Computing the Discrepancy

David Dobkin* David Eppstein†

Abstract

We develop algorithms for computing the discrepancy of point sets in various Euclidean range spaces.

1 Introduction

In this paper we derive efficient algorithms and data structures for evaluating the quality of point sets for use by statistical sampling methods relevant to a central problem of computer graphics. Statistical sampling methods are used with Monte Carlo integration to approximate solutions of computational problems for which closed form solutions are not available [1, 6, 9, 12, 13, 14]. The asymptotic convergence of these approximations depends on measures of quality (typically called *discrepancy*) of the point sets where sampling occurs. We give here static and dynamic algorithms for evaluating the discrepancy of point sets. Our work is motivated by ray tracing problems of computer graphics which require sampling sets of low discrepancy.

A standard rendering technique in computer graphics is ray tracing [5]. This technique is used to produce realistic images of computer modelled scenes. These images are produced by sampling the radiance field once, or possibly many times, per pixel. This sampling corresponds to solving the rendering equation, an integral equation derived by Kajiya [8]. Ordinary ray tracing involves sampling in 2 dimensions. Distribution ray tracing [2] makes a pixel a point in a higher dimensional space so that effects such as motion blur, shadows, focusing in depth of field, ... can also be computed.

The number of samples evaluated (ie the number of rays cast) enhances the quality of the rendered image. The quality of this enhancement depends upon how the sampling points are chosen within the pixel. The standard measure of the quality of a point set is its discrepancy. This is justified by work of Koksma (see [3]) which relates the error in evaluating an integral to the discrepancy of the sampled point set. In this paper, we focus on the problem of creating data structures and algorithms for computing discrepancy and for dynamically maintaining the discrepancy in the presence of insertions and deletions.

Our domain in all that follows will be the unit cube $[0,1]^d$. The goal will be to measure the quality of a point set $X \subset [0,1]^d$. Quality will be measured as the maximum discrepancy of the set with respect to families of regions $S \subset R^n$ of a particular type. We let $\mu(S)$ the Euclidean measure of $S \cap [0,1]^d$, 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 for this involves iteratively picking points at random and including a subset chosen to decrease the discrepancy; 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. The importance

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

†Dept. of Information and Computer Science, University of California, Irvine, CA 92717. Supported in part by NSF grant CCR-9258355.

9th Annual Comptutational Geometry,5/93/CA,USA
© 1993 ACM 0-89791-583-6/93/0005/0047...$1.50

of these families in practice is described in [3]. We develop algorithms for computing the discrepancy of a static point set, and for maintaining the discrepancy as the set undergoes fully 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 Linear discrepancy

In a single dimension, halfspaces and orthants both become half-lines (rays). Further it is clear that the discrepancy will be maximized by a ray having a point of X as its boundary: any other ray 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 $i/n - x_i$. 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, if n is fixed, both $x_i - (i-1)/n$ and $i/n - x_i$ become linear functions of the two-dimensional points with coordinates (x_i, i). 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 the problem to one of dynamically maintaining convex hulls; however 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 [10]. 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.

Along with the tree itself, we maintain an implicit representation of the convex hulls of each of the point sets corresponding to a tree node. Each such convex hull is the convex hull of the union of two linearly separated sets, one for each child in the tree. Such a convex hull may be produced from the hulls of the two child sets by adding two *bridge* edges, and removing any edges interior to the combined hull. Our representation consists simply of storing, for each node, the 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.

A binary search in this convex hull structure, optimizing some linear function, can be performed simply by tracing a path down through the 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 for the nodes containing the changed points, and the counts for those nodes and their children, $O(\log n)$ nodes altogether. For each node, the counts can be updated in constant time. The bridges (and positions of the bridge endpoints) can be found using binary searches, in $O(\log n)$ time each. 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 $O(n \log n)$.*

The algorithm can actually optimize any linear function of (x_i, i). Even 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 these generalizations in our algorithms for higher dimensional halfspace discrepancy.

3 Orthants

In two dimensions, we wish to find a quadrant of the unit square containing the origin and maximizing 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 necessar-

ily 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 [7] 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:

Theorem 2. *For any dimension d, we can compute the orthant discrepancy $D_O(X)$ in time $O(n^{d-1} \log^2 n)$ and space $O(n \log n)$.*

This improves the naive $O(n^d)$ 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 based on an algorithm of Overmars and Yap for Klee's measure problem [11].

Instead of directly searching for the maximum discrepancy orthant, we find for each possible k the orthants with minimum and maximum area that contain exactly k 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 problem thus becomes one of finding, in an arrangement of orthants, the point contained in exactly k orthants and minimizing or maximizing the product of its coordinates.

We now apply a technique described by Overmars and Yap [11]. This technique subdivides space into $O(n^{d/2})$ 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 $(d-1)$-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 k, the minimum or maximum point contained in exactly k 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 only need to find minimal and maximal points in each of those numbers of orthants.

As stated above, the crossing orthants 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.

Because each halfspace points away from the origin, if we only consider the halfspaces perpendicular to the first axis, we can 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 $f_d(j-m)$ and $g_d(j-m)$. 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 3. *For any dimension d, we can compute the orthant discrepancy $D_O(X)$ in time $O(n^{d/2+1})$ and space $O(n)$.*

Proof: Overmars and Yap [11] 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 $O(n^{d/2})$ 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 $O(n^{d/2+1})$. □

4 Halfspaces

We now discuss computing the halfspace discrepancy for higher dimensional point sets.

The first complication is that, although in general a halfspace will be defined by d points on its boundary, the halfspace with maximum discrepancy may have fewer than d boundary points in X. Nevertheless, we can prove the following lemmas.

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

Since the measure $\mu(h)$ can be expressed as an algebraic formula in h, it has $O(1)$ 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 $O(n^d)$ sets X' examined, so the total time is $O(n^{d+1})$.

This can be improved as follows. We treat sets X' with $|X'| < d$ as before, in time $O(n^d)$. Each set X' with $|X'| = d$ 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 $|X'| = d$, 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 $|Y| = d-2$, and adding all possible pairs of points to form each set X'.

If we project the space R^n 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 [4]. We have proved the following:

Theorem 4. *For any dimension $d > 1$, we can compute the halfspace discrepancy $D_H(X)$ in time $O(n^d)$ and space $O(n)$.*

This algorithm is practical. Indeed, we have implemented a variant as part of a searching method for generating low discrepancy point sets for sampling (see [3]). The implementation computes the discrepancy of 1000 points in R^2, in 204 seconds on a Sparcstation 2 as opposed to 2789 seconds for the naive method.

Our methods can 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 4 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 $d-1$. 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 global halfspace discrepancy problem. We simply keep a separate data structure for each possible subset Y; there are $O(n^{d-1})$ such data structures, 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 $O(n^{d-2})$ 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 $O(n^{d-1})$ time as before.

Theorem 5. *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 $O(n^d \log n)$.*

5 Conclusions and Open Problems

We have described here the problem of discrepancy computation both statically and dynamically in a number of cases of interest. These problems were motivated by practical problems and are also of theoretical interest.

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 appoximation 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 a 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? 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?

References

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

[2] R. L. Cook, T. Porter, and L. Carpenter. Distributed ray tracing. *Computer Graphics* 18 (1984) 137–145.

[3] David P. Dobkin and Don P. Mitchell. Random-edge discrepancy of supersampling patterns. Graphics Interface '93, York, Ontario, May, 1993.

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

[5] A. Glassner. *An Introduction to Ray Tracing*, Academic Press (1989).

[6] J.H. Halton. On the efficiency of certain quasirandom sequences of points in evaluating multidimensional integrals. *Num. Math.* 2 (1960) 84–90.

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

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

[9] H. Niederreiter. Methods for estimating discrepancy. In *Applications of Number Theory to Numerical Analysis*, S.K. Zaremba, ed., Academic Press (1971) 203–236.

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

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

[12] K.F. Roth. On irregularities of distribution. *Mathematika* 1 (1954) 73–79.

[13] W.M. Schmidt. Irregularities of distribution VII. *Acta Arith.* 21 (1972) 45–50.

[14] 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.