Computing Half-Plane and Strip Discrepancy of Planar Point Sets

M. de Berg

UU-CS-1994-34
August 1994

Utrecht University

Department of Computer Science

Padualaan 14, P.O. Box 80.089,
3508 TB Utrecht, The Netherlands,
Tel. : ... + 31 - 30 - 531454

Computing Half-Plane and Strip Discrepancy of Planar Point Sets

M. de Berg

Technical Report UU-CS-1994-34
August 1994

Department of Computer Science
Utrecht University
P.O.Box 80.089
3508 TB Utrecht
The Netherlands

ISSN: 0924-3275

Computing Half-Plane and Strip Discrepancy of Planar Point Sets

Mark de Berg*

Abstract

We present efficient algorithms for two problems concerning the discrepancy of a set S of n points in the unit square in the plane. First, we describe an algorithm for maintaining the half-plane discrepancy of S under insertions and deletions of points. The algorithm runs in $O(n \log n)$ worst-case time per update, and it requires only relatively simple data structures. Second, we give an algorithm that computes the strip discrepancy of S in $O(n^2 2^{\alpha(n)} \log n)$ time, where $\alpha(n)$ is the extremely slowly growing functional inverse of Ackermann's function.

1 Introduction

For many computational problems in computer graphics a closed form solution is not available or extremely expensive to compute. In such cases statistical sampling methods can often be used to compute an approximate solution of the problem.

Consider as an example the rendering of a three-dimensional scene using ray tracing. The intensity of a pixel on the screen should correspond to the amount of light that is 'visible' in the finite area of that pixel. More precisely, one should integrate the intensity function over the pixel area. This integral is difficult to compute exactly, so in distributed ray tracing [7] the intensity of a pixel is approximated by sampling the intensity at a finite number of points. Another example is the computation of form factors for radiosity [5]. Here one wants to know for each pair of patches in the scene the fraction of the light emitted by one patch that reaches the other patch. Again, the exact computation of form factors would require evaluating a complicated (double) integral, so the form factors are usually approximated using some sampling strategy.

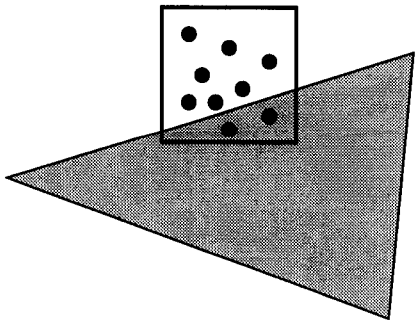

Figure 1: The polygon covers about 20% of the pixel, so we also want 20% of the points to fall within the polygon.

The quality of an approximation based on sampling depends on the distribution of the sample points. In distributed ray tracing, for example, shooting rays through points that are all very close to one corner of a pixel usually leads to a bad approximation. At first glance it may seem that uniform sample patterns—that is, grid-like patterns—will perform fine. Unfortunately, uniform patterns can lead to aliasing, and artifacts such as Moiré patterns can result. Nonuniform sampling methods are therefore to be preferred in most applications. It is clear that not all nonuniform sample patterns perform equally well, so the question arises of what constitutes a good pattern and how to find such a pattern. One way of doing this is to use variance-reducing techniques from statistical sampling theory [16, 15]. Other methods use techniques from image sampling theory [6, 8]. A third measure for the quality of sampling patterns—the one that has our interest—is the discrepancy, or irregularity of distribution [3].

*Dept. of Computer Science, Utrecht University, P.O.Box 80.089, 3508 TB Utrecht, the Netherlands. Supported by the Dutch Organisation for Scientific Research (N.W.O.) and by ESPRIT Basic Research Action No. 7141 (project ALCOM II: *Algorithms and Complexity*).

Suppose that a pixel is partially covered by a polygon, as in Figure 1. When we sample this pixel we would like the fraction of the pixel area covered by the polygon to be the same as the fraction of the sample points inside the polygon. The concept of discrepancy, which we define next, captures this type of quality measure.

Let $U = [0:1] \times [0:1]$ be the unit square in the plane, and let $S = \{p_1, \ldots, p_n\}$ be a set of n points inside the unit square. Let $\mathcal{F}$ be a family of objects in the plane. For an object $o \in \mathcal{F}$ we define $\mu(o)$, the *continuous measure* of o, to be the measure—that is, the area—of $o \cap U$. We define $\mu_S(o)$, the *discrete measure* of o with respect to S, as $\text{card}(S \cap o)/\text{card}(S)$. Here $\text{card}(\cdot)$ is used to denote the cardinality of a set. Thus the continuous measure of o is the fraction of U that is covered by o, and the discrete measure of o is the fraction of the points in S that are contained in o. The discrepancy $\Delta_S(o)$ of o with respect to S is given by

$$\Delta_S(o) := |\mu(o) - \mu_S(o)|.$$

Finally, the discrepancy $\Delta_{\mathcal{F}}(S)$ of the point set S with respect to the family $\mathcal{F}$ is the maximum discrepancy of any object in $\mathcal{F}$:

$$\Delta_{\mathcal{F}}(S) := \sup_{o \in \mathcal{F}} \Delta_S(o).$$

Discrepancy was introduced to computer graphics by Shirley [18]. His experiments show that samples with low discrepancy indeed perform well. The discrepancy that he used is the quadrant discrepancy, where the family $\mathcal{F}$ consists of all axis-parallel quadrants or, equivalently, of all axis-parallel rectangles having at least one corner coincident with a corner of U. Figure 1 suggests that the set of all quadrants may not be the best choice for the set $\mathcal{F}$. Hence, Dobkin and Mitchell [10] propose to use half-plane discrepancy, where $\mathcal{F}$ consists of all possible half-planes.

Now that we now have a measure for the quality of a sample pattern, we want to be able to evaluate it for a given pattern. That is, given a set of points in the unit square we want to compute its discrepancy with respect to a some family $\mathcal{F}$. Dobkin and Mitchell [10] describe such an algorithm for the family of half-planes. Their algorithm computes the half-plane discrepancy of a set of n points in $O(n^2)$ time. They used their algorithm to compute the half-plane discrepancy of several popular sampling patterns (Zaremba, jittered, dart-throwing, and others). Their algorithmic results were extended to higher dimensions by Dobkin and Eppstein [9], who showed that the half-space discrepancy of a point set in the d-dimensional unit cube can be computed in $O(n^d)$ time. Dobkin and Eppstein also showed how to compute orthant discrepancy in d-dimensional space in $O(\min(n^{d-1} \log^2 n,\ n^{d/2+1}))$ time. Thus they can compute the quadrant discrepancy for a set of points in the unit square in $O(n \log^2 n)$ time.

Another result on the computation of discrepancy is given by Chazelle [4]. He shows that an approximation of the half-plane discrepancy can be computed more efficiently than the exact discrepancy. More precisely, for any constant ε it is possible to compute an approximation D of $\Delta_{\mathcal{H}}(S)$ with $(1-\varepsilon)\Delta_{\mathcal{H}}(S) \leqslant D \leqslant (1+\varepsilon)\Delta_{\mathcal{H}}(S)$ in time $O(n^2/\Delta_{\mathcal{H}}(S))$. Since $\Delta_{\mathcal{H}}(S)$ cannot be smaller than $n^{1/4}$ this method is considerably faster than the exact algorithm of Dobkin and Eppstein for large values of n.

A possible heuristic to find low-discrepancy point sets is to iteratively add random points. Dobkin and Mitchell study the following variant of this process: at each iteration a small set of random points is generated, and the point whose addition results in the lowest discrepancy is added to the current set. This way they were able to construct sets with a very small discrepancy. One can compute the discrepancy of each of the sample sets in such an iterative procedure from scratch, but it would be nice if one could speed up the computation by using that the sample sets are almost identical. These considerations lead to the following question: is it possible to maintain the discrepancy of a point set under insertions into (or deletions from) the set?

Dobkin and Eppstein study this dynamic version of the discrepancy problem. They present a data structure such that the half-plane discrepancy can be maintained in $O(n \log^2 n)$ time per update. This is considerably faster than the $O(n^2)$ time that is needed when the discrepancy has to be computed from scratch. Their approach uses $O(n^2 \log n)$ storage.

We present the following two new results on the computation of discrepancy.

In Section 2 we show that half-plane discrepancy can be maintained in $O(n \log n)$ time per update, using $O(n^2)$ storage. This improves both the time and the storage bound of Dobkin and Eppstein [9] by a logarithmic factor. Furthermore, we believe that our method is simpler: whereas they use an advanced data structure (the data structure of Overmars and van Leeuwen [17] for dynamic convex hulls) our method only uses fairly standard data structures (red-black trees, and a quad-edge structure [14] for representing planar subdivisions).

Our second result, presented in Section 3, concerns the discrepancy of planar point sets with respect to the family of all possible strips. (A strip is the area enclosed by two parallel lines.) We show that the strip discrepancy can be computed in $O(n^2 2^{\alpha(n)} \log n)$ time, where $\alpha(n)$ is the extremely slowly growing functional inverse of Ackermann's function. This result is interesting from a theoretical point of view, since it gives an example where it is possible to avoid inspecting all 'combinatorially different' strips defined by the points in S, of which there can be $\Theta(n^3)$.

2 Maintaining half-plane discrepancy

Let S be a set of points in the unit square $U = [0:1] \times [0:1]$ and let $\mathcal{H}$ be the (infinite) family of all possible half-planes. We show how to maintain $\Delta_{\mathcal{H}}(S)$, the half-plane discrepancy of S, under insertions and deletions.

Although we consider all half-planes, open and closed, we may restrict our attention to open half-planes.This is allowed because the discrepancy of a half-plane h is equal to the discrepancy of the complement of h: the discrepancy of the open half-plane above (below) a line ℓ is the same as the discrepancy of the closed half-plane below (above) ℓ. Hence, the maximum discrepancy will remain the same if we only consider open half-planes. We will further restrict our attention to the positive half-planes, that is, the half-planes that lie above their bounding line (or, in case the bounding line is vertical, to the right of it). Maintaining the discrepancy with respect to the negative half-planes can be done in a completely symmetrical manner. We let ℓ^+ denote the positive open half-plane bounded by the line ℓ.

The half-plane that achieves the maximum discrepancy has some useful properties, which were already noted by Dobkin and Mitchell [10]. First, the maximum discrepancy must be realized by a half-plane ℓ^+ whose bounding line ℓ contains at least one of the points in S, because if ℓ does not contain a point we can translate ℓ such that the discrepancy increases. Now consider the lines ℓ that contain a fixed point $p \in S$. When we rotate ℓ around p then the discrete measure $\mu_S(\ell^+)$ changes only when we sweep over a point. On the other hand, the continuous measure $\mu(\ell^+)$ changes continuously. Something interesting happens to the continuous measure only when a local maximum or a local minimum of the function $\mu(\ell^+)$ is reached. A local extremum can be reached when ℓ sweeps over a corner of U or when ℓ becomes vertical. In between two such events the continuous measure is given by

$$c_1\phi + c_2 + c_3/\phi, \tag{1}$$

where ϕ denotes the slope of ℓ, and c_1, c_2, c_3 are constants that depend on the coordinates of the point p around which we rotate and on which two sides of U are intersected by ℓ. It follows that the continuous measure function only has a constant number of local extrema. The following lemma summarizes the preceding discussion.

Lemma 2.1 ([9, 10]) *Let S be a set of n points in the unit square. Suppose that ℓ^+ realizes the maximum discrepancy with respect to S over all positive half-planes. Then one of the following two cases occurs:*

- *ℓ contains one point $p \in S$, and $\mu(\ell^+)$ is a local extremum of the function $\mu(\cdot)$ in the space of all half-planes having p on their boundary*
- *ℓ contains two points of S*

Moreover, the total number of candidate half-planes of the former type is $O(n)$, and they can be computed in $O(n)$ time.

The static case. To introduce some concepts that we shall need in the dynamic case, we first sketch the algorithm of Dobkin and Mitchell[10] and Dobkin and Eppstein [9] to compute the half-plane discrepancy of a fixed set of points.

By Lemma 2.1 we only have to consider a finite set of candidate half-planes. The candidates of the first type can be handled in a brute-force way: for each of the $O(n)$ candidates ℓ^+ we compute its continuous measure $\mu(\ell^+)$ in constant time, and its discrete measure $\mu_S(\ell^+)$ in $O(n)$ time by testing every point in S for inclusion in ℓ^+. This gives us in $O(n^2)$ time all the discrepancies of candidates of the first type, of which we take the maximum.

The candidates of the second type, whose bounding line contains at least two points in S, are more difficult, since there are $\Theta(n^2)$ of them. So computing the discrete measure of each such candidate brute-force would take $\Theta(n^3)$ time. But using duality all the discrete measures can be computed in $O(n^2)$ time. We use the duality described in Edelsbrunner's book [11, Section 1.4]. Let's review some basic facts about this duality transform. A point $p = (p_x, p_y)$ is mapped to the non-vertical line

$$p^* : \quad y = 2p_x x - p_y,$$

and a non-vertical line ℓ is mapped to the point p such that $p^* = \ell$. The duality transform is not defined for vertical lines, but we will explain below that this is not a problem in our application. The important property of this duality transform is that it is *incidence preserving* and *order preserving*: a point p lies on a line ℓ if and only if the point ℓ^* lies on the line p^*, and p lies above ℓ if and only if ℓ^* lies above p^*.

Recall that we want to compute the discrete measure of all the half-planes bounded by lines through two or more points in S. In other words, for each such line ℓ we want to know the number of points in ℓ^+. Observe that the dual of the line ℓ_{pq} through two points $p, q \in S$ is the intersection of the lines p^* and q^*. (If ℓ_{pq} is vertical then the lines p^* and q^* are vertical and ℓ^*_{pq} is undefined. Fortunately, there are only $O(n)$ such vertical lines, so we can handle each of them in a brute-force way.) Moreover, the number of points in S above ℓ_{pq} is equal to the number of lines in $S^* = \{p^* : p \in S\}$ below ℓ^*_{pq}. This number of called the *(lower) level* of the vertex ℓ^*_{pq} in the arrangement $\mathcal{A}(S^*)$ defined by the lines in S^*. Hence, in the dual setting our problem becomes the following: given a set S^* of n lines, compute the level of each vertex in the arrangement $\mathcal{A}(S^*)$. A convenient representation of an arrangement of lines is the quad-edge structure [14]. This structure allows us to traverse the arrangement in a systematic way: we can visit all the edges that bound a given cell, we can step from one cell to the cell that shares a given edge with it, and so on. A quad-edge structure for an arrangement of n lines can be constructed in $O(n^2)$ time [11]. Given the structure, we can compute the level of each vertex as follows. For each line p^* in the arrangement, compute the number of lines below the leftmost vertex on the line in a brute-force manner. Next, visit all the vertices on that line in left-to-right order, meanwhile keeping track of the level of the current vertex. This is easy, because the change in level (either plus one or minus one) can be decided locally if we pass a vertex. Hence, the level of each vertex can be computed in $O(n^2)$ time in total.

Going back to primal space, we conclude that the discrete measure of all half-planes whose boundary contains two or more points can be computed in $O(n^2)$ time in total. Since the continuous measure of each half-plane can be computed in constant time, we can compute all discrepancies of half-planes of the second type in $O(n^2)$ time. We already saw how to compute the discrepancies of the first type in the same time. Hence, the half-plane discrepancy of S can be computed in $O(n^2)$ time. This is the result obtained by Dobkin and Eppstein [9] and by Dobkin and Mitchell [10].

Dynamization. We now turn our attention to the main contribution of this section, the dynamic maintenance of the discrepancy of S under insertions into and deletions from the set S. Our goal is to update the discrepancy of S in roughly linear time. Recall that the candidate half-planes for realizing the maximum discrepancy are of two types: there are $O(n)$ candidate half-planes with one point on their boundary, and there are $O(n^2)$ candidates with two or more points on their boundary.

Maintaining the maximum discrepancy of all the candidates of the first type, and of the half-planes bounded by vertical lines, is easy. We simply store each such half-plane together with the number of points in it in a linear list $\mathcal{L}$. When a new point p is inserted into S we compute in $O(1)$ time the local extrema of $\mu(\ell^+)$ in the space of all half-planes ℓ^+ having p on their boundary. These half-planes, plus the half-plane bounded by the vertical line through p, are new candidate half-planes. For each new candidate we compute its discrete measure brute-force in $O(n)$ time, after which we can compute its discrepancy in constant time. We then insert the new candidates into $\mathcal{L}$. To update the discrepancies of each old candidate half-plane in $\mathcal{L}$, we check if p is contained in it; if this is the case we update the discrete measure and compute the new discrepancy. Finally, walk along $\mathcal{L}$ to determine the new maximum discrepancy. Deletions are performed in a similar way. We obtain the following lemma.

Lemma 2.2 *The maximum discrepancy over all the half-planes that contain exactly one point of S on their boundary, and over the half-planes bounded by vertical lines through a point in S, can be maintained in $O(n)$ time per insertion or deletion, using $O(n)$ storage.*

It remains to maintain the discrepancy of the non-vertical half-planes of the second type. Let's see what happens in dual space when a new point p is inserted into S. Recall that the discrete measure of the non-vertical line ℓ_{pq} through the points $p, q \in S$ corresponds to the level of the vertex ℓ^*_{pq} in the arrangement $\mathcal{A}(S^*)$ in dual space. Adding a new point p to S corresponds to adding the line p^* to $\mathcal{A}(S^*)$. Now two things happen: the line p^* generates a number of new vertices (one for each line it intersects), and the level of some of the already existing vertices (namely the ones above p^*) increases by one. The number of vertices for which the level—and, hence, the discrepancy of the corresponding line in primal space—changes can be $\Theta(n^2)$. So if we want to do the update in close to linear time, we somehow have to avoid updating all the levels and discrepancies explicitly.

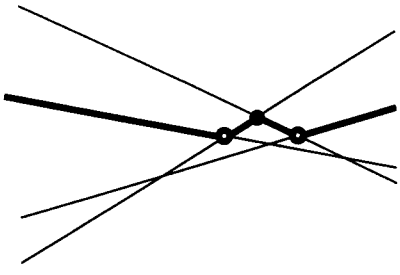

Figure 2: The 2-level in an arrangement of four lines.

Let's have a closer look at what happens to the levels when a line p^* is added to the arrangement $\mathcal{A}(S^*)$. Let V be the set of all vertices of $\mathcal{A}(S^*)$, and let $V_k \subset V$ be the set of vertices at level k. Before we can proceed we need to define the concept of *k-level* in an arrangement of n lines [11]. Let L be a set of n lines. The (lower) k-level[1] in the arrangement $\mathcal{A}(L)$, $0 \leqslant k < n$, is the set of points in the plane that lie above or on at least $k+1$ of the lines in L and below or on at least $n-k$ of the lines in L. Observe that these points must lie on the lines in L, and that a vertex at level k is part of the k-level and of the $(k+1)$-level. It is not difficult to see that the k-level is an x-monotone polygonal chain. Figure 2 gives an example; the vertex of the chain that is depicted solid belongs to the set V_2; the two other vertices of the chain belong to the set V_1.

Let's see what happens to the levels when we add the line p^* to the arrangement. We denote the set of vertices of the k-level by L_k. Recall that $L_k \subset V_k \cup V_{k-1}$. The line p^* that we add to the arrangement may intersect the polygonal chain that forms the k-level in several places. The pieces below p^* remain on the same level, whereas the pieces above p^* now belong to the $(k+1)$-level. The idea of our algorithm is to cut each level into pieces at the intersection points with p^*, as in Figure 3, and then glue the pieces together to obtain the new levels, adding pieces of the newly inserted line p^* at the appropriate positions. Since the line p^* intersects each of the already existing lines (at most) once, we have to perform at most a linear number of such splitting and glueing operations. So we want to store the levels in such a way that splitting and glueing is easy. If we use a linear list for each level, storing its set L_k of vertices in left-to-right order, then splitting and glueing is an easy operation. However, for reasons to become apparent soon, we store L_k in a balanced binary search tree $\mathcal{T}_k$ that allows for splitting and merging (concatenating) of two trees in logarithmic time. For instance, red-black trees [13] or 2-3 trees [2] have this property.

[1] In fact, what is defined next is usually considered to be the $(k+1)$-level, but in our setting it is more natural to view it as the k-level.

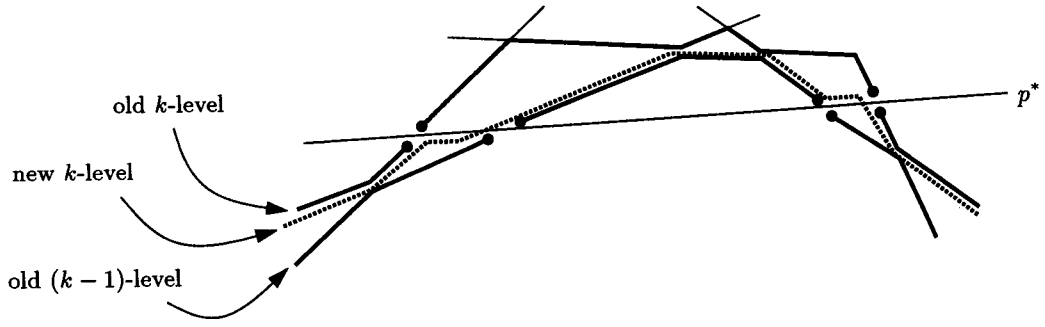

Figure 3: Constructing the new k-level.

The vertices of L_k are stored in the leaves of $\mathcal{T}_k$, ordered on the their x-coordinates. An internal node of $\mathcal{T}_k$ stores an x-value that is the maximum x-coordinate of the vertices in its left subtree. This representation allows us to split a level L_k at any given point into pieces, to glue two pieces together, and to insert a new vertex in $O(\log n)$ time. The idea is that since we have to perform only a linear number of splitting operations, we can assemble the new levels in $O(n \log n)$ time. Later we shall give more details on the assembly of the new levels; for now we continue to describe a second ingredient that is needed to maintain the discrepancy.

Define H_k to be the set of positive half-planes bounded by the duals of the vertices in V_k, and let $H := \bigcup H_k$. Using the scheme above we can maintain the levels of the vertices in V—and, hence, the discrete measures of the half-planes in H—when we insert a new line p^* into $\mathcal{A}(S^*)$. This does not buy us much, however, if we still have to recompute the discrepancy of all the half-planes in H whose discrete measure changes. To avoid this we need the following observation.

Observation 2.1 *The maximum discrepancy over all half-planes in H_k is either attained by the half-plane with the maximum continuous measure in H_k, or by the half-plane with the minimum continuous measure in H_k.*

Proof: The discrepancy $\Delta_S(h)$ of a half-plane h is defined as $\Delta_S(h) := |\mu(h) - \mu_S(h)|$. The observation now readily follows from the fact that the discrete measure $\mu_S(h)$ of all half-planes $h \in H_k$ is the same, namely k. □

So it suffices to recompute the discrepancy of the half-planes in H_k with maximum and minimum continuous measure. Because we have stored each set V_k as a red-black tree it is possible to maintain these extrema efficiently, in the following manner. For a node ν in $\mathcal{T}_k$, let $V_k(\nu)$ be the set of vertices in V_k that are stored in the subtree rooted at ν, and let $H_k(\nu)$ be the corresponding set of half-planes. Note that $V_k(\nu)$ does not necessarily contain all vertices in the subtree, as $\mathcal{T}_k$ may store vertices from V_{k-1} also. We store at node ν the two half-planes in $H_k(\nu)$ with maximum and minimum continuous measure. We denote these half-planes by $h_{\max}(\nu)$ and $h_{\min}(\nu)$, respectively. If $H_k(\nu) = \emptyset$ then we set $h_{\max}(\nu) = 0$ and $h_{\min}(\nu) = 1$. Figure 4 gives an example. The vertices of V_k are depicted as solid dots in the figure, and the vertices of V_{k-1} are depicted as small circles. The discrete measure corresponding to each vertex of V_k is written above the vertex. Observe that the half-planes in H_k with maximum and minimum continuous measure are stored at the root of $\mathcal{T}_k$. We show next that this extra information can be maintained without affecting the $O(\log n)$ time bound for these operations. (This is the reason for storing the levels in balanced search trees instead of in linear lists: if we would use linear lists, then it would be difficult to maintain the maxima and minima efficiently during splitting operations.) In fact, maintaining the maxima and minima is quite easy: for all operations we can still use the standard algorithms. After having executed the standard algorithms we must update the maxima and minima stored at each node. But these values only change at nodes for which the set of leaves in their subtree has changed. For instance, when we perform a rotation in $\mathcal{T}_k$, then we only have to adapt the values at three nodes.

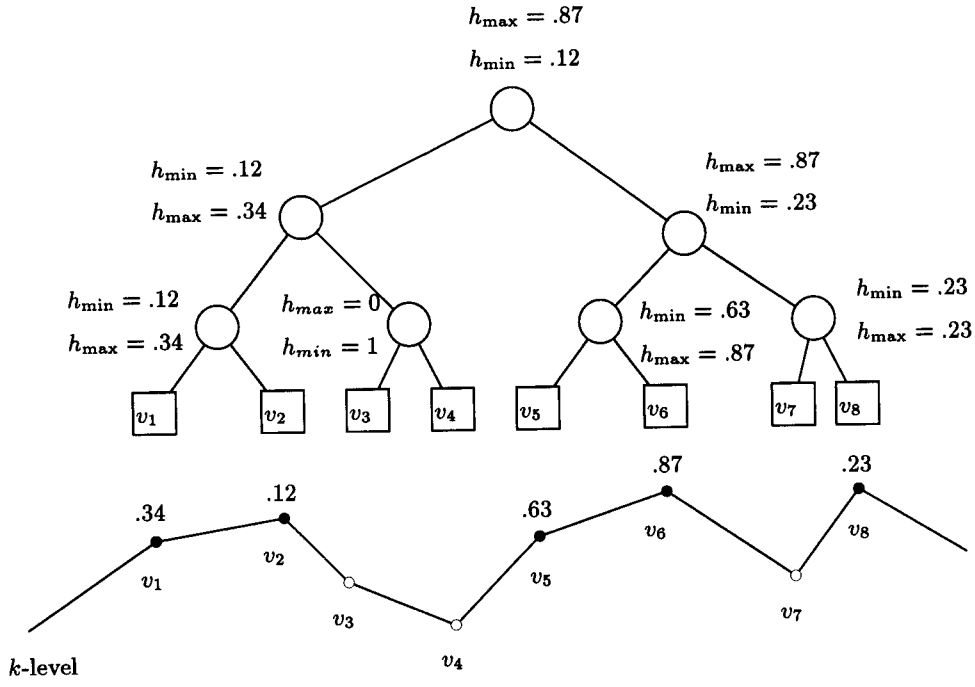

Figure 4: The tree corresponding to L_k.

We call this the set of *affected* nodes. This set has size $O(\log n)$ for the each of the operations. Recomputing the $h_{\max}$- and $h_{\min}$-values of an affected node takes constant time if we know the correct $h_{\max}$- and $h_{\min}$-values at its children. So if we treat the affected nodes in a bottom-up fashion we can update the $h_{\max}$- and $h_{\min}$-values of all affected nodes in $O(\log n)$ time.

So our algorithm basically works as follows. We compute the points where the levels are cut by the new line p^*, and split the trees $\mathcal{T}_k$ at the appropriate positions. Next we obtain the new tree for each level k by glueing together the trees that correspond to pieces of the k-level, adding new vertices created by p^* where necessary. Note that for the new vertices we have to compute the continuous measure of the corresponding half-plane, but this can be done in constant time. Finally, we recompute for each set H_k the discrepancy of the two half-planes with maximum and minimum continuous measure, which are stored at the root of $\mathcal{T}_k$. The time complexity of this algorithm is dominated by the time to perform a linear number of splitting, glueing and insertion operations on the trees $\mathcal{T}_k$. Each of these operations takes $O(\log n)$ time, so the total time is $O(n \log n)$.

One final issue remains: how do we find where to split the levels? To this end we update the quad-edge structure for the arrangement $\mathcal{A}(S^*)$ to obtain the quad-edge structure of the new arrangement $\mathcal{A}(S^* \cup \{p^*\})$. This is done using basically the same method that Edelsbrunner et al. [12] use to compute an arrangement incrementally. Next we describe this method in more detail.

The idea of the method is to walk along the line p^* through the arrangement $\mathcal{A}(S^*)$, meanwhile updating its quad-edge structure. So we first determine one of the two unbounded cells, say the leftmost one, such that the slope of p^* lies in between the slopes of the two unbounded edges of this cell. This can be done in $O(n)$ time by checking all unbounded cells. The cell that we find is the cell c_0 where we start the traversal. In a generic step in the traversal we have just entered a cell c_i and we want to find the next cell c_{i+1} that is intersected by p^*. In other words, given an edge e of c_i that is intersected by p^* we want to find the other edge e' of c_i that is intersected. If we have found this edge then we can step to the other side of it (that is, to c_{i+1}) in constant time using the quad-edge data structure. To find e' we simply walk around c_i: we go from e to

the next edge of c_i in clockwise order, and so on, until we reach the edge e'. Figure 5 illustrates the process.

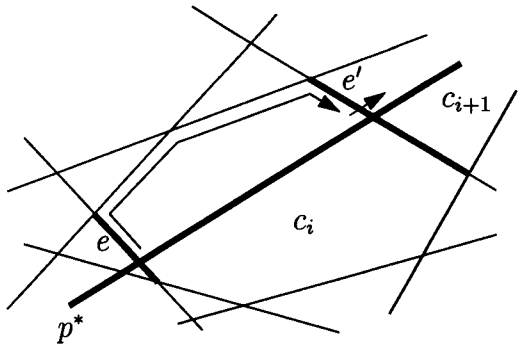

Figure 5: Traversing the arrangement.

After we have found e' we update the relevant information in the quad-edge data structure: we replace e' by two new edges, add a new vertex (the intersection of p^* with e') and a new edge (between this intersection and the intersection with e). This can be done in constant time. Then we step to the next cell c_{i+1} and we repeat the process. When we do not find a second edge e' that is intersected by p^*, we have reached the rightmost cell of the arrangement that is intersected by p^* and we are ready.

The time that we spend in cell c_i is bounded by $O(|c_i|)$, where $|c_i|$ is the number of vertices of c_i. Hence, the total time is linear in the total complexity of all the cells in $\mathcal{A}(S^*)$ that are intersected by the line p^*. This set of cells is called the *zone* of p^* in $\mathcal{A}(S^*)$, and it has been shown [12] that its total complexity is $O(n)$. It follows that we can update the quad-edge data structure in $O(n)$ time.

To find the places where to split the levels is now a matter of simple bookkeeping. We maintain cross-pointers between the vertices in the quad-edge structure and the leaves in the trees $\mathcal{T}_k$ that correspond to these vertices. If we find that we have to split an edge e of the arrangement during the traversal we know its incident vertices and, hence, the tree $\mathcal{T}_k$ that is involved. We split $\mathcal{T}_k$ at the appropriate position, and we add the point x where e is intersected as a new vertex to both subtrees. This takes $O(\log n)$ time. Note that x belongs to the set V_k, so it only influences the $h_{\max}$– and the $h_{\min}$–fields of one of the two subtrees. After we have traversed and updated the arrangement, we are left with a set of subtrees $\mathcal{T}_k^{(1)}$, $\mathcal{T}_k^{(2)}$, ... for each $k = 0, \ldots n-1$. We merge the trees $\mathcal{T}_k^{(i)}$ to obtain the new $\mathcal{T}_k$. (The order of the subtrees is easily maintained during the traversal.) Since the total number of subtrees is linear, the merging takes $O(n \log n)$ time in total.

Deletion of a point p from S can be performed in a completely symmetrical way: we follow the line p^* through the arrangement, meanwhile updating the quad-edge structure, deleting vertices on p^* from the trees $\mathcal{T}_k$, and splitting them where necessary. Finally we assemble the new trees. The following theorem summarizes the result.

Theorem 2.1 *The half-plane discrepancy of a set of points in the unit square can be maintained in $O(n \log n)$ time per insertion or deletion, using $O(n^2)$ storage, where n is the number of points in the current set.*

3 Computing strip discrepancy

In this section we give an algorithm for computing the discrepancy of a set S of n points in the unit square with respect to the family $\mathcal{S}$ of all possible strips. We shall consider closed, half-open

and open strips, that is, we consider strips where both bounding lines are part of the strip, strips where one of the lines is part of it, and strips where none of the lines is part of the strip.

For a strip s we denote the line bounding s from below by $\ell_b(s)$ and the line bounding s from above by $\ell_a(s)$. In the case of half-plane discrepancy there are only $O(n^2)$ candidates for the half-plane with the maximum discrepancy (Lemma 2.1). For strip discrepancy it turns out that there are $O(n^3)$ candidates, as follows from the following lemma.

Lemma 3.1 *Let S be a set of n points in the unit square. Suppose that s realizes the maximum discrepancy with respect to S over all strips. Then one of the following two cases occurs:*

- *$\ell_b(s)$ contains exactly one point $p \in S$ and $\ell_a(s)$ contains exactly one point of $q \in S$, and $\mu(s)$ is a local extremum of the function $\mu(\cdot)$ in the space of all strips having p on their lower boundary and q on their upper boundary.*
- *$\ell_b(s)$ contains at least two points of S and $\ell_a(s)$ contains at least one point of S, or vice versa.*

Moreover, the maximum discrepancy over all candidate strips of the former type can be computed in $O(n^2 \log n)$ time in total.

Proof: It is immediate that both bounding lines of the strip must contain a point: if a bounding line does not contain a point then it can be translated such that a strip with larger discrepancy results. By the same reasoning it follows that the continuous measure of a candidate strip s of the first type with p and q on its bounding lines must be a local extremum in the space of all strips having p and q on their bounding lines: if this is not the case then s can be 'rotated' (while keeping p and q on its bounding lines) such that the discrepancy increases. So it remains to show that the maximum discrepancy of candidate strips of the first type can be computed in $O(n^2 \log n)$ time in total.

For a point p and an angle ϕ, let $\ell_p(\phi)$ be the line through p that makes an angle ϕ with the positive x-axis. If we fix a pair $p, q \in S$, then the candidates of the first type are the strips $s_{pq}(\phi) := \ell_p(\phi)^+ \cap \ell_q(\phi)^-$ where $\mu(s_{pq}(\phi))$ is a local extremum of the function $\mu(s_{pq}(\cdot))$. (This is not really well defined, as we should distinguish between open, half-open, and closed strips. But the argument holds for all types, so we permit ourselves this slight abuse of notation.) Recall that the function $\mu(\ell_p^+(\phi))$ consists of four pieces of the form given by Equation (1). Hence, $\mu(s_{pq}(\phi))$ has $O(1)$ local extrema, which can be computed in $O(1)$ time. We are left with the problem of computing the discrete measure $\mu_S(s_{pq}(\phi))$ of all the $O(n^2)$ candidate strips. We briefly sketch how this can be done in $O(n^2 \log n)$ time. Observe that $\mu_S(s_{pq}(\phi))$ can be computed in constant time if we know the number of points in S above $\ell_p(\phi)$ and the number of points above $\ell_q(\phi)$. Thus we are given a set S of n points and a set L of $O(n^2)$ lines through these points, and we want to compute for each line in L the number of points in S above it. In dual space (see the previous section for details) this corresponds to the following problem: given a set S^* of n lines and a set L^* of $O(n^2)$ points lying on these lines, compute for each point in L^* the number of lines in S^* below it. To do this we construct the arrangement $\mathcal{A}(S^*)$ in $O(n^2)$ time, and we sort the points lying on each of the lines in $O(n^2 \log n)$ time in total. Finally, we walk from left to right along each of the lines to compute the level of the encountered points. □

In the remainder of this section we concentrate on the computation of the maximum discrepancy of the candidate strips of the second type. In general there are $\Theta(n^3)$ such candidates, three for each triple of points. Nevertheless, it turns out to be possible to avoid looking at all candidates, and solve the problem in close to quadratic time.

We denote the line through two points p and q by ℓ_{pq}, and we denote the angle that ℓ_{pq} makes with the positive x-axis by ϕ_{pq}. Consider a fixed pair of points p, q. The strips that we must consider are the strips[2] $s_{pq,r} := \ell_{pq}^+ \cap \ell_r(\phi_{pq})^-$ or $s_{pq,r} := \ell_{pq}^- \cap \ell_r(\phi_{pq})^+$, depending on whether

[2] Again, this is not well defined since we should consider four versions of each strip, depending on which of the bounding lines are part of the strip and which not. To simplify the notation we simply write $s_{pq,r}$ as a generic name for all four strips, making the distinction only where necessary.

the point r lies above or below ℓ_{pq}. Our strategy will be to preprocess the set S such that for a given pair p, q we can quickly find the point r that maximizes $\Delta_S(s_{pq,r})$.

For an open half-plane ℓ^+ we define the *signed discrepancy* $\delta(\ell^+)$ as

$$\delta(\ell^+) \;:=\; \mu(\ell^+) \;-\; \mu_S(\ell^+).$$

For closed half-planes we write $\overline{\delta}(\ell^+)$. These quantities are essentially the discrepancy of the half-plane, except that we do not take absolute values. We have the following lemma.

Lemma 3.2 *Let p, q be a fixed pair of points in S. The four strips (one open, one closed and two half-open strips) defined by p, q and a point $r \in S$ have discrepancies*

$$\begin{aligned}
&|\,\overline{\delta}(\ell^+_{pq}) - \delta(\ell^+_r(\phi_{pq}))\,|,\\
&|\,\delta(\ell^+_{pq}) - \overline{\delta}(\ell^+_r(\phi_{pq}))\,|,\\
&|\,\delta(\ell^+_{pq}) - \delta(\ell^+_r(\phi_{pq}))\,|, \qquad \text{and}\\
&|\,\overline{\delta}(\ell^+_{pq}) - \overline{\delta}(\ell^+_r(\phi_{pq}))\,|.
\end{aligned}$$

Furthermore, any point r that defines a strip $s_{pq,r}$ of maximum discrepancy must be a point that maximizes or minimizes either $\delta(\ell^+_r(\phi_{pq}))$ or $\overline{\delta}(\ell^+_r(\phi_{pq}))$ over all points $r \in S$.

Proof: Consider a strip $s_{pq,r}$ such that point r that lies below ℓ_{pq}. Assume that the strip is open on both sides. We then have

$$\begin{aligned}
\Delta_S(s_{pq,r}) &= |\mu(s_{pq,r}) - \mu_S(s_{pq,r})|\\
&= |\{\mu(\ell^+_r(\phi_{pq})) - \mu(\ell^+_{pq})\} - \{\mu_S(\ell^+_r(\phi_{pq})) - \mu_S(\ell^+_{pq})\}|,\\
&\quad \text{where } \ell^+_r(\phi_{pq}) \text{ is open and } \ell^+_{pq} \text{ is closed}\\
&= |\delta(\ell^+_r(\phi_{pq})) - \overline{\delta}(\ell^+_{pq})|.
\end{aligned}$$

The other expressions arise in the same way for the other types of strips (half-open and closed), as is easily verified. We get exactly the same four expressions when r lies above ℓ_{pq}. It is important that not only every strip discrepancy can be expressed as one of the above formulas, but that every formula also corresponds to a valid strip. Consequently, the point r that maximizes $\Delta_S(s_{pq,r})$ must be a point that maximizes or minimizes either $\delta(\ell^+_r(\phi_{pq}))$ or $\overline{\delta}(\ell^+_r(\phi_{pq}))$. □

To compute the discrepancy with respect to the strips $s_{pq,r}$ of the second type we proceed as follows. First, we compute the values $\delta(\ell^+_{pq})$ and $\overline{\delta}(\ell^+_{pq})$ for each pair p, q. Then we preprocess the two sets of functions $\{\delta(\ell^+_r(\phi)) : r \in S\}$ and $\{\overline{\delta}(\ell^+_r(\phi)) : r \in S\}$ such that for any given angle ϕ we can quickly find the points that maximize and minimize $\delta(\ell^+_r(\phi))$ and $\overline{\delta}(\ell^+_r(\phi))$. Finally, for each pair p, q we query with the angle ϕ_{pq} in this data structure. According to the lemma above we can now determine the maximum discrepancy with respect to the set of strips $\{s_{pq,r} : r \in S\}$. The next two lemmas describe the two ingredients that we need in more detail.

Lemma 3.3 *The values $\delta(\ell^+_{pq})$ and $\overline{\delta}(\ell^+_{pq})$ for each pair p, q can be computed in $O(n^2)$ time in total.*

Proof: This is essentially the same problem as the computation of the half-plane discrepancy of S. (The difference is that we are only interested in half-planes whose bounding line contains two points, and that we do not take the absolute value.) Hence, the problem can be solved using dualization, as in the previous section. □

Lemma 3.4 *The set S can be preprocessed in $O(n^2 2^{\alpha(n)} \log n)$ time into a data structure of size $O(n^2 2^{\alpha(n)})$ such that, given a query angle ϕ, one can compute the maximum and the minimum values of $\delta(\ell^+_r(\phi))$ and $\overline{\delta}(\ell^+_r(\phi))$ over all points $r \in S$ in $O(\log n)$ time.*

Proof: We discuss the structure for finding the maximum value of any of the functions $\delta(\ell_r^+(\phi))$, $r \in S$, at a query value of ϕ. Finding the minimum value, and finding the maximum and minimum for the functions $\overline{\delta}(\ell_r^+(\phi))$, can be done in a similar way. So we are interested in the function $\delta(\phi) := \max_{r \in S} \delta(\ell_r^+(\phi))$. The function $\delta(\phi)$ is called the *upper envelope* of the functions $\delta(\ell_r^+(\phi))$. It is known that the number of breakpoints on the upper envelope of a set of N continuous functions such that any pair intersects at most k times is $O(\lambda_{k+2}(N))$, where $\lambda_{k+2}(N)$ is the maximum length of an $(N, k+2)$ Davenport-Schinzel sequence. For any constant k, $\lambda_{k+2}(N)$ is nearly linear in N [1].

Now let's study the function $\delta(\ell_r^+(\phi))$ for a fixed point $r \in S$ more closely. By definition we have $\delta(\ell_r^+(\phi)) = \mu(\ell_r^+(\phi)) - \mu_S(\ell_r^+(\phi))$. The function $\mu(\ell_r^+(\phi))$ is a continuous function. The function $\mu_S(\ell_r^+(\phi))$, on the other hand, is a step-function: it changes value at the angles ϕ where the line $\ell_r(\phi)$ crosses a point of S, and in between two such values it is constant. So splitting $\delta(\ell_r^+(\phi))$ at these special angles gives us n continuous pieces. We also split $\delta(\ell_r^+(\phi))$ at the angles where $\ell_r(\phi)$ contains one of the corners of the unit square. We now get $n+4$ pieces. The discrete measure is constant for each piece, and the two sides of the unit square intersected by the lines corresponding to a piece are also fixed. It follows from Equation (1) that each piece is of the form

$$b_1\phi + b_2 + b_3/\phi,$$

where the constants b_1, b_2, b_3 depend on the coordinates of the point r, on which two sides of the unit square are intersected, and on the discrete measure. Hence, such a piece intersects a piece of any other point $r' \in S$ at most twice. It follows that the number of breakpoints on $\delta(\phi)$ is $O(\lambda_4(4n^2)) = O(n^2 2^{\alpha(n)})$, where $\alpha(n)$ is the extremely slowing growing functional inverse of Ackermann's function [1]. Moreover, $\delta(\phi)$ can be computed in $O(n^2 2^{\alpha(n)} \log n)$ time with a divide-and-conquer algorithm. Once we have computed $\delta(\phi)$ we can find the maximum value of the functions $\delta(\ell_r^+(\phi))$ for a given angle ϕ in $O(\log n)$ time by binary search. □

Putting it all together we obtain the following theorem.

Theorem 3.1 *The strip discrepancy of a set of n points in the unit square can be computed in $O(n^2 2^{\alpha(n)} \log n)$ time using $O(n^2 2^{\alpha(n)})$ storage.*

4 Concluding Remarks

We have presented new algorithms for computing the discrepancy of a set of points in the unit square. The first algorithm is a dynamic one, for maintaining the half-plane discrepancy. It runs in $O(n \log n)$ time per update and it uses only fairly standard data structures. The second algorithm computes the strip discrepancy in $O(n^2 2^{\alpha(n)} \log n)$ time.

It would be interesting to find efficient algorithms for other types of discrepancy, such as triangle discrepancy. Half-plane discrepancy essentially estimates the quality of a sample set with respect to polygons that are relatively large compared to a pixel, such that only one edge of the polygon crosses the pixel; triangle discrepancy would give a better estimate for the quality with respect to small (sub-pixel size) polygons. However, the combinatorial complexity (the number of candidate triangles) is very large, namely $\Theta(n^6)$. But perhaps one can avoid to consider all candidates, as in the case of strip discrepancy. Another possibility is to try and find approximation algorithms for triangle discrepancy, in the same spirit as Chazelle's work on half-plane discrepancy [4].

References

[1] P. K. Agarwal, M. Sharir, and P. Shor. Sharp upper and lower bounds on the length of general Davenport-Schinzel sequences. *J. Combin. Theory Ser. A*, 52:228–274, 1989.

[2] A. V. Aho, J. E. Hopcroft, and J. D. Ullman. *The Design and Analysis of Computer Algorithms.* Addison-Wesley, Reading, MA, 1974.

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

[4] B. Chazelle. Geometric Discrepancy Revisited. In *Proc. 34th Annu. IEEE Sympos. Found. Comput. Sci.*, pages 392–399. 1993.

[5] M.F. Cohen and J.R. Wallace. *Radiosity and Realistic Image Synthesis*, Academic Press Professional, 1993.

[6] R.L. Cook, Stochastic sampling in computer graphics. *ACM Trans. Graphics*, 5:51–72, 1986.

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

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

[9] D. Dobkin and D. Eppstein. Computing the discrepancy. In *Proc. 9th Annu. ACM Sympos. Comput. Geom.*, pages 47–52. 1993.

[10] D. Dobkin and D. Mitchell, Random-edge discrepancy of supersampling patterns. In *Graphics Interface '93*, York, Ontario, May, 1993.

[11] H. Edelsbrunner. *Algorithms in Combinatorial Geometry.* Springer-Verlag, Heidelberg, West Germany, 1987.

[12] H. Edelsbrunner, J. O'Rourke, and R. Seidel. Constructing arrangements of lines and hyperplanes with applications. *SIAM J. Comput.*, 15:341–363, 1986.

[13] L. J. Guibas and R. Sedgewick. A diochromatic framework for balanced trees. In *Proc. 19th Annu. IEEE Sympos. Found. Comput. Sci.*, pages 8–21. 1978.

[14] L. J. Guibas and J. Stolfi. Primitives for the manipulation of general subdivisions and the computation of Voronoi diagrams. *ACM Trans. Graph.*, 4:74–123, 1985.

[15] J.T. Kajiya. The rendering equation. *Computer Graphics*, 20:143–150, 1986.

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

[17] M. H. Overmars and J. van Leeuwen. Maintenance of configurations in the plane. *J. Comput. Syst. Sci.*, 23:166–204, 1981.

[18] P. Shirley. Discrepancy as a quality measure for sample distributions. In *Proc. Eurographics '91*, pages 183–193, 1991.