TUFTS-CS Technical Report 2003-04

September, 2003

Computational Geometry and Statistical Depth Measures[1]

by

Eynat Rafalin,[2]
Dept. of Computer Science
Tufts University
Medford, Massachusetts 02155
and Diane L. Souvaine[1]
Dept. of Computer Science
Tufts University
Medford, Massachusetts 02155

[1]Also appeared in *Theory and Applications of Recent Robust Methods*, edited by M. Hubert, G. Pison, A. Struyf and S. Van Aelst, 2004 in *Series: Statistics for Industry and Technology*, Birkhauser, Basel
[2]Partially supported by NSF grant EIA-99-96237

ABSTRACT

The computational geometry community has long recognized that there are many important and challenging problems that lie at the interface of geometry and statistics (e.g. [39, 5]). The relatively new notion of *data depth* for non-parametric multivariate data analysis is inherently geometric in nature, and therefore provides a fertile ground for expanded collaboration between the two communities. New developments and increased emphasis in the area of multivariate analysis heighten the need for new and efficient computational tools and for an enhanced partnership between statisticians and computational geometers.

Over a decade ago point-line duality and combinatorial and computational results on arrangements of lines contributed to the development of an efficient algorithm for two-dimensional computation of the LMS regression line [42, 17]. The same principles and refinements of them are being used today for more efficient computation of data depth measures. These principles will be reviewed and their application to statistical problems such as the LMS regression line and the computation of the halfspace depth contours will be presented. In addition, results of collaborations between computational geometers and statisticians on data-depth measures (such as halfspace depth and simplicial depth) will be surveyed.

1 Introduction

The field of *Computational Geometry* deals with the systematic study of algorithms and data structures for geometric objects [12]. Computational geometry usually focuses at the outset on exact algorithms that are asymptotically fast. Yet once exact algorithms have been obtained, refined, and are still slow, approximation algorithms of provable performance are sought. The field emerged in the 1970's with the work of Michael Shamos [40]. Even in these early days the computational geometry community has recognized that there are many important and challenging problems that lie at the interface of geometry and statistics (e.g. [39, 5]).

The field is commonly related to problems in *robotics, CAD/CAM* and *geographic information systems*. However, any problem that can be represented using geometric objects and operators can be viewed as a computational geometry problem, including the relatively new notion of *data depth* for non-parametric multivariate data analysis. Data depth is inherently geometric in nature, and therefore provides a fertile ground for expanded collaboration between the two communities.

A *data depth* measures how deep (or central) a given point $x \in \mathbb{R}^d$ is relative to F, a probability distribution in $\mathbb{R}^d$ or relative to a given data cloud. Some examples of data depth functions are *Halfspace Depth* [19, 44], *Majority Depth* [41], *Simplicial Depth* [25], *Oja Depth* [30], *Convex Hull Peeling Depth* [3, 14] and *Regression Depth* [34]. The data depth concept provides *center-outward* orderings of points in Euclidean space of any dimension and leads to a new *non-parametric* multivariate statistical analysis in which *no* distributional assumptions are needed. Most depth functions are defined in respect to a probability distribution F, considering $\{X_1, .., X_n\}$ random observations from F. The *finite sample version* of the depth function is obtained by replacing F by F_n, the empirical distribution of the sample $\{X_1, .., X_n\}$. In general, computational geometers study the finite sample case in which sets of points are investigated.

Over a decade ago, point-line duality and combinatorial and computational results on arrangements of lines contributed to the development of efficient algorithms for two-dimensional computation of the LMS regression line [42, 17]. The same principles and refinements of them are being used today for more efficient computation of data depth measures.

The technique of point-line duality, its application to LMS, and various sweep techniques will be sketched in Section 2. More recent results for data-depth related problems will be reviewed in Section 3 (including halfspace depth and simplicial depth). Sections 4 and 5 will include suggestions for future collaboration and a summary.

2 History

2.1 The Duality Transform [6, 13]

The structure of a collection of lines is more readily observed than the structure implied by a set of points. This structure can be exploited to create efficient algorithms. A set of points

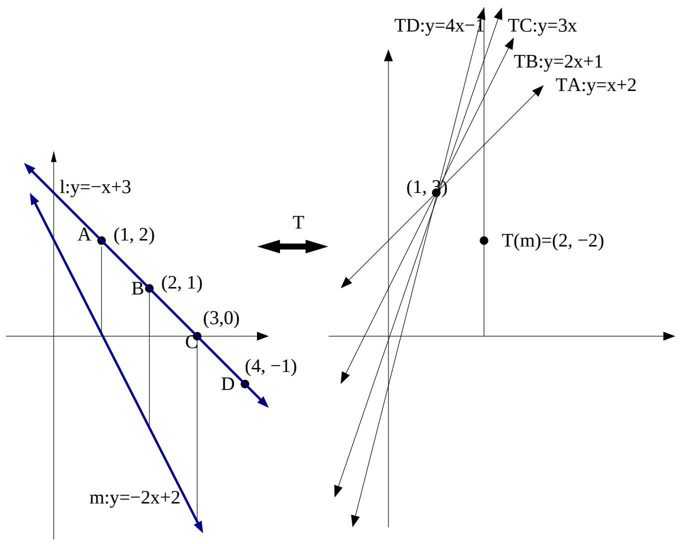

Figure 1: The duality transform T: The set of points and lines on the left is transformed into the set on the right. The points $A = (1, 2), B = (2, 1), C = (3, 0)$ and $D = (4, -1)$ lying on the line $l : y = x + 3$ are mapped to the lines $T(A) : y = x + 2$, $T(B) : y = 2x + 1$, $T(C) : y = 3x$ and $T(D) : y = 4x - 1$, all passing through the point $(1, 3)$, dual to the line l. The points A, B, C, D are located above $m : y = 2x + 2$, as are their dual lines, above point $T(m) : (2, -2)$. In addition, the vertical distance between every point A, B, C, D to the line m is identical to the vertical distance between their dual lines and the dual point $T(m)$.

can be transformed into an arrangement of lines using a duality transform that preserves key properties [6]. Many problems based on sets of points have been solved efficiently by first transforming the points into a set of lines, solving a related problem on the set of lines, and converting the answer to a solution for the original problem. The transformation itself is conducted in linear time, and consequently does not affect the total computational complexity of the solution. Different transformations apply to different problems. Here, we will focus on a particular transform suitable for the statistical problems we treat. For more details see [13].

The *duality transform* T (see Figure 1) maps a point $P = (a, b)$ to a line $T(P) : y = ax+b$. We need to define the image of a line $l : y = cx + d$ under transform T consistently. Let us pick points $Q = (q, cq + d)$, $R = (r, cr + d)$ lying on line l. $T(Q) : y = qx + (cq + d)$ and $T(R) : y = rx + (cr + d)$. Both of these lines pass through the point $(-c, d)$. Every other point on l will also be mapped to a line passing through $(-c, d)$. Consequently we say that $T(l) : (-c, d)$.

Note that the slope of l is preserved in the x-coordinate of the image point. If the slope of line l exceeds the slope of line k, then $T(l)$ lies to the left of $T(k)$. The vertical distance of a point $P = (a, b)$ to a line $l : y = cx + d$ equals $b - (ca + d)$. In the dual, the vertical distance from the line $T(p) : y = ax + b$ to the point $T(l) = (-c, d)$ is $(-ac + b) - d$, the same value. Hence, the transformation T preserves slope, vertical distance and the above/below relationship.

For a vertical line $l : x = m$, T maps it to a pencil of parallel lines of slope m. We assume that these parallel line intersect at a point ∞_m (this point is called an *improper point*). For every direction m, there is an improper point ∞_m, associated with it. Conversely, each line $y = mx + a$ which passes through ∞_m gets mapped to a point $(-m, a)$ which lies on the line $x = -m$. Consequently, the image of the improper point ∞_m under T is the vertical line $x = -m$.

The duality transform T can be extended to higher dimensions: in $\mathbb{R}^3$ the dual of the point $P = (a, b, c)$ is the plane $z = ax + by + c$.

2.2 Least Median of Squares Regression

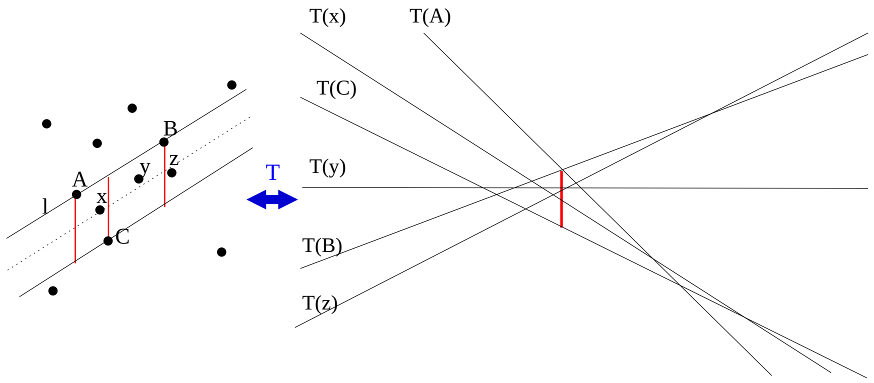

Figure 2: LMS regression: A set of points and its dual arrangement (only some of the lines in the arrangement are drawn), and the LMS slab and regression line. Line l passes through points A and B, and its dual point $T(l)$ is the intersection of lines $T(A)$ and $T(B)$. The slab created from line l and the line parallel to it passing through point C contains 6 points. In the dual, the vertical lines between $T(l)$ and the line $T(C)$ crosses 6 lines.

Consider the problem of fitting a line to a set of data points. The familiar *ordinary least squares (OLS)* method minimizes the sum of the squares of the y-distance between the fitted line and the data points (the *residual*). This method has the disadvantage that a single corrupt point (*outlier*) can significantly perturb the fitted line. The *Least Median of Squares (LMS) Regression line* [33] is the line that minimizes the median of the squares of the residuals. This method has a high breakdown point as up to 50% of the data points can be outliers, without perturbing the fitted line.

It is easy to prove that the problem of finding the line l that minimizes the median residual is equivalent to the problem of finding the slab bounded by a pair of parallel lines of minimum vertical separation that contain $\lceil \frac{n}{2} \rceil$ of the data points. Furthermore, one bounding line L_1 must pass through two points and the other line L_2 through a point whose x-coordinates lies between those of L_1's points. To find the slab, one can check each pair of points (p, q) and find a line l parallel to them, passing through some other point r, such that precisely $\lceil \frac{n}{2} \rceil$

points lie in the closed slab defined by $\overline{pq}$ and l. This can be solved naively using an $O(n^3)$ time algorithm.

Mapping the set of points using the transform T, the above problem becomes one of looking for the intersection point of two lines $T(p)$ and $T(q)$ such that the y-distance between this point and the line $T(r)$, which is $\lceil \frac{n}{2} \rceil$ above or below, is the shortest among all such distances. By sweeping (see Section 2.3) a line through the arrangement dual to the set of points, one can check all intersection points and lines such that the closed vertical segment from the point to the line intersects $\lceil \frac{n}{2} \rceil$ lines and is as short as possible.

Souvaine and Steele [42] proposed an $O(n^2 \log n)$ time algorithm for computing the LMS line in $\mathbb{R}^2$ using the duality concept described above. Their result was improved by Edelsbrunner and Souvaine [17] to an $O(n^2)$ time algorithm, by using a topological line sweep (see Section 2.3). A practical approximation algorithm for 2 and higher dimensions, using additional computational geometry concepts, was recently presented by Mount *et al.* [28].

2.3 Line Sweep, Topological Line Sweep and Levels in an Arrangement

Vertical line sweep [4] is a classical technique in computational geometry. The algorithm sweeps a vertical line across an arrangement of objects (points, lines, segments, ..) from left to right, reporting all intersection points, in a series of elementary steps. For an arrangement of n line segments consisting of k intersections points this technique achieves a time complexity of $O((n+k)\log n)$ and requires $O(n)$ space [7]. For an arrangement of n infinite lines that contains $k = \frac{n(n-1)}{2} = O(n^2)$ intersections, vertical line sweep could report all intersection pairs sorted in order of x-coordinate in $\Theta(n^2 \log n)$ time and $O(n)$ space. If one needs to report the intersection points of the lines according only to a partial order related to the *levels* in the arrangement greater efficiency is possible using *topological sweep* [16]. In topological line sweep, to report all the intersection points of the lines, a topological line, which is monotonic in the y-direction but not necessarily straight, and which intersects each of the n lines in the arrangement exactly once, sweeps the arrangement, in a series of elementary steps, in $O(n^2)$ time and $O(n)$ space.

The *kth level* of an arrangement of lines is the set of points that lie on lines and have at most $k-1$ lines above them and at most $n-k$ lines below them (see Figure 3). Consider an arrangement that was created from a set of points, using the duality transform T. If a point $T(p)$, the image of line p, that lies on line $l = T(L)$, is in the kth level, then line p has at most $k-1$ points above it and at most $n-k$ points below it. This property will be used in Section 3.1.1, to compute halfspace depth contours.

3 Current Applications

3.1 Halfspace Depth and Halfspace Depth Contours

The *halfspace depth*[1] of a point x relative to a set of points $\mathcal{S} = \{X_1, \ldots, X_n\}$ is the minimum number of points of $\mathcal{S}$ lying in any closed halfspace determined by a line through x [19, 44].

Using the duality transform T, the set of point $\mathcal{S}$ maps to an arrangement of lines $T(\mathcal{S})$. The number k of points of $\mathcal{S}$ above (resp. below) a line L through x, equals the number of lines of $T(\mathcal{S})$ above (resp. below) the dual point $T(L)$. The depth of a point x relative to $\mathcal{S}$ is the minimum number of points of $\mathcal{S}$ lying in any closed halfspace determined by a line through x (above or below the line through x). Consequently, the depth of a line $T(x)$, dual to the point x, relative to $T(\mathcal{S})$, is the minimal number of lines of $T(\mathcal{S})$ above or below $T(x)$.

3.1.1 Efficient computation of Depth Contours using Duality

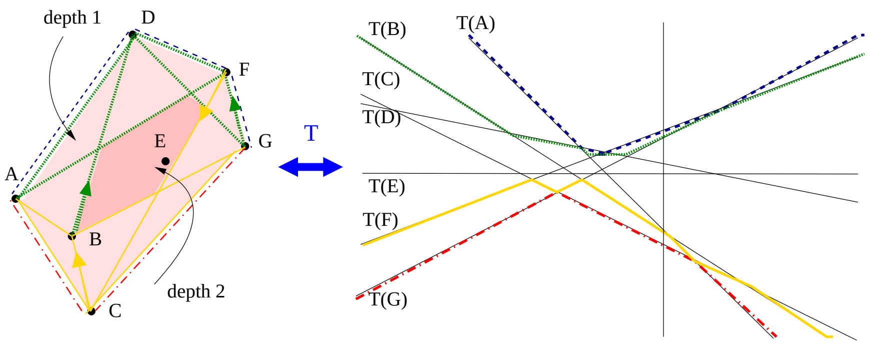

Figure 3: Levels in an arrangement and halfspace depth computation: The set of 7 points $A - G$ on the left is transformed using the duality transform T into the arrangement of lines $T(A) - T(G)$ on the right. The levels $1, 2, 6, 7$ are drawn in blue, green, yellow and red, respectively. Any vertical line cutting the kth level will pass through exactly $k - 1$ lines of the arrangement above it. The intersection points on the 1st and 7th level are candidate points for the first halfspace depth contour (as are the 2nd and 6th levels for the second depth contour). Therefore transforming the intersection points on each level back to the primal plane (as lines) and computing their intersection will result in that depth contour. The 1st and 2nd contours are drawn as grayed areas.

The kth *depth contour* for a set of points $\mathcal{S}$ in $\mathbb{R}^d$ is the boundary of the points of $\mathbb{R}^d$ with depth $\geq k$, according to the chosen depth function.

The kth depth contours for halfspace depth in $\mathbb{R}^2$ can be constructed by taking the intersection of all halfplanes containing k points of $\mathcal{S}$. Specifically, the significant halfplanes are the ones bounded by lines connecting pairs of points of $\mathcal{S}$. Every line L passing through a pair of points l, m is dualized to an intersection point I of two lines $T(l), T(m)$. The vertical

[1]In the literature this is sometimes called *location depth* or *Tukey depth*.

line through I in the dual intersects every line of the arrangement, some above I and some below. The number of lines intersected above (resp. below) I equals the number of points lying above (resp. below) $T^{-1}(I) = L$ in the primal. The contour to which L contributes is the minimum of these two numbers (this is also said to be the *level* of L, see Section 2.3). Examining all intersection points in the arrangement of dual lines will produce a list of halfplanes/intersections for each potential contour.

Miller *et al.* [27] presented an $O(n^2)$ time algorithm for computing all depth contours for a planar data set, using the idea sketched above. The authors used *topological sweep* (see Section 2.3) to examine all intersection points and to report the level of each intersection point in optimal time and space.

Note that the duality concept and the algorithmic idea can be easily extended to higher dimensions. For example, in $\mathbb{R}^3$ the halfspace depth of a point x is the minimum number of points of a given set $\mathcal{S}$ lying in any closed halfspace bounded by a hyperplane (instead of a line, as in $\mathbb{R}^2$) through x.

3.1.2 Overview of Related Results

The halfspace depth of a single point can be computed in $O(n \log n)$ time as shown by Rousseeuw and Ruts [38]. This matches the lower bound that was recently proved by Aloupis *et al.* [1].

Algorithms for computing the deepest point relative to the data set (also known as the *Tukey median*) were introduced by Rousseeuw and Ruts. In [38], they give an $O(n^2 \log n)$ time implementation to compute a single contour. In [36], the same technique is applied repeatedly to give an $O(n^2 \log^2 n)$ time implementation to compute the two-dimensional Tukey median. A fast implementation to approximate the deepest location in high dimensions is given in [37, 43, 45]. Theoretical complexity results on two-dimensional *depth contours*, or k-hulls, have been known for some time [11]. The best known theoretical result for computing the $2D$ Tukey median is an $O(n \log^5 n)$ algorithm by Matoušek [26], but its complex structure makes it an impractical candidate for implementation. The improved $O(n \log^4 n)$ algorithm by Langerman and Steiger [24] uses parametric search, also difficult to implement.

Rousseeuw and Ruts developed a program called ISODEPTH [38], that computes the kth contour in $O(n^2 \log n)$ time and all contours in $O(n^3 \log n)$. Johnson, Kwok and Ng gave a program called FDC to compute the k outermost depth contours [21] which outperforms ISODEPTH for small ks. The $\Theta(n^2)$ time implementation by Miller *et al.* that was described above computes all the depth contours, the depth of all the data points, and the Tukey median [27]. The implementation was expanded by Rafalin *et al.* [31] to handle degenerate data sets, that contain 3 or more points on a line or points that share the same x-coordinate. A different approach based on parallel arrangement construction by Fukuda and Rosta [32] allows to compute high dimensional depth contours. In addition, halfspace depth contours can be computed for display in 2D using hardware assisted computation, as suggested by Krishnan *et al.* [23].

Centerpoints, which are points of depth $\geq n/(d+1)$ (by Helly's theorem [15], such points are guaranteed to exist), have been widely studied in the computational geometry literature [10, 20, 26, 29].There are efficient algorithms for centerpoints in $\mathbb{R}^2$ by Matoušek [26] and in linear time by Jadhav and Mukhopadhyay [20]. Naor and Sharir [29] give an algorithm in $\mathbb{R}^3$. The best known time in high dimensions is $O(n^{d+1})$, but a faster approximation algorithm is given by Clarkson *et al.* [10]. The algorithms are difficult to implement.

3.2 Simplicial Depth

Simplicial depth was introduced by Liu in 1990 [25]: The *simplicial depth* of a point x with respect to a data set $\mathcal{S} = \{X_1, \ldots, X_n\}$ is the fraction of the closed simplices formed by $d+1$ points of $\mathcal{S}$ containing x, where I is the indicator function: $SD_{Liu}(\mathcal{S}; x) = \binom{n}{d+1}^{-1} \sum I_{(x \in S[X_{i_1}, \ldots X_{i_{d+1}}])}$ (see Figure 4(a)).

A simplex in $\mathbb{R}^d$ is the boundary of the region defined by $d+1$ vertices in general position, and represents the simplest possible polytope in any dimension. In one dimension, a simplex is simply a line segment; in 2 dimensions, it is a triangle; and in 3 dimensions, a tetrahedron. Simplicial depth has been studied by computational geometers since it was first introduced [22, 18]. The simplicial depth is computationally more difficult in the finite sample case than some other depth measures. Recently new results and concepts were discovered by computational geometers, some discussed below.

3.2.1 A Revised Definition

Several problems arise in the finite sample case of simplicial depth under Liu's definition. As suggested by Zuo and Serfling [46], a depth function should attain maximum value at the center (*the maximality property*) and as a point x moves away from the *deepest point* along any fixed ray through the center, the depth at x should decrease monotonically (*monotonicity property*). However, the simplicial depth function for the finite sample case fails to satisfy these properties [46] (see e.g. Figure 4(b)). In addition, the depth of points on facets causes discontinuities in the depth function: The depth of all points on the boundary of a cell is at least the depth of a point on the interior of the cell. In most cases the depth values on the boundaries can be higher than the depth in each of the adjacent cells (see e.g. Figure 4(b)).

Burr *et al.* [8] proposed a modification to the definition that corrects the irregularity at boundaries of simplicies by making the depth of a point on the boundary between two cells the *average* of the depth of the two cells and fixes the counterexamples provided by Zuo and Serfling [46]:
Given a data set $\mathcal{S} = \{X_1, \ldots, X_n\}$ in $\mathbb{R}^d$, the *simplicial depth* of a point x is the average of the fraction of closed simplicies containing x and the fraction of open simplicies containing x:

$$SD_{BRS}(\mathcal{S}; x) = \frac{1}{2}\binom{n}{d+1}^{-1} \left(\sum I_{(x \in S[X_{i_1}, \ldots X_{i_{d+1}}])} + I_{(x \in int(S[X_{i_1}, \ldots X_{i_{d+1}}]))}\right)$$

where *int* refers to the open relative interior[2] of $S[X_{i_1}, \ldots, X_{i_{d+1}}]$. Equivalently, this could be

[2]See Edelsbrunner [15], page 401.

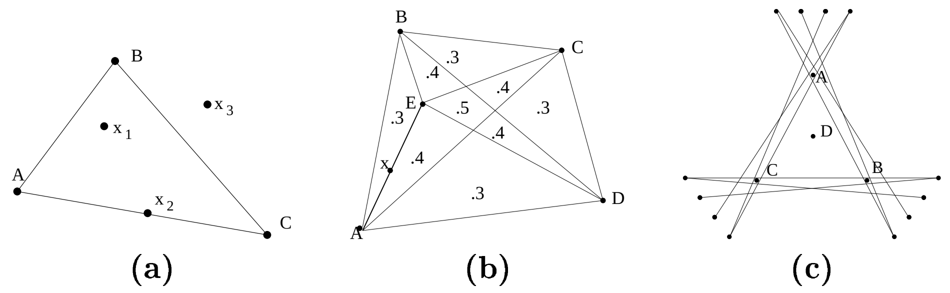

Figure 4: (a) Computation of simplicial Depth, according to Liu and according to Burr *et al.*. $SD_{Liu}(x_1) = SD_{BRS}(x_1) = 1$, $SD_{Liu}(x_2) = 1$, $SD_{BRS}(x_2) = .5$. $SD_{Liu}(x_3) = SD_{BRS}(x_3) = 0$. (b) Problems with simplicial depth: The total number of simplicies is $\binom{5}{3} = 10$. The depth of the open regions is drawn on the figure. $SD_{Liu}(p) = \binom{4}{2}/10 = .6$, for $p = \{A, B, C, D\}$, while $SD_{Liu}(E) = .8$. For a point x on $\overline{AE}$ $SD_{Liu}(x) = .5$, violating the monotonicity property and causing discontinuities in the depth function at edges. According to the revised definition $SD_{BRS}(p) = .3$, for $p = \{A, B, C, D\}$, $SD_{BRS}(E) = .5$ and $SD_{BRS}(x) = .35$. (c) A problem with the revised definition. The data points A,B, and C all have depth $(241 + \frac{1}{2}\binom{15}{2})/\binom{16}{3} = \frac{587}{1120}$ and the data point D, which is at the unique center of the data set has depth $(5^3 + \frac{1}{2}\binom{15}{2})/\binom{16}{3} = \frac{355}{1120}$. For clarity reasons not all cells are drawn.

formulated as: $SD_{BRS}(\mathcal{S}; x) = \rho(\mathcal{S}, x) + \frac{1}{2}\sigma(\mathcal{S}, x)$ where $\rho(\mathcal{S}, x)$ is the number of simplicies with data points as vertices which contain x in their open interior, and $\sigma(\mathcal{S}, x)$ is the number of simplicies with data points as vertices which contain x in their boundary.

The revised definition reduces to the original definition for continuous distributions and for points lying in the interior of cells, and it maintains the ranking order of data points. In addition, it can be calculated using the existing algorithms (see Section 3.2.2), with slight modifications. However, it does not achieve all desired properties in the sample case. Figure 4(c) shows an example where the data set has a unique center, D, but it neither attains maximality at the center, nor does it have monotonicity relative to the deepest point.

In addition, data points are still over-counted. For example data points in $\mathbb{R}^2$ are a vertex of $\binom{n-1}{2}$ simplicies, whereas edges are counted only $n-2$ times. This implies that the weight λ of a data point should be $\frac{1}{2}(n-2) = \lambda\binom{n-1}{2} \Rightarrow \lambda = \frac{1}{n-1}$. However, this factor isn't enough: consider a data set of n points, where $n-1$ points are evenly distributed angularly around one point. Then the depth of the center point should be at least as large as the depth of the n cells which use it as a vertex; this is not guaranteed by the λ factor. Thus the depth of a data point in $\mathbb{R}^d$ should be based both on the λ factor and the geometry of the $n-1$ other data points.

3.2.2 Overview of Related Results

The simplicial depth of a point in $\mathbb{R}^2$ can be computed in $O(n \log n)$ time, as proposed independently by Gil *et al.* [18], Khuller and Mitchell [22], and Rousseeuw and Ruts [35]. This bound matches the lower bound, as proved by Gil *et al.* [18] and Aloupis *et al.* [1].

In 3 dimensions, Gil *et al.* gave an $O(n^2)$ algorithm for computing the simplicial depth of x relative to $\mathcal{S}$ [18] that was recently improved by Cheng and Ouyang [9]. Cheng and Ouyang also give a generalization of this algorithm to $\mathbb{R}^4$, with a time complexity of $O(n^4)$. Rousseeuw and Ruts proposed an $O(n^2)$ algorithm for $\mathbb{R}^3$ [35], but some missing details may not be resolvable [9]. For space higher than 4 dimensional, there are no known algorithms faster than the straightforward $O(n^{d+1})$ method (generate all simplices and count the number of containments). The depth of all n points in $\mathbb{R}^2$ can be computed in $O(n^2)$ using the duality transform [18, 22]. The *simplicial median* is the point with the highest simplicial depth. Aloupis *et al* [2] showed that in 2 dimensions it can be computed in $O(n^4)$. Their method was slightly improved by Burr *et al.* [8].

No known algorithm exists for the computation of simplicial depth contours, apart from the straightforward one. Recently an approximation method, using local information about the depth function (using a discretized version of the gradient, the vector where simplicial depth of positions is increasing most rapidly) was proposed by Burr *et al.* [8].

4 Future Work

Collaborations between computational geometers and statisticians on data-depth measures have produced more efficient algorithms and implementations with significant speedup. Some problems are solved in theory by algorithms with efficient asymptotic running times (see Sections 3.1.2, 3.2.2) where, in practice, the hidden constants make implementations infeasible, prompting the need for continued research and implementable solutions. In addition, most of the algorithms and their implementations work for the low dimensional case. High dimensional data sets pose more interesting research questions. Exact computation of the depth functions will probably never be efficient enough, as the complexity of the algorithm is in most cases exponential in the dimension of the data. This calls for more approximation algorithms, that can compute depth functions and depth contours efficiently, with provable error bounds.

5 Summary

The computational geometry community is thirsty for new and challenging open problems with real applications; the statistical community clearly needs improved computation speed in order to handle the large data sets of today. Much is to be gained by increased collaboration and solutions that are not only provably efficient but also effective in practice.

References

[1] G. Aloupis, C. Cortes, F. Gomez, M. Soss, and G. Toussaint. Lower bounds for computing statistical depth. *Computational Statistics & Data Analysis*, 40(2):223–229, 2002.

[2] G. Aloupis, S. Langerman, M. Soss, and G. Toussaint. Algorithms for bivariate medians and a Fermat-Torricelli problem for lines. *Comp. Geom. Theory and Appl.*, 26(1):69–79, 2003.

[3] V. Barnett. The ordering of multivariate data. *J. Roy. Statist. Soc. Ser. A*, 139(3):318–355, 1976.

[4] J. L. Bentley and T. A. Ottmann. Algorithms for reporting and counting geometric intersections. *IEEE Trans. Comput.*, C-28(9):643–647, September 1979.

[5] J. L. Bentley and M. I. Shamos. A problem in multivariate statistics: algorithm, data structure, and applications. In *Proc. 15th Allerton Conf. Commun. Control Comput.*, pages 193–201, 1977.

[6] K. Q. Brown. *Geometric transforms for fast geometric algorithms.* Ph.D. thesis, Dept. Comput. Sci., Carnegie-Mellon Univ., Pittsburgh, PA, 1980. Report CMU-CS-80-101.

[7] K. Q. Brown. Comments on “Algorithms for reporting and counting geometric intersections”. *IEEE Trans. Comput.*, C-30:147–148, 1981.

[8] M. Burr, E. Rafalin, and D. L. Souvaine. Simplicial depth: An improved definition, analysis, and efficiency for the discrete case. DIMACS technical report 2003-28, Rutgers University–New Brunswick, 2003. `http://dimacs.rutgers.edu/TechnicalReports/abstracts/2003/2003-28.html`.

[9] A. Y. Cheng and M. Ouyang. On algorithms for simplicial depth. In *Proc. 13th Canadian Conference on Computational Geometry*, pages 53–56, 2001.

[10] K. L. Clarkson, D. Eppstein, G. L. Miller, C. Sturtivant, and S.-H. Teng. Approximating center points with iterated Radon points. *Int. J. Comp. Geom. Appl.*, 6(3):357–377, 1996.

[11] R. Cole, M. Sharir, and C. K. Yap. On k-hulls and related problems. *SIAM Journal on Computing*, 15(1):61–77, 1987.

[12] M. de Berg, M. van Kreveld, M. Overmars, and O. Schwarzkopf. *Computational Geometry: Algorithms and Applications.* Springer-Verlag, Berlin, 1997.

[13] D. P Dobkin and D. L. Souvaine. *Advances in Robotics 1: Algorithmic and Geometric Aspects of Robotics*, chapter Computational Geometry - A User’s Guide, pages 43–93. Lawrence Erlbaum Associates, 1987.

[14] W. Eddy. Convex hull peeling. In H. Caussinus, editor, *COMPSTAT*, pages 42–47. Physica-Verlag, Wien, 1982.

[15] H. Edelsbrunner. *Algorithms in Combinatorial Geometry.* Springer-Verlag, Berlin, 1987.

[16] H. Edelsbrunner and L. J. Guibas. Topologically sweeping an arrangement. *J. Comput. Syst. Sci.*, 38:165–194, 1989.

[17] H. Edelsbrunner and D. L. Souvaine. Computing median-of-squares regression lines and guided topological sweep. *J. Amer. Statist. Assoc.*, 85:115–119, 1990.

[18] J. Gil, W. Steiger, and A. Wigderson. Geometric medians. *Discrete Math.*, 108(1-3):37–51, 1992. Topological, algebraical and combinatorial structures. Frolík's memorial volume.

[19] J. Hodges. A bivariate sign test. *The Annals of Mathematical Statistics*, 26:523–527, 1955.

[20] S. Jadhav and A. Mukhopadhyay. Computing a centerpoint of a finite planar set of points in linear time. *Discrete & Computational Geometry*, 12(3):291–312, 1994.

[21] T. Johnson, I. Kwok, and R. Ng. Fast computation of 2-dimensional depth contours. In *Proc. 4th International Conference on Knowledge Discovery and Data Mining*, pages 224–228, 1998.

[22] S. Khuller and J. S. B. Mitchell. On a triangle counting problem. *Inform. Process. Lett.*, 33(6):319–321, 1990.

[23] S. Krishnan, N. H. Mustafa, and S. Venkatasubramanian. Hardware-assisted computation of depth contours. In *13th ACM-SIAM Symposium on Discrete Algorithms*, 2002.

[24] S. Langerman and W. Steiger. Computing a maximal depth point in the plane. In *Proceedings 4th Japan Conf. on Discrete and Computational Geometry*, 2000.

[25] R. Liu. On a notion of data depth based on random simplices. *The Annals of Statistics*, 18:405–414, 1990.

[26] J. Matoušek. Computing the center of planar point sets. *DIMACS Series in Discrete Mathematics and Theoretical Computer Science*, 6:221–230, 1991.

[27] K. Miller, S. Ramaswami, P. Rousseeuw, T. Sellarés, D. Souvaine, I. Streinu, and A. Struyf. Efficient computation of location depth contours by methods of combinatorial geometry. *Statistics and Computing*, 13(2):153–162, 2003.

[28] D. M. Mount, N. S. Netanyahu, K. R. Romanik, R. Silverman, and A. Y. Yu. A practical approximation algorithm for the LMS line estimator. In *Proc. 8th ACM-SIAM Sympos. Discrete Algorithms*, pages 473–482, January 1997.

[29] N. Naor and M. Sharir. Computing a point in the center of a point set in three dimensions. In *Proc. 2nd Canadian Conference on Computational Geometry*, pages 10–13, 1990.

[30] H. Oja. Descriptive statistics for multivariate distributions. *Statistics and Probability Letters*, 1:327–332, 1983.

[31] E. Rafalin, D. Souvaine, and I. Streinu. Topological sweep in degenerate cases. In *Proc. 4th Workshop on Algorithm Engineering and Experiments (ALENEX)*, volume 2409 of *Lecture Notes in Comput. Sci.*, pages 577–588, Berlin, 2002. Springer-Verlag.

[32] V. Rosta and K. Fukuda. Exact parallel algorithms for the location depth and the maximum feasible subsystem problems. In C.A. Floudas and P.M. Pardalos, editors, *Frontiers In Global Optimization.* Kluwer Academic Publishers, 2003.

[33] P. J. Rousseeuw. Least median of squares regression. *J. Amer. Statist. Assoc.*, 79(388):871–880, December 1984.

[34] P. J. Rousseeuw and M. Hubert. Regression depth. *J. Amer. Statist. Assoc.*, 94:388–433 (with discussion), 1999.

[35] P. J. Rousseeuw and I. Ruts. Bivariate location depth. *Applied Statistics-Journal of the Royal Statistical Society Series C*, 45(4):516–526, 1996.

[36] P. J. Rousseeuw and I. Ruts. Constructing the bivariate Tukey median. *Statistica Sinica*, 8:827–839, 1998.

[37] P. J. Rousseeuw and A. Struyf. Computing location depth and regression depth in higher dimensions. *Statistics and Computing*, 8:193–203, 1998.

[38] I. Ruts and P. J. Rousseeuw. Computing depth contours of bivariate point clouds. *Computational Statistics and Data Analysis*, 23:153–168, 1996.

[39] M. I. Shamos. Geometry and statistics: problems at the interface. In J. F. Traub, editor, *Algorithms and Complexity: New Directions and Recent Results*, pages 251–280. Academic Press, New York, NY, 1976.

[40] M. I. Shamos. *Computational Geometry.* Ph.D. thesis, Dept. Comput. Sci., Yale Univ., New Haven, CT, 1978.

[41] K. Singh. On the majority data depth. Technical report, Department of Statistics, Rutgers University, 1993.

[42] D. L. Souvaine and J. M. Steele. Time- and space- efficient algorithms for least median of squares regression. *J. Amer. Statist. Assoc.*, 82:794–801, 1987.

[43] A. Struyf and P. Rousseeuw. High-dimensional computation of the deepest location. *Computational Statistics & Data Analysis*, 34:415–426, 2000.

[44] J. W. Tukey. Mathematics and the picturing of data. In *Proceedings of the International Congress of Mathematicians (Vancouver, B. C., 1974), Vol. 2*, pages 523–531. Canad. Math. Congress, Montreal, Que., 1975.

[45] K. Verbarg. Approximate center points in dense point sets. *Inform. Process. Lett.*, 61(5):271–278, 1997.

[46] Y. Zuo and R. Serfling. General notions of statistical depth function. *Ann. Statist.*, 28(2):461–482, 2000.