure of the quality of the sampling set. We note that this measure of discrepancy (in absolute value) will be realized by an interval (half-open or closed) having a point of X as its boundary: any other segment can be extended or shrunk, increasing or decreasing the length of its intersection with $[0,1]$ but not changing its intersection with X.

Thus the problem can be reformulated as follows. We are given a sequence of points x_i, $1 < i < n$, $0 \leq x_i < x_{i+1} \leq 1$, and we wish to maximize either $x_i - (i-1)/n$ or $i/n - x_i$ depending upon whether the interval is half-open or closed. If the point set is fixed, this is trivial; just compare all $O(n)$ possible such values (in linear time once the points are sorted). The problem becomes more interesting if we are allowed to insert or delete points, and we will use a data structure for this dynamic problem as part of our higher dimensional algorithms.

First note that for n fixed, both $x_i - (i-1)/n$ and $i/n - x_i$ are affine functions of the two-dimensional points with coordinates (x_i, i). The problem of maximizing a linear function relative to a set of points in the plane is well-studied: if we know the convex hull of all such points, we can use binary search to find the maximum of either function in logarithmic time. Thus we can reduce our dynamic one-dimensional discrepancy problem to one of dynamically maintaining certain two-dimensional convex hulls. When we insert or delete a new point in the one-dimensional problem, we must also adjust the positions of all succeeding points in the two-dimensional convex hull, since for each such point i will be increased or decreased by one.

We use a modification of the convex hull algorithm of Overmars and van Leeuwen [22]. We maintain a balanced binary tree representing the sorted order of points. The points themselves are stored at the leaves of the tree; the internal nodes represent subsequences of several points. If the tree is maintained as a red-black tree, each point insertion or deletion causes $O(1)$ tree rotations.

At each internal vertex v in the tree, we maintain an implicit representation of the convex hull of the points at leaves descending from v. Each such convex hull is the convex hull of the union of two linearly separated sets, one for each child of v. The convex hull at v may be produced from the hulls at its two children by adding two *bridge* edges, and removing any edges interior to the combined hull. Our representation consists simply of storing, for each red-black tree vertex, these two bridges; we do not remove the other edges. We also store at each node a count of the number of points in X descending from the node, and the positions of the bridge endpoints among the sequence of points descending from the node.

Lemma 1. *The data structure described above can be used to find the point maximizing any linear function of x_i and i, in time $O(\log n)$ per query, and can be updated if a point is* *inserted or deleted in time $O(\log^2 n)$. The space used by the data structure is $O(n)$.*

Proof: A binary search in this convex hull structure, optimizing some linear function, can be performed simply by tracing a path down through the red-black tree. As we progress down the tree, we keep track of how many points occur before the node we are examining, by using the counts of descendants. At each node, we compare the values of the linear function at the endpoints of the bridge edges; for each endpoint x_i we can determine the value of i by adding the number of points before the points in the node to the number of points in the node before x_i. If one endpoint has a larger value, the maximum of the function will be found in the corresponding child. We perform $O(1)$ work at each level, and $O(\log n)$ work overall.

It remains to show how to update this structure if we insert or delete a point. We only need to recompute the bridges and counters for the red-black tree vertices that are ancestors of the changed points or involved in a rotation, $O(\log n)$ nodes altogether. For each vertex, the counters can be updated in constant time by adding the counters for its two children. The bridges (and positions of the bridge endpoints) can be found using binary searches, in $O(\log n)$ time each, as in [22]. Thus the total time per update becomes $O(\log^2 n)$. □

We have proved the following result.

Theorem 1. *We can insert or delete points from a set $X \subset [0,1]$, and recompute the discrepancy $D(X)$ after each update, in time $O(\log^2 n)$ per update, and space $O(n)$.*

Proof: We maintain the data structure described above, in time $O(\log^2 n)$ per update. As noted earlier, the discrepancy can then be found after each update by maximizing the two linear functions $x_i - (i-1)/n$ and $i/n - x_i$. □

More generally, if we give each point x_i a weight y_i we can optimize linear functions of $(x_i, \sum y_i)$ by modifying the data structure to store these sums in place of the point counts described above. We will need this generalization for the application of this algorithm to higher dimensional halfspace discrepancy.

3 Discrepancy of Axis-Aligned Rectangles and Boxes

Assume we are given a pattern X of n samples in the unit square. For some (x, y) in the square, we can estimate the area of the rectangle $[0, x] \times [0, y]$ by counting the number of samples within it. The true area is given by the product xy and we will call the error the local discrepancy at the point (x, y):

$$d_X(x,y) = |xy - \frac{|([0,x] \times [0,y]) \cap X|}{n}|$$

(This is essentially the same as the function $d_S(X)$ defined in the introduction.) The irregularity of the distribution of samples can be measured by averaging the local discrepancy over all possible values of x and y in the square (with the obvious extension to higher dimensions). The L^∞-discrepancy is defined to be the maximum absolute value of (x, y):

$$D(X) = \sup_{x,y \in [0,1]} d_X(x,y)$$

and the L^2-discrepancy is given by:

$$T(X) = \left(\int_0^1 \int_0^1 d_X(x,y)^2 \, dx \, dy \right)^{1/2}$$

This is motivated by the importance of low-discrepancy sampling in numerical integration. For example, in one dimension, there is the significant result:

Theorem 2 (Koksma 1942). *If f is a function of bounded variation $V(f)$ on the unit interval $[0,1]$ and $X = \{x_1, \ldots x_n\}$ are points in $[0,1]$ with L^∞-discrepancy $D(X)$, then*

$$\left| \frac{1}{n} \sum_{i=1}^{n} f(x_i) - \int_0^1 f(t) dt \right| \le V(f) D(X)$$

Koksma's theorem was extended to higher dimensions [20], but the definition of bounded variation is problematic. Nevertheless, for a given function obeying the bounded-variation conditions, the error of numerical integration is $O(D_N)$. Roth has proven that, in k dimensions, the best sampling patterns have discrepancy tightly bounded by $T(X) = \Theta(n^{-1}(\log n)^{(k-1)/2})$ [2].

Sampling patterns such as the *Hammersley points* have been constructed with discrepancies of $D(X) = O(n^{-1}(\log n)^{k-1})$. Let $\phi_r(i)$ be the radical-inverse function of i base r. Its value is a real number from 0 to 1 constructed by taking the integer i, represented in base r, and reflecting its digits about the decimal point to form a fraction, base r. Given the sequence of prime numbers $2, 3, 5, \ldots$, one of n Hammersley points is given by:

$$\mathbf{x}_i = (i/n, \phi_2(i), \phi_3(i), \phi_5(i), \ldots).$$

An improvement suggested by Zaremba and generalized to higher dimensions by Warnock is based on the folded radical inverse $\psi_r(i)$ [27]. Here, the jth most significant digit a_j is replaced by $(a_j + j) \bmod r$ before the reflection about the decimal point. In the same paper, Warnock presents an $O(n^2)$ algorithm for computing $T(X)$ and experimental results for several proposed low-discrepancy patterns.

We now describe methods for computing the L^∞-discrepancy $D(X)$ of a point set X, with $n = |X|$. In two dimensions, we wish to find the quadrant of the unit square containing the origin that maximizes the discrepancy. The corner of the optimal quadrant will have as its coordinates some of the coordinates of points in X, but different coordinates may be drawn from different points so the optimal quadrant will not necessarily have a point of X as its corner. Nevertheless there are only $O(n^2)$ possibilities, and it is not hard to search through this space in $O(n^2)$ time.

We can improve this naive bound by the following plane sweep technique. Sweep a horizontal line across the unit square. Whenever we sweep across point (x_i, y_i), we insert x_i in the one-dimensional data structure described in the previous section, and find the optimal quadrant having the sweep line as its horizontal boundary. The area of this quadrant is a linear function of the x-coordinate of the vertical boundary, so this optimal quadrant can be found by an appropriate linear optimization in the one-dimensional data structure.

Thus we can compute the two-dimensional discrepancy in $O(n)$ data structure operations, for a total time of $O(n \log^2 n)$. It may be possible that a more specialized convex hull data structure such as that of Hershberger and Suri [15] can reduce this to $O(n \log n)$.

For orthants in any higher dimension, we perform a hyperplane sweep, and each time we cross a point we compute the optimal orthant having the sweep plane as part of its boundary, using the discrepancy algorithm for the next lower dimension. We summarize our results so far:

Theorem 3. *For any dimension k, we can compute $D(X)$ in time $O(n^{k-1}\log^2 n)$ and space $O(n)$.*

Proof: The proof is by induction on dimension. In each dimension, the coordinates of the optimal orthant must come from those of the input points else we could increase or decrease the orthant volume without changing the number of points it contains. Therefore the algorithm produces the correct result by only computing discrepancies at these coordinates. By induction, each discrepancy computation takes time $O(n^{k-2}\log^2 n)$, and $O(n)$ such computations are made, from which the given time bound follows. □

This improves the naive $O(n^k)$ bound, and provides the best algorithm we know of for dimensions 2 and 3. In higher dimensions, we can further improve this bound using an alternative approach which we now outline. Our algorithm for this case is based on an technique of Overmars and Yap for Klee's measure problem [23],

Instead of directly searching for the maximum discrepancy orthant, we find for each value of i from 1 to n the orthants with minimum and maximum area that contain exactly i of the n points. Once these are found, the discrepancy can then be computed in linear time.

We dualize the problem by replacing every input point with an orthant extending from that point away from the origin, and replacing potential maximum-discrepancy orthants by the points at their corners. The containment relation between points and orthants is preserved by this dualization. The problem thus becomes one of finding, in this dual arrangement of orthants, the point contained in exactly i orthants and minimizing or maximizing the product of its coordinates.

We now apply a technique of Overmars and Yap [23], to subdivide space into $O(n^{k/2})$ boxes, the positions of which depend on the input orthant arrangement. We say that an orthant of our dual problem *crosses* a box of the partition if some portion of the orthant boundary is interior to the box. We say that an orthant of our dual problem *covers* a box if the box is entirely contained in the orthant. Overmars and Yap show that, with the partition generated by their construction, each box is crossed by only $O(n^{1/2})$ orthants, and in a special way: the only boundary points of any orthant that are interior to a box are those in the relative interior of a $(k-1)$-dimensional facet of the orthant. In other words, from the interior of the box, the orthant behaves as if it were simply an axis-aligned halfspace.

We compute, for each i, the minimum or maximum point contained in exactly i orthants, separately within each box of the partition. For the rest of this section we fix our attention on some particular box b. Suppose b is covered by m orthants. Since b is crossed by $O(n^{1/2})$ orthants, each point in b is contained in a number of orthants between m and $m + O(n^{1/2})$, so we need only consider values of i between these numbers.

As stated above, the orthants crossing crossing b appear as axis-aligned halfspaces with respect to containment of points in b. We sort these halfspaces by the axis to which they are perpendicular, and within each class of parallel halfspaces by the coordinates of their projection onto that axis, in time $O(n^{1/2}\log n)$. Let $h_{i,j}$ be the projected coordinate of halfplane j in the sorted list of halfplanes perpendicular to axis i.

If we only consider the halfspaces perpendicular to the first axis, we could find the minimum first coordinate of a point contained in exactly j of the halfspaces to be exactly $f_1(j) = h_{1,j}$. The maximum first coordinate is $g_1(j) = h_{1,j+1}$. If we only consider the halfspaces perpendicular to the first two axes, we can find the minimum product of the first two coordinates of points contained in exactly j halfspaces to be

$$f_2(j) = \min_{j=k+\ell} f_1(k) h_{2,\ell},$$

and in general the minimum product of the first i coordinates of points contained in exactly j halfspaces perpendicular to one of the first i coordinates will be computed by

$$f_i(j) = \min_{j=k+\ell} f_{i-1}(k) h_{i,\ell}.$$

Similarly, the maximum such product can be computed as

$$g_i(j) = \max_{j=k+\ell} g_{i-1}(k) h_{i,\ell+1}.$$

Each of these recurrences can be computed in $O(n)$ total time, as there are $O(n^{1/2})$ values of j, k, and ℓ to examine.

Then the minimum and maximum products of coordinates of points contained in exactly j orthants, among points in box b, are simply $f_k(j-m)$ and $g_k(j-m)$. The points themselves can be found by a standard technique of keeping back pointers to the values giving each min or max in the computation.

Theorem 4. *For any dimension k, we can compute $D(X)$ in time $O(n^{k/2+1})$ and space $O(n)$.*

Proof: Overmars and Yap [23] show how to enumerate the boxes of the partition in the given time and space bounds, without having to keep the entire partition in memory at once.

For each of the $O(n^{k/2})$ boxes, we determine the orthants crossing it and covering it in $O(n)$ time, sort the crossing hyperplanes in $O(n^{1/2}\log n)$ time, and then perform the above computations of f_i and g_i in $O(n)$ time. Thus the total time is $O(n^{k/2+1})$. $\square$

4 Discrepancy of Arbitrary Edges and Halfspaces

Theorems such as Koksma's and the remarkable $O(n^{-1}(\log n)^{k-1})$ low-discrepancy patterns give an initial impression that vast improvements can be easily made in the efficiency of ray tracing or distribution ray tracing (by replacing Monte Carlo integration with quasi-Monte Carlo methods). This section may, to some extent, dash those hopes. First, it should be noted that it is easy to create an image which does not have bounded variation everywhere (a checkerboard viewed in perspective, for example). A more fundamental problem occurs if we notice that synthetic images contain edges and curved boundaries at arbitrary orientations. The existence of $\Theta(n^{-1}(\log n)^{(k-1)/2})$ low-discrepancy patterns is limited to the problem of estimating the area of *axis-aligned* rectangles.

Consider the problem of estimating the area of disks or of boxes at arbitrary orientation. Surprisingly, the discrepancy is much larger. Work by Schmidt and others have shown that one can do no better than the lower bound of $\Omega(n^{-1/(2k)-1/2})$ in k dimensions [2]. Upper bounds on the best discrepancy are known, and are typically larger than the lower bounds by a polylogarithmic factor (for various generalized discrepancy problems).

A common problem in computer graphics is a pixel in the neighborhood of an edge. Therefore, consider a unit square with an arbitrary line passing through it. As before, we have a pattern of samples inside the square. We can define a *arbitrary-edge discrepancy* between the area of the square above the line and the fraction of sample points above the line. Our goal is to develop an algorithm for measuring the L^∞ norm (worst-case arbitrary line) for a given sampling pattern and then to analyze the behavior of the four sampling patterns.

Theoretical analysis has been done on the similar problem of arbitrary edges through samples within a unit-area disk [2]. Bounds on the worst-case (the best pattern you can get, given the worst-case line) are $\Omega(n^{-3/4})$ and $O(n^{-3/4}(\log n)^{1/2})$. These bounds also apply to our case of points in a unit square [3, 1, 5].

We now discuss algorithms for computing the arbitrary-edge discrepancy for point sets in $[0,1]^2$. More generally we consider the *halfspace discrepancy* for point sets in $[0,1]^k$ defined to be the maximum absolute value difference between the relative number of points intersected by a halfspace and the fraction of the hypercube $[0,1]^k$ covered by the halfspace. As we note below, this higher dimensional problem has applications even in the plane, to problems of computing discrepancy with respect to circles and ellipses.

Just as with orthants, a collection of k points determines a halfspace, so one might imagine a naive algorithm that tests the discrepancy of all $O(n^k)$ such collections of points in X. However such an algorithm would not be correct: the complication is that, although in general a halfspace will be defined by k points on its boundary, the halfspace with maximum discrepancy may have fewer than k boundary points in X. Nevertheless, we can prove the following lemmas.

Lemma 2. *Let $X' \subset X \subset [0,1]^k$ be the points on the boundary of the halfspace h with maximum discrepancy. Then $\mu(h)$ will be a local maximum or minimum in the space of all halfspaces having X' on their boundaries.*

Proof: If not, we could find a different halfspace covering the same set of points but giving larger or smaller volume in $[0,1]^k$, and therefore having larger discrepancy. $\square$

Since the measure $\mu(h)$ can be expressed as an algebraic formula in h, it has a constant number of local minima. Thus we can compute the halfspace discrepancy $D_H(X)$ as follows. For each set $X' \subset X$, with $|X'| \leq d$, determine the halfspaces forming local minima of $\mu(h)$, and compute the discrepancy of each such space. There are $O(n^k)$ sets X' examined, so the total time is $O(n^{k+1})$.

This can be improved as follows. We treat sets X' with $|X'| < k$ as before, in time $O(n^k)$. Each set X' with $|X'| = k$ determines a unique halfspace $h(X')$ (assuming general position of X; if X' is not in general position we can ignore the halfspaces it generates, as they will have already been generated when we examined smaller cardinality subsets).

It remains to determine, for each set X' with $|X'| = k$, the measure $\mu(h(X'))$ and the cardinality of $X \cap h(X')$. The former can be done in constant time for fixed dimension. For the latter, we enumerate sets X' by considering all sets $Y \subset X$, with $|Y| = k - 2$, and adding all possible pairs of points to form each set X'.

If we project the space R^k perpendicularly to the affine hull of Y, two dimensions remain. We wish to know, for each pair of points (a, b) in this projected plane, the cardinality of the projection of X intersected with the halfplane below line ab. This can be solved in constant amortized time per pair using the topological sweeping algorithm of Edelsbrunner and Guibas [13]. We have proved the following:

Theorem 5. *For any dimension $k > 1$, we can compute the halfspace discrepancy $D_H(X)$ in time $O(n^k)$ and space $O(n)$.*

This algorithm is practical. Indeed, as we discuss in the next section, we have implemented an $O(n^2 \log n)$ time variant of our algorithm for R^2, as part of a searching method for generating low discrepancy point sets for sampling. The implementation computes the discrepancy of 1000 points in 204 seconds on a Sparcstation 2, as opposed to 2789 seconds for the naive method.

Our methods can also be extended to shapes bounded by algebraic curves such as spheres and ellipsoids. For example, we can consider discrepancy with respect to circles in the plane by lifting the points to the paraboloid $z = x^2 + y^2$ in R^3; each circle can then be lifted to a plane cutting the paraboloid, and we can count the number of input points in a circle as the number of lifted points below the corresponding plane. Using this method, we can find the discrepancy in time $O(n^3)$, which improves the naive $O(n^4)$ method of testing circles one at a time. There is a matching $\Omega(n^3)$ lower bound on the number of possible circles determined by the input, so any faster algorithm must eliminate many circles without computing the discrepancies of each one. Similarly, ellipses can be linearized in a five dimensional space by lifting points (x, y) to (x, y, x^2, y^2, xy), so we can compute ellipse discrepancy in time $O(n^5)$; this is again tight unless we can eliminate many ellipses without computing individual discrepancies.

The same projection method of Theorem 5 is useful in dynamic versions of the halfspace discrepancy problem. Suppose X is undergoing insertions and deletions, and consider any $Y \subset X$ of cardinality $k - 1$. Suppose $X' \supset Y$. If we project perpendicularly to the affine hull of Y, the set of halfspaces with Y on their boundaries projects to a set of lines sweeping around the point into which Y projects.

As this line sweeps from the origin through 180° back to the origin again, the measure $\mu(h)$ varies algebraically. We can partition the possible angles into $O(1)$ intervals in which the measure is monotonic; within each interval we parametrize the sweep angle by this measure, so that the measure becomes linear in the parametrized angles.

If we weight points $+1$ if they are added to the halfplane when swept across, and -1 if they are removed, then the higher dimensional discrepancy problem for halfspaces involving

Y becomes a static one-dimensional weighted discrepancy problem in the parametrized space of sweep angles. Thus we can use our one-dimensional data structure to solve this problem in time $O(\log^2 n)$ for each update not involving Y. But this also gives us a data structure for the dynamic global halfspace discrepancy problem:

Theorem 6. *For any dimension d, we can insert or delete points from a set $X \subset [0,1]^d$, and recompute the discrepancy $D_H(X)$ after each update, in time $O(n^{d-1}\log^2 n)$ per update, and space $O(n^d \log n)$.*

Proof: We simply keep a separate data structure for each possible subset Y. There are $O(n^{k-1})$ data structures for sets Y, each taking $O(\log^2 n)$ time when any point is inserted or deleted. When a point is deleted from the subset Y corresponding to one of these data structures, we simply erase the structure itself from our memory. When a point is inserted, we must create $O(n^{k-2})$ structures, in time $O(n \log n)$ each. All locally optimal halfspaces with fewer than $d-1$ points on the boundary can be enumerated in $O(n^{k-1})$ time as before. □

5 Implementation Details

In the previous section, we gave an $O(n^2)$ time algorithm for computing the maximum arbitrary-edge discrepancy of a set of n points in the unit square, and discussed its generalization to higher dimensional halfspace discrepancy. In this section we elaborate on the details needed to implement the algorithm, and discuss the experimental performance of a simplified version of the algorithm that runs in $O(n^2 \log n)$ time.

We are given as input a collection of points $P_i = (x_i, y_i)$ in the unit square. Initially, we might think we have to consider discrepancies of infinitely many lines through the unit square. As shown in the previous section, this is not the case; rather we need only consider the $O(n^2)$ lines determined by pairs of points, together with $O(1)$ additional lines per point, which are sufficient to determine the maximum arbitrary-edge discrepancy of the set.

We think of lines as being oriented, so that a line gives rise to a region of points above the line within the unit square. Points of this region are in counterclockwise orientation with respect to the line. For each line we can measure an edge discrepancy, determined as the difference between the fraction of the points P_i we expect to see in the region (i.e. the region's area) and the fraction we actually see. We show how to compute the area of the unit square above a clipping line and the number of points above that line. From these quantities, it is a simple matter to compute the edge discrepancy of the line in either orientation.

Our original points are given in primal space as coordinate pairs (x, y). Lines are expressed most naturally by equations $y = mx + b$. They can also be represented as points (the coordinate pair (m, b)) in dual space. The point (x, y) in the primal space can similarly be translated to a line $b = (-x)m + y$ in the dual space; since this is the same as the equation $y = mx + b$, a point and line will be incident in the primal space exactly when the corresponding line and point are incident in the dual space. We will move between these two spaces as needed to simplify our discussion.

	$b > 1$	$1 > b > 0$	$0 > b$
$m > 1 - b$	0	$(1-b)^2/m$	$(1-2b)/m$
$1 - b > m > -b$	$-(m+b-1)^2/m$	$2-(m+2b)$	$2-(m+b)^2/m$
$-b > m$	$(2m+2b-1)/m$	$2+b^2/m$	2

Table 1. A(m,b), twice the area of the region.

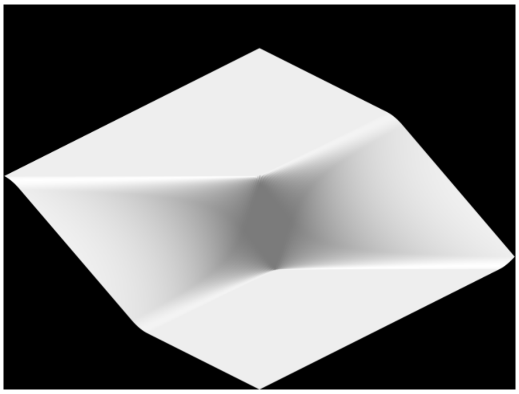

Figure 1. The Function A(m,b).

We next turn to a discussion of how to compute the area of the region of the unit square corresponding to a given (m, b) clipping line. If this area function is well-behaved, in a manner to be made precise below, it will suffice to compute the maximum discrepancy as the maximum discrepancy at any vertex of the line arrangement.

The area function is computed by a case analysis. We consider 9 classes of lines, given three ranges of the y intercept and which of three sides of the square (not on the y axis) the line passes through. For convenience, we define $A(m, b)$ as twice the area of the region in Table 1. The area $A(m, b)/2$ is displayed as the surface in Figure 1.

We observe that the area function $A(m, b)$ is continuous between the regions defined in the table. Furthermore, there are no local extrema in the area function: if a line (m, b) intersects the entire unit square, it can always be moved in some direction to increase the area of the clipped region. Similarly there is a direction of motion which will decrease this area. This leads to the observation:

Lemma 3. *Let R be any region in (m, b) space which is defined by a convex polygon and lies totally in one region of the partition defined above. Then, the extrema of $A(m, b)$ over R occur along the edges and vertices of R.*

We can extend the power of this lemma by adding the lines which define the boundaries between regions of the area function to the arrangement. This then leads to the following

Lemma 4. *Consider the set of points P_i, $i = 1, \ldots n$. If we form the arrangement of the lines $\hat{P}_i$ (where $\hat{P}_i$ is dual to P_i) with the additional lines $b = 1$, $b = 0$, $b = 1 - m$, and $b = -m$, the maximum edge discrepancy occurs along an edge or vertex of this arrangement.*

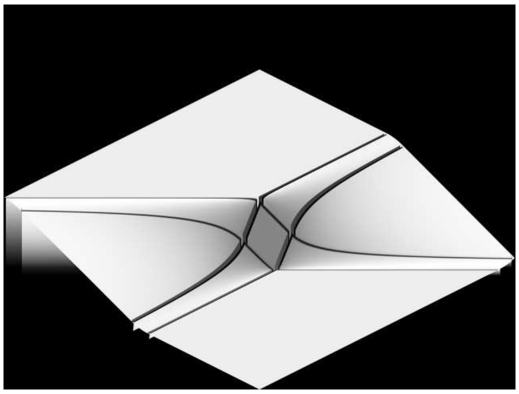

Figure 2. The Function A(m,b) with boundary lines added.

Proof: We observe that the added lines create regions in the arrangement such that each region is convex. Corresponding to each region is a subset of the P_i which lie above all lines (in primal space) represented by points (in dual space) in the region. Further, the partial derivatives of the area function are nonzero and of the same sign within the region (ie the function is also convex). The previous lemma combined with this observation implies that the extrema of the function occur along edges of the region. Since the discrepancy is defined at points of the region by subtracting a constant from the area, the extrema of the discrepancy must also occur along the edges. $\square$

Note that the added lines yield the subdivision of the area function as shown in Figure 2 illustrating the proof of the theorem. We have specified above that the regions be convex and that both partial derivatives be non-zero within a region. This assures that if we translate the origin of our coordinate system to any point of the region, we get 4 quadrants. In one of these, the area function increases as we traverse a vector. In another, the function decreases. We can make no assumptions about the other two since we are taking linear combinations of a positive and a negative quantity. Indeed there may be vectors in these quadrants along which the directional derivative passes through zero. It is these vectors that require us to consider the behavior of the area function along edges as well as at vertices.

Consider now a line in dual space, corresponding to a point in primal space. The points on the line in dual space correspond to the lines through the point in primal space. Traversing the dual line is the same as rotating a line about the primal point. Suppose we were to compute the area clipped by each line through the point. This computation is the same as computing this area at each point of the line in dual space. This area function in primal space potentially has 8 extrema. Four of these occur at the corners of the unit square (and correspond exactly to the 4 clipping lines we added to the arrangement). These correspond to situations where the shape of the clipped area changes.

The other 4 potential extrema are more subtle. These occur in situations where our rotating line clips regions through which the area function is non-monotonic even though we pass through no corner point. To see this, observe that the rotating line always divides the unit square into either 2 trapezoids or a pentagon and a triangle. In the former case, extrema are realized at vertices of the unit square and nowhere else. It is the latter case that interests us.

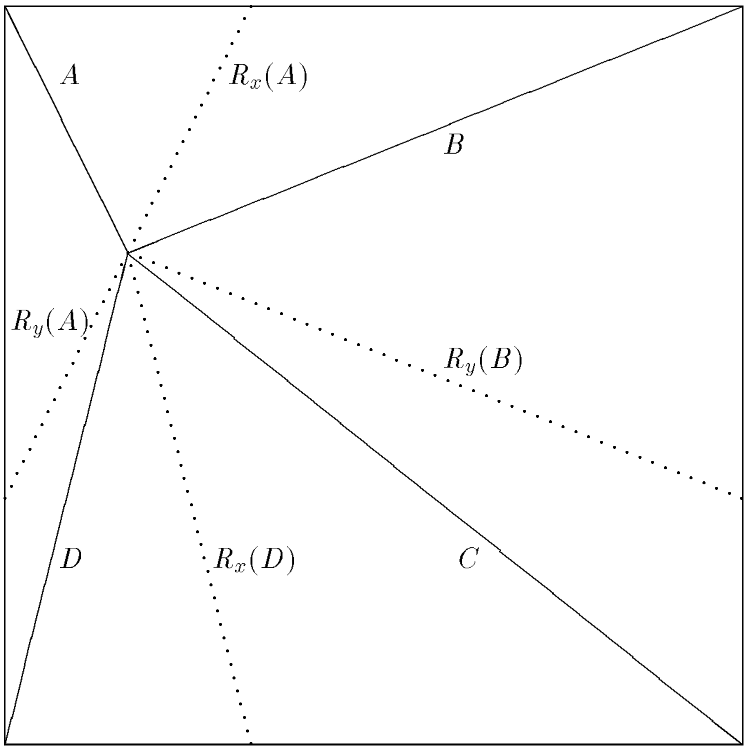

Figure 3. Extrema of the area function for line segments through a point.

Consider the counterclockwise rotation from the line labelled B to the line labelled A in Figure 3 At B, a triangle is being clipped. The vertices of this triangle lie on the line $y = 1$, at the vertex $(0, 1)$ and on the line $x = 0$. The triangle is a right triangle, so its area is the product of the lengths of its two legs along the square's boundary. Consider a line of slope m passing through the point (p, q); its area is $(1 - (q - mp))^2/m$. Differentiating, we determine that this results in an extremum when $m = -(q - 1)/p$. This extremum is realized by the segments labelled $R_x(A)$ and $R_y(A)$ in Figure 3. This construction could be translated to any of the other corners resulting in possible clipping lines of slope $-q/p$, $-q/(p - 1)$, and $-(q - 1)/(p - 1)$. We realize endpoints of these lines by the reflections shown in Figure 3. Notice that some of the reflections yield triangle vertices outside the unit square (i.e., the region is a trapazoid) and so can be ignored. Others, such as $R_x(D)$ lie within the square but lead to clipping lines that create trapezoids and so are uninteresting.

By considering all of the reflections (i.e. 4 lines of special slope for each input point) we assure that we find all extrema. Each of the cutting lines correspond to a point along the dual line. The addition of these $4n$ vertices is sufficient to assure that the maximum discrepancy occurs (ie all extrema of the area function occur) at vertices of the arrangement.

This leads to the result:

Theorem 7. *The maximum edge discrepancy for the input points P_i, $i = 1, \ldots n$ occurs at a vertex of the arrangement constructed as follows:*

1. *We form the arrangement of the lines $\hat{P}_i$ dual to the P_i.*

2. *We add to this arrangement the boundary lines $b = 1$, $b = 0$, $b = 1 - m$, and $b = -m$.*

3. *For each point $P_i = (x_i, y_i)$, we add to the corresponding dual line $\hat{P}_i$ the points of slope (i.e. $m-$coordinate in dual space) $-y_i/x_i$, $-(1-y_i)/x_i$, $-y_i/(1-x_i)$, and $-(1-y_i)/(1-x_i)$ as vertices of this arrangement.*

It remains to turn the theorem above into an algorithm. To satisfy condition (3), we can in quadratic time compute the discrepancy at each of the $4n$ slopes mentioned there. Since this running time will be dominated by anything else we do, we perform this computation and then turn to satisfying conditions (1) and (2).

We could satisfy the first two conditions by first computing the arrangement of the $n + 4$ lines (comprised of the n dual lines and the 4 additional lines). Each region of the arrangement could record the number of points dominated by lines in that region. We would then determine the $O(n^2)$ vertices of the arrangement, compute the discrepancy at each and determine the maximum discrepancy. We can examine all vertices of the arrangement in $O(n^2)$ time and $O(n)$ space by the algorithm of [13]. In practice, this approach is prone to difficulties if data are not in general position.

An alternative is to consider how vertices in the arrangement (in dual space) arise. A vertex is at the intersection of two lines, each of which is the dual of a point in primal space. In primal space, the vertex corresponds to the line connecting the 2 points. Thus, we can restate the theorem above as follows:

Theorem 8. *Given the set of points P_i, $i = 1, \ldots n$, consider the set of lines formed by connecting pairs of the $n + 4$ points consisting of the P_i enhanced by the 4 points $(0,0), (0,1), (1,0), (1,1)$. The maximum arbitrary-edge discrepancy is the edge discrepancy of one of these lines, or of one of the $4n$ lines determined by condition (3) of the previous theorem.*

This theorem gives rise to a trivial $O(n^3)$ algorithm, and to an easily implemented $O(n^2 \log n)$ algorithm for computing the discrepancy. Let p be one of the the $n + 4$ points and sort the other points radially about p. Now, we choose p and the first point in sort order and determine how many points are in the region they define from which the discrepancy of the line they define can be found. Having done so, we can update to the next point in sorted order by sweeping through the sorted list. $C(m, b)$ can be updated in constant time at each step as it differs from the previous value by $+1$ or -1. Note that the four extra points should not be counted by $C(m, b)$, just used to define extra candidate worst-case lines.

Note that if the sorted points are labeled $q_1, q_2, ...q_k$ and q_i and q_j lie in the same portion of the unit square when clipped by the line connecting p and q_r, then so do all of the points between q_i and q_j in either clockwise or counterclockwise order. We use this observation to show that we need sweep through the sorted list no more than 2 times to find the discrepancies of all lines having p as one endpoint. Finally, we compare this maximum discrepancy with that computed under condition (3) to get the true answer.

We have implemented both the naive $O(n^3)$ algorithm and the $O(n^2 \log n)$ algorithm discussed above. Timings (running time in seconds on a SparcStation 2) are given in Table 2. These times concur with our theoretical analysis and display clearly the benefit of a faster algorithm.

Points	Fast algorithm	Slow algorithm
100	1.7	3.2
200	6.8	23.9
400	28.7	178.7
800	120.2	1399.5
1600	513.2	11272.3
3200	2258.9	89351.3
6400	9554.8	738041.4

Table 2. Experimental runtimes of $O(n^2 \log n)$ and $O(n^3)$ algorithms.

Process	16 points	256 points	1600 points
Zaremba	0.0358	0.00255	0.000438
jittered	0.0501	0.00627	0.00161
Poisson	0.0900	0.0211	0.00867
Dart-Throwing	0.0521	0.00794	0.00258

Table 3. 2-Dimensional rectangular L^2-discrepancies.

6 Experimental Results

The fast algorithms for discrepancy presented above allow us to immediately explore two problems. First, we can make observations about the asymptotic behavior of some point processes. This is relevant to numerical integration, and possibly to computing form factors in progressive radiosity. Secondly, we can use this fast algorithm to search for ultra-low discrepancy patterns. This is relevant to the rendering problem where we are unlikely to place more than a few hundred samples in a pixel, but we might want to have a nearly optimal pattern.

We first consider the rectangle discrepancy as computed by the algorithms in Section 3.

Shirley computed the L^∞- and L^2-discrepancies of a number of commonly used nonuniform sampling patterns. It is worth doing a similar set of experiments again, but with progressively higher densities to get a flavor of the asymptotic behavior. We will consider four sampling patterns. The first is Zaremba's low-discrepancy pattern generated with the folded radical inverse function. The second is the jittered sampling pattern obtained by randomly perturbing a regular periodic pattern. The third is the random Poisson-distributed pattern which is approximated by generating n uniformly-distributed points on the unit hypercube (in a section of a true Poisson process, n itself would have a Poisson distribution about the mean sample density). The fourth pattern is a "Poisson-disk" pattern generated by a dart-throwing algorithm on the unit torus (promoted as the "best pattern known" by advocates of the signal-processing viewpoint).

The numbers in Table 3 are averages from 100 trials (except for the deterministic Zaremba pattern). The values for $n = 16$ agree fairly well with Shirley's results, and the

Process	16 points	256 points	1600 points
Zaremba	0.184	0.0345	0.0158
jittered	0.183	0.0296	0.00854
Dart-Throwing	0.180	0.0339	0.0118
Poisson	0.299	0.0791	0.0337
On-Line MC	0.169	0.0281	-
Off-Line MC	0.106	0.0215	-

Table 4. 2-Dimensional arbitrary-edge L^∞-discrepancies.

values for the Poisson process agree with the theoretical value of $T(X)^2 = n^{-1}(2^{-k} - 3^{-k})$ for k-dimensional patterns.

The most important feature to notice is the asymptotic behavior as n increases. For Poisson patterns, the $O(n^{-1/2})$ behavior is evidenced by the fact that increasing n by a factor of 100 only decreased the discrepancy by a factor of 10. At the other extreme, the discrepancy of Zaremba's pattern decreases at an impressive rate. The jitter and dart-throwing patterns are intermediate, but it is very interesting to note that their discrepancy seems to be better than $O(n^{-1/2})$.

We next discuss experimental results for arbitrary edge discrepancy.

We computed the L^∞ arbitrary-edge discrepancy for four important types of sampling patterns, as shown in Table 4. For the stochastic point processes (dart-throwing, jittered, Poisson), 100 trials were made to get an average for the process.

In addition, two Monte-Carlo algorithms have been used to search for patterns of low discrepancy. These are very preliminary experiments. An on-line algorithm builds up patterns one point at a time by trying to find the best next point. Given a pattern of n points, this method generates mn random points and then selects the one which causes the smallest increase in discrepancy (see [19] for discussion of a similar on-line blue-noise algorithm). An off-line algorithm starts with n random points and attempts to replace points at random if the replacement will reduce the discrepancy. The fast discrepancy computation algorithms discussed earlier are critical for the efficient performance of such methods.

Zaremba's pattern, which had dramatically low axis-aligned discrepancy, is not particularly good at dealing with arbitrary edges. Once again, random sampling is consistent with $O(n^{-1/2})$ accuracy. The jittered and dart-throwing patterns perform best. Of the two, jittered sampling shows a slightly better discrepancy, which is not consistent with its spectral and visual properties, which are inferior to dart-throwing. The Monte Carlo experiments provide a low-water mark, patterns with the lowest discrepancy that we have found. A very interesting open problem is to find the minimum-discrepancy pattern of n points.

A simulated annealing approach has also been applied to this final problem. Starting from a random pattern, there are a number of algorithms for climbing down to a local minimum. We used the "downhill simplex" algorithm to perform this local minimization in 2N dimensions [Press88]. As an example, a randomly jittered pattern of 16 points was created having a discrepancy of 0.21058531. After local minimization, a pattern resulted

N	Poisson	Jittered	Uniform	Annealed
4	0.46254099	0.40702848	0.25000000	0.25000009
9	0.38279813	0.24642622	0.16666671	0.13984264
16	0.34753269	0.21058531	0.12500000	0.09380108
25	0.14757580	0.12370891	0.10000000	0.06937759
36	0.22695184	0.11139708	0.08333347	0.05488351
49	0.18920535	0.08516756	0.07142873	0.04543021
64	0.11701021	0.06950806	0.06250000	0.03751558

Table 5. Arbitrary-edge discrepancies for some patterns.

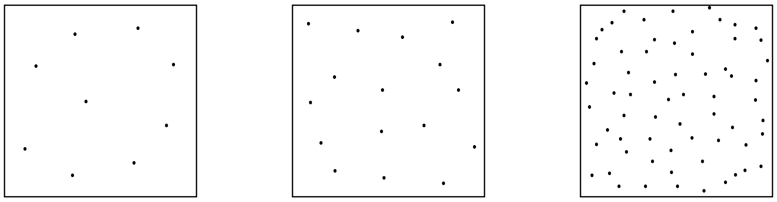

Figure 4. Low-discrepancy patterns for 9, 16, and 64 points.

with a discrepancy of 0.13278586.

This is not likely to be a global minimum, however. To find patterns of lower discrepancy, we employed a simulated-annealing heuristic. A current lowest-discrepancy pattern is jittered by a normal-distributed random amount and then locally minimized. If this gives an improvement pattern, it becomes the current lowest. This process is repeated and the standard deviation of the jitter is gradually reduced. Table 5 compares the resulting annealed patterns with several other interesting point patterns.

The simulated annealing procedure was computationally expensive and has only been used to find patterns of up to 64 points. Once again, these are likely not global minima of discrepancy, but they give what are currently the best known upper bounds. Figure 4 shows the patterns for 9, 16 and 64 points.

7 Conclusions and Open Problems

In conclusion, we have defined a arbitrary-edge discrepancy measure, motivated by the edge antialiasing problem in graphics. We have described both static and dynamic algorithms for computing rectangular and arbitrary-edge discrepancy. Two-dimensional variants of the algorithms have been implemented; we have not attempted to implement the asymptotically fastest $O(n^2)$ algorithm for arbitrary-edge discrepancy.

Applying this algorithm to some common supersampling patterns, we find evidence for convergence faster than $O(n^{-1/2})$. This is interesting, but the relative discrepancy of jittered and dart-throwing sampling are somewhat inconsistent with spectral and image-quality properties.

Numerous open problems remain which we mention here briefly.

1. Our algorithms compute the exact deficiency, but in many cases have slow running times because of the underlying combinatorial complexity. Is it possible to find approximation schemes with better running times and reasonable worst case (or average case) error bounds?

2. Given a point set and a discrepancy measure, is there an efficient algorithm to find the point which when added to the set results in the lowest discrepancy?

3. Given a point set S for which the discrepancy is known and a second point set T such that $|T| << |S|$, is there an efficient method for finding the $|T|/2$ points of T which when added to S result in the lowest discrepancy?

4. Can we solve the discrepancy problem in polynomial time for families of sets other than those given here? In particular, can we compute discrepancy with respect to the family of arbitrary convex sets, this being the general case of most interest? As mentioned earlier, we can do so in cubic time for circles, and $O(n^5)$ time for ellipses.

5. Is there a family of sets which is interesting in practice and for which discrepancy can be computed in time which does not grow exponentially with the dimension of the problem?

Since we first reported on our discrepancy algorithms [10], some further research has extended our results and partially answered some of these questions. In particular Chazelle [4] has described fast approximation algorithms for arbitrary-edge discrepancy, Dobkin et al. [11] have described algorithms for computing discrepancy with respect to arbitrary rectangles (not having the origin as a corner), and de Berg has described algorithms for strip discrepancy [8]. Dobkin et al. also note applications of discrepancy computation to other areas of computer science including computational learning theory.

Acknowledgements

Work of D. P. Dobkin was supported in part by NSF grant CCR90-02352 and by The Geometry Center, University of Minnesota, an STC funded by NSF, DOE, and Minnesota Technology, Inc. Work of David Eppstein was supported in part by NSF grant CCR-9258355. Portions of this paper have previously appeared at the 9th ACM Symp. Computational Geometry [10] and at Graphics Interface '93 [12].

References

[1] Alexander, R. Geometric methods in the study of irregularities of distribution, *Combinatorica*, 10 (1990), 115–136.

[2] J. Beck and W.W.L. Chen. *Irregularities of Distribution*, Cambridge University Press, 1987.

[3] J. Beck. Personal communication.

[4] B. Chazelle. Geometric discrepancy revisited. *Proc. 34th IEEE Symp. Found. Comp. Sci.* (1993) 392-399.

[5] B. Chazelle, J. Matoušek and M. Sharir. An elementary approach to lower bounds in geometric discrepancy. *Discrete and Computational Geometry*, to appear.

[6] R. L. Cook, Thomas Porter and Loren Carpenter, Distributed ray tracing. *Computer Graphics* 18:3 (1984) 137–145.

[7] R. L. Cook. Stochastic sampling in computer graphics. *ACM Trans. Graphics* 5:1 (1986) 51–72.

[8] M. de Berg. Computing half-plane and strip discrepancy of planar point sets. To appear.

[9] M. A. Z. Dippé and E. H. Wold. Antialiasing through stochastic sampling. *Computer Graphics* 19:3 (1985) 69–78.

[10] D. P. Dobkin and D. Eppstein. Computing the discrepancy. *Proc. 9th ACM Symp. Computational Geometry* (1993) 47–52.

[11] D. P. Dobkin, D. Gunopulos, and W. Mass. Computing the rectangle discrepancy. Manuscript, 1994.

[12] D. P. Dobkin and D. P. Mitchell. Random-edge discrepancy of supersampling patterns. *Graphics Interface*, York, Ontario (1993).

[13] H. Edelsbrunner and L. Guibas. Topologically sweeping an arrangement. *J. Comput. Sys. Sci.* 38 (1989) 165–194.

[14] J. H. Halton. A retrospective and prospective survey of the Monte Carlo method. *SIAM Review* 12 (1970) 1–63.

[15] J. Hershberger and S. Suri. Offline maintenance of planar configurations. *Proc. 2nd ACM/SIAM Symp. Discrete Algorithms* (1991) 32–41.

[16] J. T. Kajiya. The Rendering Equation. *Computer Graphics* 20 (1986) 143–150.

[17] M. Lee, R. A. Redner, and S. P. Uselton. Statistically optimized sampling for distributed ray tracing. *Computer Graphics* 19:3 (1985) 61–67.

[18] D. P. Mitchell. Generating antialiased images at low sampling densities. *Computer Graphics* 21:4 (1987) 65–72.

[19] D. P. Mitchell. Spectrally optimal sampling for distribution ray tracing. *Computer Graphics* 25:4 (1991) 157–164.

[20] H. Neiderreiter. Quasi-Monte Carlo methods and pseudo-random numbers. *Bull. Amer. Math. Soc.* 84 (1978) 957–1041.

[21] H. Neiderreiter. Quasirandom sampling computer graphics. *Proc. 3rd Internat. Sem. Digital Image Processing in Medicine*, Riga, Latvia (1992) 29–33.

[22] M. Overmars and J. van Leeuwen. Maintenance of configurations in the plane. *J. Comput. Sys. Sci.* 23 (1981) 166–204.

[23] M. Overmars and C.K. Yap. New upper bounds in Klee's measure problem. *SIAM J. Comput.* 20 (1991) 1034–1045.

[24] J. Painter and K. Sloan. Antialiased ray tracing by adaptive progressive refinement. *Computer Graphics* 23:3 (1989) 281–288.

[25] W. Purgathofer. A statistical model for adaptive stochastic sampling. *Proc. Eurographics* (1986) 145–152.

[26] P. Shirley. Discrepancy as a quality measure for sample distributions. *Proc. Eurographics* (1991) 183–193.

[27] Tony T. Warnock. Computational investigations of low-discrepancy point sets. In *Applications of Number Theory to Numerical Analysis*, S.K. Zaremba, ed., Academic Press (1971) 319–344.

[28] T. Whitted. An improved illumination model for shaded display. *Commun. Assoc. Comput. Mach.* 23 (1980) 343–349.