Dynamic Update of Half-space Depth Contours

Michael A. Burr[1], Eynat Rafalin[1], and Diane L. Souvaine[1]

Tufts University, Medford MA 02155, USA,
mburr,erafalin,dls@cs.tufts.edu

Abstract. *Data depth* is a statistical analysis method that assigns a numeric value to each point, corresponding to its centrality relative to a data set. *Depth contours*, nested contours that enclose regions with increasing depth, provide a powerful method to visualize, quantify and compare data sets. We present the first *dynamic algorithm* for computing the half-space depth contours of a set of n points in $\mathbb{R}^2$ in $O(n \log n)$ and $O(n \log^2 n)$ time per operation for two flavors of depth contours and in quadratic space, an improvement over the static version of $O(n^2)$ time per operation. Our analysis characterizes key constraints on potential contour changes under insertion and deletion, that have not been identified before.

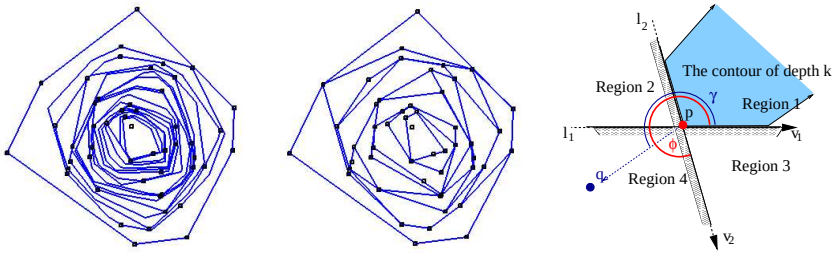
1 Introduction

Data depth, a statistical analysis method, measures how deep (or central) a given point $x \in \mathbb{R}^d$ is relative to a probability distribution or relative to a given data cloud, by assigning x a *depth value*. *Data depth* has recently attracted attention from the computational geometry community. It does not require prior assumptions on the probability distribution of data and handles observations that deviate from the data (outliers). *Depth contours* are nested contours that enclose regions of increasing depth [23]. They provide a powerful tool to visualize, quantify, and compare data sets (e.g. [19, 14]).

The statistics community has produced distinct definitions of depth contours, yielding distinctive contours. The two main approaches were termed *cover* and *rank* [18] (Fig. 1(a-b)). The **sample *cover* contour of depth** d [23] is the boundary of the set of all points (whether from the original data set or not) with depth at least d. This model assumes that the data represents a random sample from a probability distribution and that the contours should represent the behavior of the distribution and not the specific points. The **sample *rank* αth region** is the convex hull containing the most central fraction of α sample points [14]. A contour that encloses, for example, 75% of the points is created by sorting all points of the original set according to their depth and taking the convex hull of the $\lceil .75n \rceil$ deepest data points. Both approaches create nested depth contours, but may not assign the same contour depth to points of $\mathbb{R}^d \setminus \mathcal{S}$. Vertices of *rank* contours are only data points and may appear on multiple contours, while vertices of the *cover* contours can be any point from $\mathbb{R}^d$ and each will appear only on a single contour. As n approaches infinity, both contours

converge to the same set [14]. Although *rank contours* lack some of the structure of *cover contours*, they often provide a reasonable and less expensive analysis.

The **half-space depth**[1] of a point x relative to a set $\mathcal{S} = \{X_1, ..., X_n\}$ in $\mathbb{R}^d$ is the minimum number of points of $\mathcal{S}$ lying in any closed half-space determined by a line through x [10, 23]. The maximal depth is at least $\lfloor n/3 \rfloor$, by Helly's theorem, and for sets in general position is less than $\lfloor n/2 \rfloor$.



(a) **(b)** **(c)**

Fig. 1. Half-space depth: (a) All *cover contours* for 50 points from a bivariate normal distribution of mean (0,0), $cov = 4I$; (b) The $10\%, 20\% \ldots 100\%$ *rank contours* for the same set; (c) Regions defined by points p, q.

Half-space depth contours are convex and *robust* in the presence of outliers: the existence of m outliers will not change the depth of the deepest points [8]. In $\mathbb{R}^2$, any half-space *cover-contour*-edge lies on a segment between two data points. At most $\binom{n}{2}$ such segments exists, so the complexity of the collection of half-space cover contours is $O(n^2)$. This bound is tight, for example, when all points of the data set lie on the convex hull of the set. *Rank-based* depth contours, by definition, enclose the $1st, \frac{n-1}{n}th, \ldots, \frac{1}{n}th$ sample regions, have a total complexity of $O(n^2)$, but contain only $\Theta(n)$ distinct edges as many contours may share edges.

The best implemented algorithms (expanding upon a theoretical result in [6]) for computing all *cover contours* and the depth of all data points in $\mathbb{R}^2$ (used to compute *rank contours*) run in $\Theta(n^2)$ time [16]. Other implementations exist [21, 9, 11]. To the best of our knowledge, however, no dynamic algorithm (allowing insertion and deletion) exists, despite interest of the statistical community [13]. Other research results on half-space depth include algorithms for computing the half-space depth of a single point in $\mathbb{R}^2$ in $O(n \log n)$ [21], and computation of a single contour relative to the data set in two dimensions in $O(n \log^2 n)$ [5] (see also [21, 20, 22, 15, 12, 1]).

This paper presents the first *dynamic algorithm* for computing the two-dimensional half-space depth contours of a set of n points in $O(n \log n)$ time per operation for *rank* contours and $O(n \log^2 n)$ time per operation for *cover* contours, both using overall quadratic space, an improvement over the static version of $O(n^2)$ time per operation. Our *rank*-based algorithm is based on key constraints on the potential changes in the depth contours during insertion or

[1] Also called *location depth* or *Tukey depth*.

deletion. The data structure, independantly presented in [24], is augmented to enable dynamic updates and to track the changes in the depth contours. Our *cover*-based algorithm is based on a dynamic scheme for updating the convex hull of a set of points [17]. In the paper, *half-plane* means *closed half-plane* unless specified otherwise. We assume that all data sets are in general position.

2 Dynamically Computing Contours According to Rank

Our algorithm computes *rank-based* half-space-depth contours dynamically in $O(n \log n)$ time and $O(n^2)$ space and the half-space depth of a single point in $O(\log n)$ time and $O(n)$ space. The algorithm does **not** explicitly compute the set of contours: it maintains the order (ranking) of points according to depth. Each point's depth is preserved by maintaining the cover contours locally. From the list of data points sorted by *x-coordinate* and labeled by depth, a constant number of contours (e.g. $10\%, 20\%, \ldots$) can be constructed in $O(n)$ time.

Inserting point q into data set $\mathcal{S}$ affects the depth of a point $p \in \mathcal{S}$ currently on the depth-k *cover contour* only if q lies outside that contour. The point p has two *defining lines* generated by its two incident edges on the contour. The *defining lines* divide the plane into four regions relative to p (Fig. 1(c)). Our algorithm updates both the depth and the defining lines for each point $p \in \mathcal{S}$ by deciding which region contains q and choosing one of four cases.

2.1 Basic Terminology

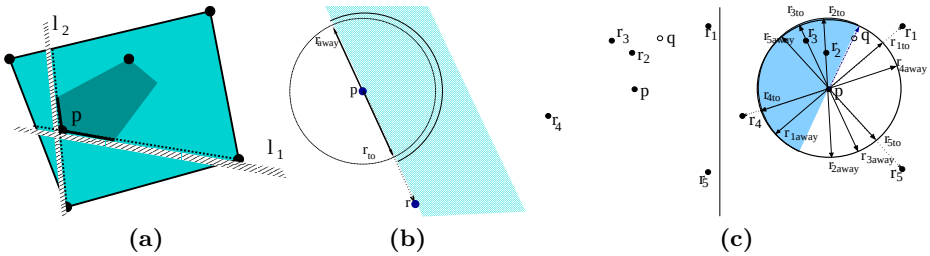
Any closed half-plane H whose bounding line passes through point x lying inside the region bounded by the kth contour **must** contain at least k data points (by definition). The k-th cover contour is the boundary of the intersection of all half-planes containing exactly $n - k + 1$ data points whose bounding line contains two data points [1]. A data point p of depth k appears exactly once on the kth contour; if the contour is non-degenerate, p has exactly two incident edges on the contour. Every contour edge is a sub-segment of a line joining two data points q_1, q_2 of $\mathcal{S}$ [1]. The *defining edges* of point p relative to set $\mathcal{S}$ are the two edges incident to p on the contour of depth k; the *defining lines* l_1, l_2 are the lines containing the *defining edges* (see Fig. 2(c)).

Each defining line l_i bisects the plane into two closed half-planes containing $k+1$ and $n-k+1$ data points, respectively [1]. The *defining half-plane* H_{l_i}, for defining line l_i, is the one that contains $k+1$ data points for $1 \leq k \leq \lfloor \frac{n}{2} \rfloor$ and does *not* contain the k-th depth contour. A defining half-plane H_l of p relative to $\mathcal{S}$ has *defining edge* $E_S(l)$, associated defining line l, and a closed half-plane H_l^C which is the closure of its complement. Every contour except the inner-most must have nonempty interior, so almost every contour vertex has two distinct *defining edges*. The innermost contour has empty interior if and only if the contour is a single point[2].

2.2 The Main Technique: Transformation of Half-Planes

The algorithm recomputes the depth and the defining edges for each data point $p \in \mathcal{S}$ in $O(\log n)$ time by analyzing the position of inserted/removed point q

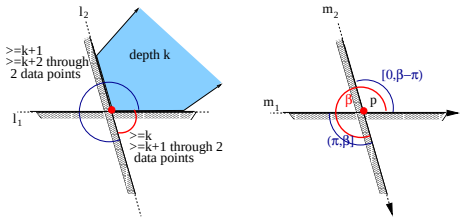
[2] A contour of two data points and the connecting segment appears only in degenerate data sets or a data set of exactly two points.



(a) (b) (c)

Fig. 2. (a) Defining lines, planes and edges of point p relative to a set of six points. (b) The transformation of the half-planes associated with point r to vectors and the semicircle that contains all vectors whose associated half-planes are incremented by one when r is inserted into a data set. (c) A transformation of the lines defined by point p and five data points into their associated vectors. The greyed area represented the set of vectors whose depth (the number of data points in their associated half-planes) increases by one when q is inserted.

relative to the defining half-planes of p. To achieve the desired efficiency, each point p has its own individual data structure defined as follows: map each half-plane H_m bounded by line m through p and one of the other $n-1$ data points to the unit vector v_{H_m} that contains the points of H_m to its right, resulting in $2(n-1)$ unit vectors at each data point (maintained as points on a unit circle); each of these vectors is either $\hat{r}_{to}$ or $\hat{r}_{away}$, where $\hat{r}_{to}$ refers to the vector pointing towards $r \in \mathcal{S} \backslash \{p\}$ (see Fig. 2(b-c)). Upon insertion/deletion of point q, the count of data points in each of the half-planes represented by these vectors around p can be updated in sub-linear time as only a contiguous set of half-planes will be affected. As half-space depth is affine invariant and not affected by rotation, translation and reflection, p's data structure can be represented thus: p lies at the origin; the vector $v_{H_{l_1}}$ for one defining half-plane and the associated defining edge lie on the positive x-axis; the vector $v_{H_{l_2}}$ for the other defining half-plane points into the lower semicircle with angle[3] ϕ and its defining edge into the upper (see Fig. 2(b-c)).



(a) (b)

Fig. 3. (a) Counting and converse counting properties. (b) Location property.

2.3 Properties of the Dynamic Contours

Property 1. **Convexity property:** Every half-space depth contour is convex, constraining the angle between the two vectors associated with the defining half-

[3] The angle of a vector is the counter-clockwise angle from the positive x-axis to the vector.

planes, H_{l_1} and H_{l_2}, to $[\pi, 2\pi]$. If edge $E_S(l_2)$ is adjacent to $E_S(l_1)$ clockwise along the kth contour, $v_{H_{l_1}}$ points towards the positive x-axis, and the interior of the contour lies immediately above the positive x-axis, then $v_{H_{l_2}}$ points into the lower semicircle and the angle of v_{l_2} is in $(\pi, 2\pi)$.

Property 2. **Insertion Property:** When a data point $r \notin \mathcal{S}$ is added to (resp. $r \in \mathcal{S}$ is deleted from) $\mathcal{S}$, the number of data points increase (decrease) by one in all half-planes that contain r: those whose vectors lie in counter-clockwise sector from $\hat{r}_{to}$ to $\hat{r}_{away}$. If $\hat{r}_{to}$ has angle γ then the affected half-planes have angles $[\gamma, \gamma + \pi]$ (Fig. 2(b)).

Property 3. **Counting Property:** If the bounding line for half-plane H_m passes through point p on the boundary of the kth contour then if it intersects the interior of the kth contour it must contain $> k$ points and if it does not intersect the interior of the contour it must contain $\geq k$ points (see Fig. 3(a)).

Assume point p lies on contour k with defining half-planes H_{l_1} and H_{l_2} where $v_{H_{l_1}}$ points in the direction of the positive x-axis and $v_{H_{l_2}}$ points into the lower semicircle with angle ϕ. For any unit vector v_{H_m} with angle β, consider the number of data points in its the associated half-plane.

- If $0 < \beta < \phi$, then the half-plane H_m intersects the interior of the kth contour and must contain $> k$ data points; if m passes through p and a second data point, then H_m must contain $> k+1$ data points. Otherwise (for both cases) there would exist a half-plane containing $\leq k$ points intersecting the inside of the kth contour.
- If $0 = \beta$ or $\phi = \beta$ then H_m is a defining half-plane for p and contains exactly $k + 1$ points.
- If $\phi < \beta < 2\pi$, then H_m contains $\geq k$ data points; if m passes through p and another data point, then H_m contains $\geq k + 1$ data points (otherwise p would have depth $< k$ and could not appear on the kth contour).

Property 4. **Converse counting property:** If the half-planes whose bounding lines passes through point p follow the conclusions of the counting property, then p must be on the kth depth contour. Assume that there is a continuous region on the unit circle, $B = (0, \beta)$, containing vectors whose associated half-planes have $\geq k + 1$ data points; the half-planes defined by p and another data point whose vectors are in B contain $k + 2$ points; the half-planes whose associated vectors are on the boundary of B contain exactly $k + 1$ and whose bounding line contains two points; the half-planes whose vectors are in the complementary region B^C contain $\geq k$ points. Then point p must be on the kth depth contour: p cannot be on the jth depth contour where $j < k$ because no half-plane that passes through p contains $< k$ points. On the other hand p cannot be on the jth contour where $j > k$ because there is a half-plane whose bounding line contains p with a lower number of points.

Property 5. **Location property:** Given two arbitrary half-planes H_{m_1}, H_{m_2}, each passing through p and one other data point and containing $k + 1$ data points, then the vectors for the defining half-planes are restricted to lie in two small intervals (see Fig. 3(b)). Assume $v_{H_{m_1}}$ points towards the positive x-axis and $v_{H_{m_2}}$ points into the lower semicircle with angle β. By the converse counting

property, the angles of the associated vectors of the *defining half-planes* must be in the interval $[0, \beta]$. Additionally, if an associated vector for a defining half-plane makes an angle in the interval $[\beta - \pi, \pi]$, then the union of this half-plane, H_{m_1}, and H_{m_2} is the entire plane, contradicting the assumption that p is on a non-trivial contour. Therefore, the angles of the associated vectors for the defining half-planes must come from $[0, \beta - \pi)$ and $(\pi, \beta]$, one vector from each interval.

Property 6. **Nestedness property:** When a point q is added to set $\mathcal{S}$, the kth depth contour relative to $\mathcal{S}$ is nested in the kth depth contour relative to $\mathcal{S} \cup \{q\}$. When q is added the depth of all half-planes either increases or remains the same. Thus, the depth of every point either increases or remains the same. Assume p lies on a nontrivial contour with defining half-planes H_{l_1} and H_{l_2} and the depth of p remains unchanged when a data point is inserted. If $v_{H_{l_1}}$ points towards the positive x-axis and $v_{H_{l_2}}$ points into the southern semicircle with angle ϕ, then the angles of the associated vectors for the defining half-planes after q is inserted must be in the interval $\{0\} \cup [\phi, 2\pi)$.

2.4 Possible Cases

Consider a data set $\mathcal{S}$ before and after an update step, where point q is either inserted into or deleted from $\mathcal{S}$. Assume point p has depth k and lies on a nontrivial contour before the update. Let the two defining half-planes of p in $\mathcal{S}$ be H_{l_1} and H_{l_2}. If p lies on a nontrivial contour after q is inserted or deleted, let the two defining half-planes for p in $\mathcal{S} \cup \{q\}$ or $\mathcal{S} \setminus \{q\}$ be H_{m_1} and H_{m_2}. Assume $v_{H_{l_1}}$ points towards the positive x-axis and $v_{H_{l_2}}$ has angle ϕ. Lines l_1 and l_2 divide the plane into 4 regions (see Fig. 1(c)). Call region $(H_{l_1})^C \cap (H_{l_2})^C$ *region 1.* Call $H_{l_1} \cap (H_{l_2})^C$ *region 2* and $(H_{l_1})^C \cap H_{l_2}$ *region 3* and call $H_{l_1} \cap H_{l_2}$ *region 4*. Let v be the unit vector from p towards q and let γ be v's angle.

Consider the possible cases created by q relative to the four regions. Point q can be inserted only into the interior of one of the four regions (inserting into the boundary will violate general position) and deleted from the interior of any of the four regions or a defining edge. The case where p is on a degenerate contour before or after an insertion or deletion are handled separately. We present an analysis of the cases created when a point q is inserted into the set. The cases associated with deletion can be analysed using similar tools and Lemma 1.

Case 1. **Point q is inserted into region 1:** q is inserted into neither of the defining half-planes H_{l_1}, H_{l_2}, so $0 < \gamma < \phi - \pi$. Since the number of data points contained in H_{l_1} and H_{l_2} does not change, the depth of p remains k by the converse counting property. By the insertion property all half-planes with vectors whose angles are between γ and $\gamma + \pi$ include an additional point. By the nestedness property, the new defining-half planes must be in the interval $\{0\} \cup [\phi, 2\pi)$. By the location property, the defining half-planes must also be in the intervals $[0, \phi - \pi)$ and $(\pi, \phi]$. The intersection of the two sets is $\{0, \phi\}$. Since after the update p lies on a nontrivial contour, then H_{m_1} and H_{m_2} are distinct so the defining half-planes for p remain unchanged.

Case 2. **Point q is inserted into region 2 or 3:** q is inserted into exactly one defining half-plane. Assume q is inserted into region 3, thus H_{l_2} contains an extra point and H_{l_1} does not contain an extra point and $\phi - \pi < \gamma < \pi$. By

the insertion property all half-planes with angles between γ and $\gamma + \pi$ include an additional point. H_{l_l} will not contain an additional point, so the depth of p will remain k, but H_{l_2} will include an extra point so it can no longer be a defining half-plane for p by the counting property. By the nestedness property, the angle of vector $v_{H_{m_2}}$ must be in the interval $U_1 = 0 \cup [\phi, 2\pi)$. However, by the insertion property, every defining plane whose vector has an angle in the interval $U_2 = [\gamma, \gamma + \pi]$ includes an additional point, so the angle for $v_{H_{m_2}}$ must be in the interval $U_1 \setminus U_2 = [\gamma + \pi, 2\pi]$. By the counting property $v_{H_{m_2}}$ must be the first vector encountered when traversing the vectors starting from $v_{H_{l_2}}$ counter-clockwise, whose associated half-plane contains $k + 1$ data points.

Assume $v_{H_{m_2}}$ has angle β and consider the location of the second defining half-plane H_{m_1}: By the location property, the associated vectors to the new defining half-planes must be in the intervals $V_1 = [0, \beta - \pi)$ and $V_2 = (\pi, \beta]$, so $v_{H_{m_1}}$ must be in the interval $[0, \beta - \pi)$, since $v_{H_{m_2}} \in (\pi, \beta]$. However, by the nestedness property, the angles of the associated vectors for each of the defining half-plane must also be in U_1, but $U_1 \cap V_1 = \{0\}$. Therefore, the associated vector for H_{m_1} must have angle 0, and $H_{m_1} = H_{l_2}$.

Case 3. **Point q is inserted into region 4:** q is inserted into both defining half-planes, thus $\pi < \gamma < \phi$. By the insertion property every half-plane whose associated vector has angle in the interval $[\gamma, \gamma + \pi]$ gains an additional point, in particular, in the interval $\{0\} \cup [\phi, 2\pi)$. Let $U = [\gamma, 2\pi) \cup [0, \gamma - \pi]$. By the counting property, every half-plane whose vector has angle in the complement of U already contains at least $k + 1$ data points and every half-plane whose vector has angle in U, which previously contained at least k data points, now contains at least $k + 1$. Therefore the depth of p increases by one to $k + 1$.

The two defining half-planes of p relative to $\mathcal{S}$, H_{l_1} and H_{l_2} contain $k + 2$ data points each. By the location property the angles of the associated vectors $v_{H_{m_1}}, v_{H_{m_2}}$ must be in the intervals $[0, \phi - \pi)$ and $(\pi, \phi]$. Assume that H_{l_2} is no longer a defining half-plane for p, then the angle of $v_{H_{m_2}}$ is in the interval (π, ϕ). By the counting property, combined with the insertion property, every half-plane whose vector has angle in (γ, ϕ) contains at least $k + 2$ data points and whose bounding line passes through two data points must contain $k+3$ data points and cannot be a defining half-plane for p. Thus, the angle for $v_{H_{m_2}}$ must be in the interval $(\pi, \gamma]$[4]. Assume $v_{H_{m_2}}$ has angle β. By the counting property $v_{H_{m_2}}$ must be the first vector counter-clockwise from π to γ where m_2 passes through two data points and H_{m_2} contains $k + 2$ data points[5].

Now, assume that H_{l_1} is also no longer a defining half-plane for p. Then by the location property the angle for v_{m_1}, must be in the interval $(0, \beta - \pi)$. By construction, $\beta \leq \gamma$, so $\beta - \pi \leq \gamma - \pi$. By the counting property, combined with the insertion property, every half-plane whose vector has angle in $(0, \gamma - \pi)$

[4] The associated vector cannot have angle γ.

[5] If instead, this associated vector has angle in the interval $(0, \phi - \pi)$ then it will be the first vector whose bounding line passes through two data points, and whose associated half-plane contains exactly $k + 2$ data points, when traversing the associated vectors in the clockwise direction starting from $\gamma - \pi$.

contains at least $k+2$ data points and whose bounding line passes through two data points must contain $k+3$ data points and cannot be a defining half-plane for p. However, since $(0, \beta - \pi) \subseteq (0, \gamma - \pi)$ there is a contradiction since the angle for v_{m_1} must be in that interval. Thus $v_{m_1} = v_{l_1}$ and $H_{m_1} = H_{l_1}$.

The **degenerate case** where p's contour becomes a trivial contour[6] is handled separately by checking that p's new depth is $\frac{n-1}{2}$.

Lemma 1. *Let p be a data point which lies on a nontrivial half-space depth contour before and after an update. Assume that point q is inserted into (or deleted from) $\mathcal{S}$. Then at least one of the two defining half-planes of p (relative to $\mathcal{S}$) remains unchanged relative to the new set.*

Proof. By cases 1-3, when q is inserted into $\mathcal{S}$, one of the defining half-planes remains unchanged and the lemma is proved for insertions. To prove the claim for deletions consider first deleting and then reinserting q and compare the defining half-planes for p for each step.

2.5 The Algorithm

Our algorithm dynamically computes the half-space-depth ranking-order of a set of points in $O(n \log n)$ time per operation and $O(n^2)$ space by recomputing the depth of every data point and resorting the updated list of data points.

A static version of the data structure was presented by van-Kreveld *et al.* [24], where it was used to report the *hyperplane depth* of a point relative to a set of lines. Our version is augmented to allow dynamic updates and to compute the defining lines. Each copy of the data structure, representing a single data point $p \in \mathcal{S}$, is a *segment tree* [2], a data structure used to report all line segments that cover a given point, and is implemented as an augmented dynamic red-black tree. Our implementation is based on the idea of representing a set of *radially sorted vectors* (the set of all half-planes relative to p) by a balanced binary tree. The *leaves* of the tree represent the $2(n-1)$ vectors associated with the lines between p and the data points in $\mathcal{S} \setminus p$, sorted according to their radial order in the range $[0, 2\pi)$. An *insertion* (resp. *deletion*) of a segment $[a, b]$ ($|a - b| = \pi$) corresponds to the insertion (resp. deletion) of a point q (with line $\overline{pq}$ in direction a) that affects all half-planes between the directions a and b. An internal node v of the tree corresponds to the interval that is the union of elementary intervals of the leaves of the tree rooted in v. Our implementation counts the number of segments covering a point, requiring a linear size segment tree. We achieve a dynamic set of intervals by building the segment tree as a *red-black tree* [7], adding one extra *color* bit to each node[7].

Every leaf or node v contains the usual red-black tree fields: its color, and pointers to its parent and left and right children. The *leaves* are augmented

[6] Can happen only if p's depth increases, since by the nestendness property, if p remains on the same contour, the contour can only increase its size.

[7] *Red-black trees* 'ensure that no path from the root to a leaf is more than twice as long as any other' [7] and thus limit the time required for insertion, deletion and search in the tree to $O(\log n)$.

with the direction of their associated vector and the `depth` value for the associated half-plane, that is updated as the algorithm proceeds. Every inner node v, associated with the leaves in its subtree, contains, in addition, the minimum and maximum `depth` values in v's subtree `minDepth(v), maxDepth(v)` and a `subtree-addition(v)` field, holding the depth added to or deleted from the `depth` of v's leaves.

When the `depth` is incremented or decremented, the `depth`, `minDepth` and `maxDepth` values of the affected leaves are updated. To maintain the time complexity, only the values associated with the nodes visited in every step of the algorithm are changed. If v's subtree was not visited then `subtree-addition(v)` is the additional depth added to or removed from the `depth` of all the leaves in v's subtree, and consequently added to or removed from the `minDepth(v)` or `maxDepth(v)` fields in all of v's descendants. Every operation of the algorithm consists of a traversal from the root to a leaf and a traversal back to the root, where only the nodes in this path are queried. Going down the tree the `subtree-addition` field of a node v visited is used to update the `minDepth, maxDepth` of v, is reset to zero and its value is than transferred to the `subtree-addition` of its children, until it reaches a leaf. The logarithmic bound on the maximal height of the red-black tree guarantees that every operation costs $O(\log n)$.

To recompute a defining edge for a point whose new depth is k, we locate the vector immidiately before or after the vector $\hat{q}$ (corresponding to the line segment $\overline{pq}$) with depth k. First the tree is traversed starting at the root to locate leaf q and to update the `depth`, `minDepth` and `maxDepth` fields. Next, to search for the leaf of depth k adjacent to vector $\hat{q}$, the path from $\hat{q}$ is traversed up towards the root until the subtree containing the requested leaf is first encountered. At each step up the tree *for every node checked v (1) the `depth, minDepth, maxDepth` fields of v are correct (2) its subtree does not contain the requested leaf, a leaf of depth k that has a larger direction than $\hat{q}$.* Lastly, the subtree s is traversed down towards the requested leaf. At each step down the tree, for every node v checked, its subtree must contain the requested leaf.

Dynamic computation of the half-space-depth of a *single* point relative to a dynamic set of n points requires $O(\log n)$ time per operation and linear space using a *single* copy of the segment tree.

To re-sort data points, we use a linked list of buckets, from 0 to $n/2$, bucket k holding a linked list of data points of depth k. Upon insertion or deletion every point q that changes its depth is moved from its old bucket to a new bucket. Since the depth of q changes by at most one, the update takes $O(1)$ time

3 Dynamic Computation of Cover-Based Depth Contours

Our algorithm computes the cover-based half-space-depth contours of a set of points dynamically in $O(n \log^2 n)$ time and overall quadratic space, based on dynamic computation of a convex hull [17]. To compute the depth contours we use the duality transform T [4], mapping a point $P = (a, b)$ to a line $T(P) : y = ax+b$ and a line $l : y = cx+d$ to point $T(l) : (-c, d)$. 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 line through I in the dual intersects every line of the arrangement. 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 (l the *level* of L, or $n-l-1$). Examining all intersection points in the arrangement of dual lines produces a list of half-planes/intersections for each potential contour. The upper (resp. lower) convex hull of a set of points is the lower (resp. upper) envelope of the set of dual lines and thus the $k-th$ depth contour is the intersection of the upper envelope of level k and the lower envelope of level $n-k-1$, see also [16].

We use Overmars *et al.*'s [17] data structure for dynamically updating the convex hull of a set of points in $O(\log^2 n)$ per update and in linear overall space[8]. Overmars *et al.*'s data structure is implemented using a balanced binary search tree, where the leaves of the tree hold data points sorted according to x-coordinate. Every node is associated with a structure Q_ϕ, the upper or lower hull of the points stored in the leaves of ϕ. To decrease space to linear, every node ϕ is associated with a fragment Q^*_ϕ of Q_ϕ, that was not used in building the hull of ϕ's father. The root of the tree stores the convex hull of the entire set of points. Q_ϕ and Q^*_ϕ are implemented using a *concatenable queue*, a binary search tree which enables efficient searching, splitting and concatenating.

When a point is inserted to (resp. removed from) the data set and consequently to the data structure, the tree is split along the path from the root to the leaf associated with the data point and then reassembled, after the associated leaf has been added to (resp. removed from) the tree. As the tree is split, the Q_ϕ structure associated with every node along the split path, starting from the root, is split and Q_γ and Q_δ, the structures associated with the children of ϕ are constructed. Q_ϕ is split into its left and right chains, which are concatenated to the remainder of the hulls, Q^*_γ, Q^*_δ. As the tree is merged, Q_ϕ associated with every node along the merge path is constructed, starting from the leaves, as the concatenation of the head of Q_γ and tail of Q_δ associated with the children γ, δ of ϕ, in $O(\log n)$ time, by locating the bridge between Q_γ and Q_δ.

Our implementation uses a combination of Overmars *et al.*'s data structure and a red-black tree to enable, given a set of points $\mathcal{S}$, a *split*, *join*[9] and *rebalancing* of the tree in $O(\log^2 n)$ time each. Two dynamic trees maintain the upper and lower convex hulls for every level of the arrangement, $2n$ trees for all levels and total of $O(n^2)$ space, the size of the entire arrangement.

When a new point q is inserted into or deleted from the data set some of the dual levels, the ones intersected by the line $T(q)$, and consequently some of the

[8] A theoretical algorithm with an improved amortized $O(\log n)$ running time [3] exists, but might not be applicable to our method as it does not store the contour explicitly. Algorithms for dynamically computing convex hulls under restricted cases (insertions or deletions only) exist, with improved time bounds. However, our method requires to *split* and *merge* the representation of the convex hulls, which some of the algorithms for the restricted cases do not allow.

[9] Two trees $\mathcal{S}_1, \mathcal{S}_2$ and a point p where $\forall p_1 \in \mathcal{S}_1, p_2 \in \mathcal{S}_2$, $xcoord(p_1) < xcoord(p) < xcoord(p_2)$ are merged, based on a scheme for merging two red-black trees [7].

convex hulls of these levels, are affected. We compute the new convex hulls by updating the convex hull trees.

Insertions After insertion of data point $q \notin \mathcal{S}$ (and its dual $T(q)$), the *old* level k, intersected by $T(q)$, splits between levels k and $k+1$. The change to the split level k, that becomes level $k+1$, includes deleting chains along the level (and adding them to the $k-1$ level, now level k) and inserting chains from the $k+1$ level (which is now $k+2$). In every stitching point a segment of $T(q)$ is added. Not all levels are necessarily updated, only those intersected by $T(q)$, forming a set between levels k' and $n-k'-1$, the first and last affected levels, where k' is the depth of the inserted point. Since $T(q)$ intersects the arrangement n times, $O(n)$ chains move between all trees and n segments (from $T(q)$) are added.

Deletions When point $q \in \mathcal{S}$ and its dual line $T(q)$ is removed from the arrangement, level k splits between levels $k-1$ and $k-2$. Since every intersected segment along $T(q)$ has a pointer to the segment that crosses it and vice versa, we follow these pointers and split the contour trees along every intersection point with segment s_i of $T(q)$[10]. The chains are joined by following s_i's pointers to its intersection points: start with the first chain affected and stitch every pair of adjacent subtrees.

4 Discussion and Future Work

Several issues arise from the current work:

- The lower bound for computing the half-space depth rank of data points (and thus their rank contours) is $\Omega(n \log n)$, based on reduction from sorting, but all known algorithms are $O(n^2)$. We are seeking a method to order data points according to their depth in $o(n^2)$.
- Brodal and Jacob's [3] amortized $O(\log n)$ time per operation algorithm for dynamically updating the convex hull of a set of points might theoretically improve the running time of our algorithm. Preliminary study is underway.
- To the best of our knowledge, no algorithm for dynamically computing depth contours according to other depth measures exist. We are currently working on a dynamic scheme for regression depth and simplicial depth contours.
- Most real life experiments are high dimensional in nature. However, since existing static algorithm for computing depth contours for most data depth measures are exponential in the dimension dynamic approximation algorithms are needed.

References

1. P. K. Agarwal, M. Sharir, and E. Welzl. Algorithms for center and Tverberg points. In *Proc. 20th Annu. ACM Sympos. Comput. Geom.*, pages 61–67, 2004.
2. J. L. Bentley and D. Wood. An optimal worst case algorithm for reporting intersections of rectangles. *IEEE Trans. Comput.*, 29(7):571–577, 1980.
3. G. S. Brodal and R. Jacob. Dynamic planar convex hull. In *The 43rd Annual IEEE Symposum on Foundations of Computer Science*, pages 617–626, 2002.

[10] Note that there are actually two copies of s_i, along levels k' and $k'+1$.

4. 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.
5. T. M. Chan. An optimal randomized algorithm for maximum tukey depth. In *Proc. of 15th ACM-SIAM Symp. on Disc. Alg. (SODA04)*. ACM Press, 2004.
6. R. Cole, M. Sharir, and C. K. Yap. On k-hulls and related problems. *SIAM Journal on Computing*, 15(1):61–77, 1987.
7. T. H. Cormen, C. E. Leiserson, R. L. Rivest, and C. Stein. *Introduction to algorithms*. MIT Press, Cambridge, MA, second edition, 2001.
8. D. L. Donoho and M. Gasko. Breakdown properties of location estimates based on halfspace depth and projected outlyingness. *The Annals of Statistics*, 20:1803–1827, 1992.
9. K. Fukuda and V. Rosta. Exact parallel algorithms for the location depth and the maximum feasible subsystem problems. In *Frontiers in global optimization*, volume 74 of *Nonconvex Optim. Appl.*, pages 123–133. Kluwer Acad. Publ., 2004.
10. J. Hodges. A bivariate sign test. *The Annals of Mathematical Statistics*, 26:523–527, 1955.
11. S. Krishnan, N. H. Mustafa, and S. Venkatasubramanian. Hardware-assisted computation of depth contours. In *13th ACM-SIAM Symp. on Disc. Alg.*, 2002.
12. S. Langerman and W. Steiger. Optimization in arrangements. In *Proc. of the 20th Int. Symp. on Theor. Aspects of Computer Science (STACS 2003)*, 2003.
13. R. Liu. Private communications, May 2003. Dept. of Stat., Rutgers University.
14. R. Liu, J. Parelius, and K. Singh. Multivariate analysis by data depth: descriptive statistics, graphics and inference. *The Annals of Statistics*, 27:783–858, 1999.
15. J. Matoušek. Computing the center of planar point sets. *DIMACS Series in Disc. Math. and Theoretical Comp. Sci.*, 6:221–230, 1991.
16. 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.
17. M. H. Overmars and J. van Leeuwen. Maintenance of configurations in the plane. *J. Comput. System Sci.*, 23(2):166–204, 1981.
18. E. Rafalin and D. L. Souvaine. Data depth contours - a computational geometry perspective. Technical Report 2004-01, Tufts University, CS Dept., May 2004.
19. P. J. Rousseeuw, I. Ruts, and J. W. Tukey. The bagplot: A bivariate boxplot. *The American Statistician*, 53:382–387, 1999.
20. P. J. Rousseeuw and A. Struyf. Computing location depth and regression depth in higher dimensions. *Statistics and Computing*, 8:193–203, 1998.
21. I. Ruts and P. J. Rousseeuw. Computing depth contours of bivariate point clouds. *Comp. Stat. and Data Analysis*, 23:153–168, 1996.
22. A. Struyf and P. Rousseeuw. High-dimensional computation of the deepest location. Manuscript, Dept. of Math. and Comp. Sci., University of Antwerp, Belgium, 1999.
23. J. W. Tukey. Mathematics and the picturing of data. In *Proc. of the Int. Cong. of Math. (Vancouver, B. C., 1974), Vol. 2*, pages 523–531. Canad. Math. Congress, Montreal, Que., 1975.
24. M. van Kreveld, J. S. B. Mitchell, P. Rousseeuw, M. Sharir, J. Snoeyink, and B. Speckmann. Efficient algorithms for maximum regression depth. In *Proc. of the 15th Ann. Symp. on Comp. Geo.*, pages 31–40, New York, 1999. ACM.