The Centroid of Points with Approximate Weights

Marshall Bern[1] David Eppstein[2] Leonidas Guibas[3] John Hershberger[4] Subhash Suri[5] Jan Wolter[6]

[1] Xerox Palo Alto Research Center, 3333 Coyote Hill Rd., Palo Alto, CA 94304
[2] Dept. of ICS, U. of California, Irvine, CA 92717. Work done in part while visiting Xerox PARC and supported in part by NSF grant CCR-9258355
[3] Dept. of Computer Science, Stanford University, Stanford, CA 94705
[4] Mentor Graphics Corp., 1001 Ridder Park Dr., San Jose, CA 95131
[5] Dept. of Computer Science, Washington University, St. Louis, MO 63130
[6] Dept. of Computer Science, Texas A & M, College Station, TX 77843

Abstract. Let S be a set of points in $\mathbb{R}^d$, each with a weight that is not known precisely, only known to fall within some range. What is the locus of the centroid of S? We prove that this locus is a convex polytope, the projection of a zonotope in $\mathbb{R}^{d+1}$. We derive complexity bounds and algorithms for the construction of these "centroid polytopes".

1 Introduction

Suppose that $S = \{s_1, s_2, \ldots, s_n\}$ is a set of points in $\mathbb{R}^d$ and that each point s_i has an unknown nonnegative weight w_i that lies within a known range $[l_i, h_i]$. The *centroid* of S—also called its "center of mass" or "weighted average"—is the vector sum $\frac{1}{W}\sum_i w_i s_i$, where $W = \sum_i w_i$. We are interested in $C(S)$, the set of all possible centroids.

A generalization allows explicit upper and lower bounds on the total weight W, tighter than the implicit bounds $\sum_i l_i \le W \le \sum_i h_i$. Let $C_{L,H}(S)$ denote the locus of possible centroids of S with the additional constraint that $L \le W \le H$. Thus $C(S) = C_{0,\infty}(S)$.

In this paper we characterize the *centroid polytope* $C_{L,H}(S)$ and give algorithms to compute it. Section 2 establishes a connection between centroid polytopes and the classical notion of zonotopes. Section 3 proves that the maximum complexity of a centroid polytope is $\Theta(n^d)$, but only $\Theta(n^{d-1})$ for the special case of $C(S)$. Section 4 describes algorithms to compute centroid polytopes. In two dimensions, our algorithm has running time $O(n \log n + k \log^2 n)$, where k is the complexity of $C_{L,H}(S)$, the output. Section 5 improves the running time to $O(n \log n)$ for $C(S)$. This section introduces a new type of semi-dynamic convex hull data structure. Since $C(S)$ is simply the convex hull of S if each l_i is zero, this algorithm is asymptotically optimal.

An application of the centroid polytope in two dimensions arises in the mathematical modeling of oil spills [8]. Oil floating on water can be represented by a two-dimensional point set; weights represent the varying thickness of the oil layer, which is known only approximately. The total volume of the spill, however, may be known more accurately, hence the explicit bounds on the total weight W. Aircraft design provides a natural application in three dimensions: the centroid of a fully loaded airplane depends on the weights of its fuel, cargo, and passengers, and the flight characteristics of the plane in turn depends on the location of its centroid. Tighter bounds on total weight are again reasonable, since a single 300-pound passenger is fairly likely, though not an entire planeload of 300-pound passengers.

2 Connection to Zonotopes

As above, let $S = \{s_1, s_2, \ldots, s_n\}$ be points in Euclidean d-space $\mathbb{R}^d$, each with an unknown weight w_i that lies in the known range $[l_i, h_i]$, where $0 \leq l_i \leq h_i$. Let L and H be bounds on the total weight of S, with $0 \leq L \leq H$. The *centroid polytope* $C_{L,H}(S)$ is

$$\left\{ \frac{1}{W} \sum_{i=1}^{n} w_i s_i \;\middle|\; w_i \in [l_i, h_i] \text{ for } 1 \leq i \leq n, \;\; W = \sum_{i=1}^{n} w_i, \;\text{ and } L \leq W \leq H \right\}.$$

Günter Rote (personal communication) points out that a change of variables—"fractional programming"—lets us write the centroid polytope as the feasible region of a linear program. This proves that the centroid polytope is indeed a polytope. If we let $t = 1/\sum_i w_i$ and $v_i = w_i t$, then

$$C_{L,H}(S) \;=\; \left\{ \sum_{i=1}^{n} v_i s_i \;\middle|\; \sum_{i=1}^{n} v_i = 1, \;\; l_i t \leq v_i \leq h_i t, \;\text{ and } Lt \leq 1 \leq Ht \right\}.$$

An interesting special case, the *unconstrained centroid polytope*, is $C_{0,\infty}(S)$, which we also denote by $C(S)$. Figure 1 shows $C(S)$ for $S = \{(0,1), (0,0), (1,0)\}$, assuming that each weight range is simply $[\frac{1}{2}, 1]$. Each of the six sides of $C(S)$ corresponds to fixing one weight to $\frac{1}{2}$, another weight to 1, and letting the third weight vary between these two extremes. The varying weight belongs to the point hit by the continuation of the side. In general, each facet of an unconstrained centroid polytope in $\mathbb{R}^d$ lies in the same hyperplane as (at least) $d-1$ points of S. The weights of these points vary, whereas other weights are fixed to either their minimum or maximum values, depending on their halfspace.

Let v_i denote $(s_i, 1)$, the vector in $\mathbb{R}^{d+1}$ that agrees with s_i on its first d coordinates and has 1 as its last coordinate. Let

$$Z \;=\; \left\{ \sum_{i=1}^{n} w_i v_i \;\middle|\; w_i \in [l_i, h_i] \text{ for } 1 \leq i \leq n \right\}.$$

Thus Z is the Minkowski sum of the n line segments of the form $\{\, w_i v_i \mid w_i \in [l_i, h_i] \,\}$. (The *Minkowski sum* of sets A and B in $\mathbb{R}^{d+1}$ is $\{p+q \mid p \in A \text{ and } q \in B\}$.) The Minkowski sum of line segments is a special type of convex polytope called a *zonotope*.

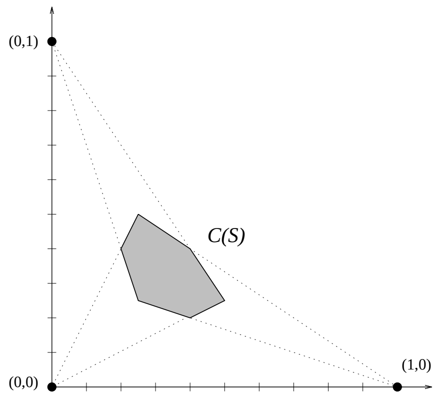

Fig. 1. The centroid polytope $C(S)$ for 3 points, each with weight range $[\frac{1}{2}, 1]$.

Edelsbrunner [3] writes, "it is very instructive to visualize the inductive construction of a zonotope". To add the ith segment z_i to the Minkowski sum, we sweep the current zonotope parallel to z_i and then take the swept volume as the new zonotope. If we assume that the ith segment runs "north-south", this step enlarges the current zonotope in $\mathbb{R}^{d+1}$ by stretching its "equator", a belt of faces of dimensions up to $d-1$, into a "zone" of faces of dimensions up to d. Faces of zonotopes are themselves lower-dimensional zonotopes.

There is a nice connection between zonotopes in $\mathbb{R}^{d+1}$ and arrangements in $\mathbb{R}^d$. We identify $\mathbb{R}^d$ with the space of directions (normal vectors) of hyperplanes tangent to zonotope Z. These directions most naturally live in projective d-space, so we adopt the projective view of an arrangement in $\mathbb{R}^d$, in which we regard antipodal pairs of unbounded cells as single cells (see [3], pp. 46–47). A hyperplane of the arrangement in $\mathbb{R}^d$ corresponds to the directions of hyperplanes tangent to a single zone of Z. A cell of the arrangement corresponds to an antipodal pair of vertices of Z. And, more generally, a k-face of the arrangement corresponds to an antipodal pair of $(d-k)$-faces of Z. Dimensions reverse, because a hyperplane through a $(d-k)$-face in $\mathbb{R}^{d+1}$ has k degrees of freedom.

Returning now to centroid polytopes, we see that $C_{L,H}(S)$ can be rewritten as

$$\left\{ \left(\frac{x_1}{x_{d+1}}, \frac{x_2}{x_{d+1}}, \ldots, \frac{x_d}{x_{d+1}} \right) \;\middle|\; (x_1, x_2, \ldots, x_d, x_{d+1}) \in Z \text{ and } L \leq x_{d+1} \leq H \right\}.$$

Notice that the last coordinate of points in Z measures total weight, that is, for the point $(x_1, x_2, \ldots, x_{d+1}) = \sum_i w_i v_i$, we have $x_{d+1} = \sum_i w_i$. Thus the condition $L \leq x_{d+1} \leq H$ restricts us to points of acceptable total weight, a section of Z lying between two hyperplanes normal to the $(d+1)$-st axis. Dividing the first d coordinates by the $(d+1)$-st then corresponds to a perspective projection of this section. (Only the "silhouette" matters; the interior of the projection is featureless.) In this projection, the viewpoint is the origin, and the "viewing screen" is the hyperplane $x_{d+1} = 1$. Figure 2 shows the zonotope connection for the example from Figure 1. Notice that in the special case of $C(S)$, the section is all of Z. We summarize with the following theorem.

Theorem 1. *The centroid polytope $C_{L,H}(S)$ is a perspective projection of the section of Z lying between the hyperplanes $x_{d+1} = L$ and $x_{d+1} = H$.* □

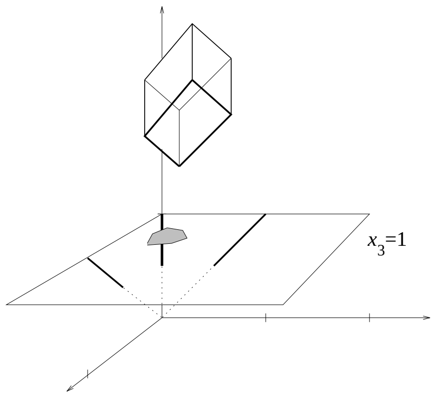

Fig. 2. The centroid polytope $C(S)$ is a perspective projection of a zonotope.

3 Complexity Bounds

Theorem 1 immediately implies that centroid polytopes are indeed convex polytopes. The zonotope connection also yields bounds on the complexity (total number of faces) of a centroid polytope. We present upper bounds first, then matching lower bounds.

Theorem 2. *The complexity of centroid polytope $C_{L,H}(S)$ is $O(n^d)$.*

Proof. A zonotope formed by n line segments in $\mathbb{R}^{d+1}$ has at most $O(n^d)$ faces [3]. Slicing Z by the hyperplanes $\sum_i w_i = L$ and $\sum_i w_i = H$ increases the complexity by at most a constant factor, and taking a perspective projection cannot increase the complexity, so altogether there are at most $O(n^d)$ faces in $C_{L,H}(S)$. □

Tighter bounds [3] follow from the correspondence between zonotopes in $\mathbb{R}^{d+1}$ and arrangements of hyperplanes in $\mathbb{R}^d$. The number of k-faces of zonotope Z for $0 \leq k \leq d$ is at most $2 \cdot \sum_{i=0}^{k} \binom{d-i}{k-i}\binom{n}{d-i}$. This implies that the number of facets of $C_{L,H}(S)$ is never more than $2(d+1)\binom{n}{d} + o(n^d)$.

We now turn to the special case of unconstrained centroid polytopes. Curiously, the projection of an entire zonotope has smaller asymptotic complexity than the projection of an arbitrary section. This is due to the following special property of a zonotope: its tangent space is not just any convex decomposition, but rather a hyperplane arrangement.

Theorem 3. *The complexity of the unconstrained centroid polytope* $C(S)$ *is* $O(n^{d-1})$. *In particular, the number of facets is at most* $2\binom{n}{d-1}$.

Proof. We exploit the zonotope-arrangement connection described above, in which zonotope Z corresponds to a hyperplane arrangement $\mathcal{A}$ in $\mathbb{R}^d$. Now consider all hyperplanes tangent to Z that pass through the origin in $\mathbb{R}^{d+1}$. The directions of these tangent hyperplanes form a convex body with boundary C in $\mathbb{R}^d$. Furthermore, C is linear within any face of $\mathcal{A}$. (From here on, we think of cells as faces too.) There is a one-to-one, dimension-reversing correspondence between faces of $C(S)$ and faces in the arrangement of piecewise-linear surfaces on C formed by the hyperplanes of $\mathcal{A}$. Thus we can count faces of $C(S)$ by counting the faces in this arrangement. By a standard argument, we can assume that $\mathcal{A}$ is a simple arrangement—meaning hyperplanes in general position—and that C does not pass through any vertex of $\mathcal{A}$; otherwise a small perturbation would increase the complexity.

We count the number of faces on C as follows. Each face is formed as a connected component of an intersection of a face in $\mathcal{A}$ with C. Since C is linear within each face of $\mathcal{A}$, each face of $\mathcal{A}$ gives rise to only one connected component, which necessarily includes a portion of the boundary of the face. Therefore we can "charge" each face of $\mathcal{A}$ cut by C to a lower-dimensional cut face, until we reach 1-faces (edges) of $\mathcal{A}$ that intersect C in single points (corresponding to facets of $C(S)$). Any line in $\mathcal{A}$ crosses C at most twice, and there are $\binom{n}{d-1}$ lines of $\mathcal{A}$, so the arrangement on C has at most $2\binom{n}{d-1}$ vertices and $C(S)$ has at most $2\binom{n}{d-1}$ facets. By general position, any 1-face in $\mathcal{A}$ has exactly $3^{d-1}-1$ neighboring cells of higher dimension, and hence each facet of $C(S)$ is charged at most 3^{d-1} times. Therefore there are at most $2 \cdot 3^{d-1}\binom{n}{d-1} = O(n^{d-1})$ faces in $C(S)$. □

We now move on to lower bounds, presenting them in reverse order from the upper bounds. The facet bound of Theorem 3 is tight. First consider the case $d = 2$. Let S be the vertices of a regular n-gon, and set all weight ranges to $[\frac{1}{n} - \epsilon, \frac{1}{n} + \epsilon]$ for $\epsilon > 0$ very small. Then $C(S)$ is a small $2n$-gon, with two sides for each s_i in S.

To generalize this bound to higher dimensions, let S be a set of points fairly evenly spaced on a sphere. For ϵ sufficiently small, $C(S)$ behaves like a point, which we may assume is in general position with respect to the points of S. In other words, through any $d-1$ points of S, some hyperplane misses $C(S)$, and hence two hyperplanes support $C(S)$. The two supporting hyperplanes each contain a facet, on which the weights of the $d-1$ points vary. Of course, this example also gives an $\Omega(n^{d-1})$ bound for the total complexity.

Figure 3(a) gives an $\Omega(n^2)$ lower bound example for $C_{L,H}(S)$ for the case $d = 2$, matching the bound given by Theorem 2. In this example, there are two rings of $\frac{n}{2}$ points, where $\frac{n}{2}$ is an odd number, forming regular $\frac{n}{2}$-gons. The inner-ring points, denoted S_I, lie just inside the vertices of the central $\frac{n}{2}$-gon formed by the diagonals of the outer-ring points, denoted S_O. Each point of S_O has range $\left[\frac{2}{n} - \epsilon, \frac{2}{n} + \epsilon\right]$, with $\epsilon > 0$ small enough that the centroid polytope $C(S_O)$

lies well inside the inner ring. Each point of S_I has range $[0, \frac{\epsilon}{n}]$. We now assert that $C_{L,H}(S)$, where $S = S_O \cup S_I$ and $L = H = 1 + \epsilon$, is a polygon with $\frac{n(n+1)}{2}$ sides.

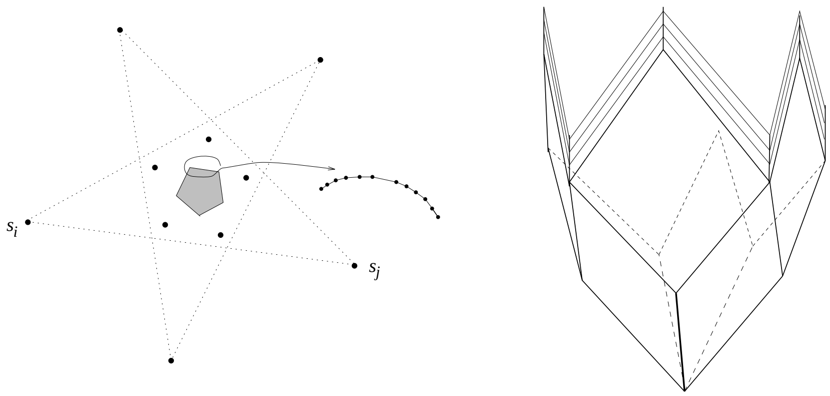

Fig. 3. (a) An $\Omega(n^2)$ example for $C_{L,H}(S)$. (b) Corresponding zonotope.

Why is this assertion true? First consider $C_{L,H}(S_O)$, the centroid polytope of the outer points, with the same restriction on the total weight. This polytope is an $\frac{n}{2}$-gon, shown shaded in Figure 3(a). Each side of $C_{L,H}(S_O)$ is defined by two points of S_O, whose weights vary along that side. For example, the weights of s_i and s_j vary along the side circled in Figure 3(a), while the weights of the top two points are fixed at $\frac{2}{n} + \epsilon$ and the weight of the bottom point is fixed at $\frac{2}{n} - \epsilon$.

Now step up to $C_{L,H}(S)$. The circled side is replaced by a chain of $n + 1$ sides. As we move along the first $\frac{n}{2}$ sides of this chain, weight $\frac{\epsilon}{n}$ transfers from s_i to each of the points of S_I in turn, in clockwise order around s_i. Along the middle side of the chain, weight $\frac{\epsilon}{2}$ transfers from s_i to s_j. Finally, along the last $\frac{n}{2}$ sides, weight transfers from S_I to s_j.

The zonotope corresponding to S looks something like a Ukrainian egg, as shown in Figure 3(b). This zonotope is the Minkowski sum of $\frac{n}{2}$ well-spaced long segments and $\frac{n}{2}$ nearly parallel short segments, and hence contains a zig-zag of $\Omega(n^2)$ faces that can all be cut by a horizontal plane.

By replacing S_O and S_I by points nearly equally spaced around concentric spheres, it should be possible to generalize this construction to give an $\Omega(n^d)$ bound in arbitrary dimension, but we have not worked out the details.

4 Output-Sensitive Algorithms

In this section, we briefly sketch algorithms for computing centroid polytopes. We start by showing how to compute $C_{L,H}(S)$ for $d = 2$. Because this polygon may have complexity anywhere from $O(1)$ to $O(n^2)$, our goal is an *output-sensitive* algorithm, meaning one with running time dependent on k, the number of sides of $C_{L,H}(S)$.

Let z be a rightmost vertex of $C_{L,H}(S)$. The next lemma shows that z is defined by separating the points of S into minimum- and maximum-weight subsets that can be separated by a vertical line ℓ. Line ℓ typically passes through a point of S that has intermediate weight.

Lemma 4. *There is a vertical line ℓ partitioning the plane into open halfplanes h^- and h^+, such that z is a linear combination $\frac{1}{W}\sum_i w_i s_i$ in which $w_i = l_i$ for each s_i in h^- and $w_i = h_i$ for each s_i in h^+. Further, if $z \in h^+$ the sum of all weights is L, and if $z \in h^-$ the sum of all weights is H.*

Proof. If s_i is to the left of s_j, with $w_i > l_i$ and $w_j < h_j$, we could increase w_j and decrease w_i causing the centroid to move further to the right. Therefore in the weights for z, the points taking their minimum value must lie to the left of those taking their maximum value. Vertical line ℓ separates these two sets, passing through the point(s) with intermediate weight. If no intermediate-weight point exists, we let ℓ be the separating line closest to z, which passes through either the rightmost minimum-weight or the leftmost maximum-weight point. Thus ℓ is always chosen to go through some particular point s_k. If the conditions on total weight in the last sentence of the lemma were not met, then by changing the weight of of s_k we could move the extremum. □

Lemma 5. *A rightmost vertex z can be found in $O(n)$ time.*

Proof. Assume that the points of S by increasing x-coordinate are $s_1, s_2, \ldots, s_n$. Let x_i denote the x-coordinate of s_i, and let $m = \lfloor \frac{n}{2} \rfloor$.

Using linear-time median finding, we compute x_m in time $O(n)$. We then assign point s_i weight l_i if $i < m$ and h_i if $i \geq m$, and compute the centroid c with this setting of weights. Let ℓ be the vertical line $x = x_m$. If the total weight is less than L (respectively, greater than H) we know that z lies to the left (right) of ℓ. If the total weight falls within these bounds, it is not hard to show that z lies on the same side of ℓ as c. Now if z lies to the left of ℓ, we collapse the set $S' = \{s_i \mid i \geq m\}$ to a single combined point—the centroid of S' with all weights set to their h_i values—and set its combined weight to $\sum_{i \geq m} h_i$. Similarly if z lies to the right of ℓ, we collapse the points s_i with $i < m$. In either case, we have reduced the problem to one with half as many points. Iterating this procedure gives an overall linear-time algorithm. □

We use a rotational sweep algorithm to compute $C_{L,H}(S)$. At all times, z is a point on the boundary of $C_{L,H}(S)$ that has a tangent parallel to sweep-line ℓ. Initially z is a rightmost vertex of $C_{L,H}(S)$ and ℓ is the line constructed in the proof of Lemma 5. As z moves counterclockwise around the boundary of $C_{L,H}(S)$, we maintain ℓ as the separating line that would be constructed if the plane were rotated to make z rightmost; thus ℓ always passes through a point of S.

In an event in the rotational sweep, line ℓ, pivoting around point s_i, crosses another point s_j. At such an event, we increase or decrease w_j (depending on whether s_j is in h^- or h^+) while making an offsetting change in w_i, until one of w_i and w_j runs into its minimum or maximum value. The one of s_i and s_j left with an intermediate value becomes the new pivot point; if both points reach extrema simultaneously, we break the tie by letting s_j be the new pivot point. If the input is in general position (no collinear triple in S and no "coincidental" sums of w_i's), these weight changes move z to a new vertex of $C_{L,H}(S)$.

Changing weights and recomputing z take only $O(1)$ time, so the overall running time of a step depends on how fast we can detect collisions between ℓ and points of S. One way to detect collisions between ℓ and S is to maintain the convex hull of each of the sets $S \cap h^+$ and $S \cap h^-$. The problem of detecting the next collision then reduces to finding the lines that pass through z and are tangent to these convex hulls. The fully dynamic convex hull data structure of Overmars and van Leeuwen [6] takes $O(\log^2 n)$ time per insertion or deletion, and supports such tangent-finding queries in time $O(\log n)$ apiece.

Theorem 6. *Polytope $C_{L,H}(S)$ in $\mathbb{R}^2$ can be computed in $O(n \log n + k \log^2 n)$ time for inputs in general position, where k denotes the output size.* □

We briefly remark on inputs with degeneracies. Coincidences among w_i's can be handled without increasing the running time, but if S includes many collinear points, ℓ might simultaneously cross many points that give only a single vertex of $C_{L,H}(S)$. This degrades the running time to $O((n + k + n^{2/3}k^{2/3}) \log^2 n)$, a bound stemming from the maximum number of incidences in any configuration of n points and k lines [7].

Before we step up to higher dimensions, let us take a dual view of the algorithm just presented. A rotational sweep through S corresponds to moving a point from left to right across the arrangement of lines $\mathcal{A}$ dual to S. When ℓ pivots around a point in the primal plane, its dual point p traces along a segment of $\mathcal{A}$. When ℓ crosses another point of S, p crosses another line of $\mathcal{A}$. Thus we can think of our previous algorithm as tracing a certain path in $\mathcal{A}$. Each successive vertex on the path can be found by a (primal) convex hull data structure, and corresponds to a vertex of the centroid polytope.

Now let us consider the case of arbitrary fixed dimension. It is straightforward to prove generalizations of Lemmas 4 and 5. Any extreme point of $C_{L,H}(S)$ corresponds to splitting S by a hyperplane and giving all points in one open half-space their minimum weights and all points in the other their maximum weights. Point set S dualizes to a hyperplane arrangement (not the same arrangement as in Theorem 3), and the set of splitting hyperplanes dualizes to a piecewise-linear surface in this arrangement. The features of this surface correspond one-to-one to faces of the centroid polytope; for example, vertices of the surface are the duals of splitting hyperplanes parallel to facets of $C_{L,H}(S)$. Thus we can compute the centroid polytope by visiting each face on the surface.

We use the following query to trace the surface: find all the cells sharing a vertex with a given cell. Moving to one of these neighboring cells corresponds to moving one of the input points from one side of the splitting hyperplane to the other. In the primal space, the query corresponds to searching for the first point hit by a hyperplane as it swings around a $(d-1)$-dimensional pivot. For points in general position, these queries, along with insertions and deletions, can be handled by an algorithm of Agarwal and Matoušek [1], which takes time $O(n^{1-2/(\lceil d/2\rceil)+\epsilon})$ per operation for any given $\epsilon > 0$. In particular, each operation takes time $O(n^{\epsilon})$ in the case $d = 3$.

Theorem 7. *Polytope $C_{L,H}(S)$ in $\mathbb{R}^d$ can be computed in $O((k+n)n^{1-2/(\lceil d/2\rceil)+\epsilon})$ time for inputs in general position and $O(kn)$ time for arbitrary inputs.* □

5 Optimal Algorithm for the Unconstrained Polytope

In this section, we show how to improve the algorithm above from $O(n \log^2 n)$ to $O(n \log n)$ for the case of $C(S)$ in $\mathbb{R}^2$. As above, let z be a rightmost vertex of $C(S)$, and let ℓ be a vertical line separating the points taking maximum and minimum weights. For the unconstrained centroid polytope $C(S)$, we may assume that line ℓ passes through z, rather than through a point of S. (We omit the easy proof.)

Now a generic step of the algorithm rotates ℓ counterclockwise about z until it hits a point of S. When ℓ hits a point s_i in half-plane h^+, the algorithm moves s_i from h^+ to h^-, changes its mass from h_i to l_i, and recomputes centroid z. Similarly, when ℓ hits a point $s_i \in h^-$, the algorithm moves s_i to h^+ and changes its mass to h_i. In either case, the new centroid lies on $\ell \cap C(S)$, counterclockwise of z. (Unlike before, special-position inputs do not cause any real problems.)

Smaller Sweeps. The improved algorithm splits the rotational sweep into four $90°$ sweeps. It is not hard to see that within a $90°$ sweep (in fact, even within a $180°$ sweep), no point of S can leave h^+ and subsequently re-enter it. This constraint allows the use of semi-dynamic, rather than fully dynamic, data structures.

Let us consider the $90°$ sweep in which ℓ starts out vertical and tangent to the right side of $C(S)$ and rotates to be horizontal and tangent to the top of $C(S)$. The starting and ending positions of ℓ partition the plane into quadrants, as shown in Figure 4.

During the sweep, the points of S in the four quadrants change sets as follows:

I) All points remain in h^+.
II) All points leave h^- and enter h^+.
III) Some points stay in h^-, but others (shaded) leave h^- and later re-enter it.
IV) All points leave h^+ and enter h^-.

Points in quadrants II and IV are easy to handle. For example, when a point s_i in quadrant IV leaves h^+, we delete it from the data structure holding the points of IV and change its weight from h_i to l_i, but we need not insert it into any other dynamic data structure, because it cannot again be crossed by ℓ within

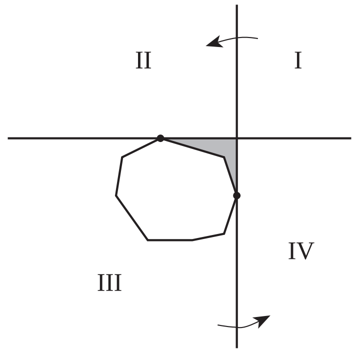

Fig. 4. Quadrants determined by starting and ending positions of ℓ.

this sweep. The deletions-only convex hull data structure of Hershberger and Suri [5] (see also Chazelle [2]) provides a suitable data structure for quadrants II and IV; it supports tangent-finding in $O(\log n)$ time and point deletion in $O(\log n)$ amortized time.

Points in quadrant III, specifically those in the shaded region, pose a difficulty. At any instant of the sweep, the points in quadrant III belong to three sets: (A) points that have not yet left h^-, (B) points now in h^+, and (C) points that have already returned to h^-. Set C does not need to be represented, because ℓ will not cross over these points again in this 90° sweep; set A can be represented using a deletions-only hull; set B, however, requires something more than a deletions-only hull, because this set is subject to both insertions and deletions.

In order to handle the points in set III(B), we modify the deletions-only hull so that it supports insertions at its left end, arbitrary deletions, and queries asking for the point closest to ℓ, each in amortized time $O(\log n)$. The modified data structure no longer supports arbitrary tangent-finding queries, but closest-point queries will suffice for our purposes.

Restricted-Insertion Convex Hull Data Structure. At any instant of the sweep, our data structure will contain the points in the trapezoid bounded by the initial and final positions of ℓ, the current position of ℓ, and the vertical line through z (the point of tangency between ℓ and $C(S)$). The points in this trapezoid are the points of set B that ℓ will hit first as it rotates around z. See Figure 5(a).

Because our new data structure is based on the deletions-only hull, we review the relevant properties of that data structure. Given a set of n points in the plane, the deletions-only structure represents the convex hull of an *active* subset of the points. All the points start out in the active state; they are inactivated by the client of the data structure in an arbitrary order. The data structure supports tangent queries in $O(\log n)$ time apiece and point deletions (inactivations) in $O(\log n)$ amortized time apiece.

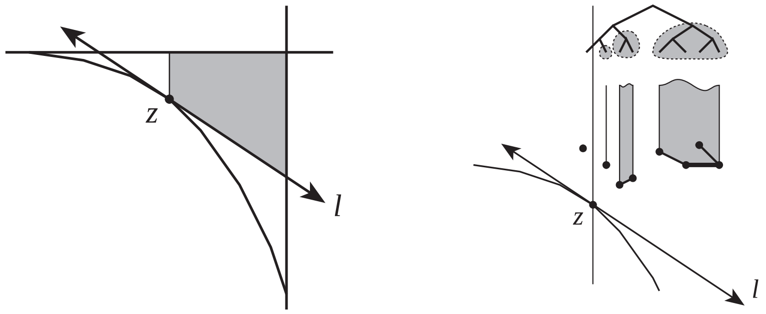

Fig. 5. The restricted-insertion data structure (a) stores the points in the shaded trapezoid and (b) maintains the convex hulls of maximal-subtree subsets.

The data structure is based on a complete binary tree T that stores all the points (in our case, all the points in quadrant III) at the leaves in x-coordinate order. Each node of T represents the lower convex hull of the active points in its subtree as a linked list. The lower convex hull at a node is the concatenation of two sublists, one from each child node, linked by the lower common tangent of the two child hulls. When a point becomes inactive, the hull is recomputed at its leaf and all of its ancestor nodes. Each active point has a level: $level(s_i)$ is the highest level in T at which s_i appears on a convex hull. The level of an active point increases monotonically. When a point is inactivated (deleted), recomputing convex hulls takes time proportional to the number of point-level changes. Thus the total work of maintaining the convex hull is $O(n \log n)$.

We now modify this deletions-only structure to allow insertions of a restricted sort. The key ideas are the following:

1. Instead of maintaining a single lower convex hull, we maintain hulls for a forest F with $O(\log n)$ connected components, namely the set of maximal subtrees of T to the right of z. The leaves of F correspond to the points of S in quadrant III whose x-coordinates lie between z and the initial position of ℓ. For any active point s_i in this interval, the maximum value of $level(s_i)$ is the height of its containing tree in F. See Figure 5(b).
2. We insert new active points at the left end of the forest F. When the vertical line through z passes a point s_i of quadrant III, above ℓ, F changes: the nodes of T whose leftmost descendant is s_i become nodes in F. We then compute lower convex hulls at these new nodes of F.
   New hulls can be computed in time proportional to the number of point-level changes (see [5]). The basic idea is to compute the common tangents between sub-hulls by marching along their lists, traversing only points whose level is about to increase. A key observation is that $level(s_j)$ does not decrease for any point s_j. Thus the central invariant of the deletions-only structure is preserved, and the total time needed for convex hull maintenance is $O(n \log n)$.
3. For each tree in F, we maintain its closest point to ℓ, in other words, the hull

vertex with a supporting line parallel to ℓ. Let E be the set of all $O(\log n)$ closest points.

Set E is dynamic in two ways: when a point of E is deleted, a new point takes its place; and when ℓ rotates to a new position, the closest point of a tree may move to the right along the hull. Since there are at most n deletions and n moves to the right, the total number of changes to E during the $90°$ sweep is $O(n)$. We now show that these two types of changes can each be handled in $O(\log n)$ amortized time.

When a point $s_j \in E$ is deleted (inactivated), a new closest point must be computed. Let s_i and s_k be the neighbors of s_j on its convex hull. After s_j is deleted, the new closest point lies on the new hull between s_i and s_k. The points strictly between s_i and s_k are those whose level has increased, so the cost of finding the new extremum is dominated by the cost of recomputing the convex hull.

When ℓ rotates to a new position ℓ', we identify the point $s_i \in E$ that has the minimum-slope (most vertical) edge to its right. There are only $O(\log n)$ candidates, so this step clearly takes only $O(\log n)$ time. If the slope to s_i's right is greater than that of ℓ', then E does not change. Otherwise we replace s_i in E by its right neighbor and repeat the process until we reach the slope of ℓ'.

The Complete Algorithm. We now describe how to use these data structures to implement one step of the $90°$ sweep. Of course, the complete algorithm consists of four $90°$ sweeps. We use deletions-only convex hull data structures for the points in quadrants II, IV, and III(A) and a restricted-insertion convex hull data structure for III(B). One step has the following substeps:

1. Find the first point s_i in II, IV, or III(A) that is hit by ℓ as it rotates around z. Let ℓ' be the line through z and s_i.
2. Rotate ℓ toward ℓ' while maintaining E. The next change in E occurs when ℓ's slope equals that of the minimum slope to the right of a point in E. Before making the change to E, check whether ℓ would cross over any point of the current set E before reaching this slope. If not, change E and continue rotating ℓ. If so, reset s_i to be the point of E that ℓ hits and and stop rotating ℓ at the slope determined by z and s_i.
3. Delete s_i from its data structure, move it from h^+ to h^- or vice versa, and recompute z. Update the forest F in the restricted-insertion data structure to include all the points in III whose x-coordinates lie between the old and the new positions of z.

We have now proved the following result.

Theorem 8. *The planar, unconstrained centroid polytope $C(S)$ can be computed in $O(n \log n)$ time.* □

6 Conclusions

In this paper, we have characterized and given algorithms for centroid polytopes. Centroid polytopes simultaneously generalize the centroid and the convex hull of a point set.

Yet another interpretation of the centroid polytope $C_{L,H}(S)$ in $\mathbb{R}^d$ involves the following system of homogeneous linear equations:

$$\begin{pmatrix} x_{11} & x_{12} & \dots & x_{n1} & z_1 \\ x_{12} & x_{22} & \dots & x_{n2} & z_2 \\ \vdots & \vdots & & \vdots & \vdots \\ x_{1d} & x_{2d} & \dots & x_{nd} & z_d \\ 1 & 1 & \dots & 1 & 1 \end{pmatrix} \begin{pmatrix} w_1 \\ w_2 \\ \vdots \\ w_n \\ -W \end{pmatrix} = \begin{pmatrix} 0 \\ 0 \\ \vdots \\ 0 \\ 0 \end{pmatrix}$$

Now if each $s_i = (x_{i1}, x_{i2}, \dots, x_{id})$, then the centroid polytope is exactly the set of $z = (z_1, z_2, \dots, z_d)$ that leaves a solvable system under the linear constraints $l_i \leq w_i \leq h_i$ and $L \leq W \leq H$. With this non-geometric formulation, it makes sense to generalize centroid polytopes to allow negative weights. If the total weight of S can equal zero, the centroid polytope becomes unbounded. In this case, the centroid polytope is best computed by dividing it into pieces containing centroids of positive and negative total weight.

References

1. P.K. Agarwal, D. Eppstein, and J. Matoušek. Dynamic half-space reporting, geometric optimization, and minimum spanning trees. In *Proc. 33rd IEEE Foundations of Computer Science*, Pittsburgh, 1992, 80–89.
2. B. Chazelle. On the convex layers of a planar set. *IEEE Trans. Inf. Theory*, IT-31 (1985), 509–517.
3. H. Edelsbrunner. *Algorithms in Combinatorial Geometry*, Springer-Verlag, 1987.
4. H. Edelsbrunner and E.P. Mücke. Simulation of simplicity, a technique to cope with degenerate cases in geometric computations. *ACM Trans. Graphics* 9 (1990) 66–104.
5. J. Hershberger and S. Suri. Applications of a semi-dynamic convex hull algorithm. *BIT* 31 (1992), 249–267.
6. M. Overmars and J. van Leeuwen. Maintenance of configurations in the plane. *J. Comput. Syst. Sci.* 23 (1981), 166–204.
7. E. Szemerédi and W.T. Trotter. Extremal problems in discrete geometry. *Combinatorica* 3 (1983), 381–392.
8. J. Tsao, J. Wolter, and H. Wang. Model-based understanding of uncertain observational data for oil spill tracking. In *Proc. 3rd Int. Conf. on Industrial Fuzzy Control & Intelligent Systems*, Houston, 1993, 149–154.

This article was processed using the LaTeX macro package with LLNCS style