International Journal of Computational Geometry & Applications

An Output-Sensitive Convex Hull Algorithm for Planar Objects

Franck Nielsen
INRIA, BP93, 06902 Sophia-Antipolis cedex, France
and
University of Nice, Parc Valrose, France
e-mail: Franck.Nielsen@sophia.inria.fr

and

Mariette Yvinec
INRIA, BP93, 06902 Sophia-Antipolis cedex, France
and
CNRS, URA 1376, Laboratoire I3S
Rue Albert Einstein, 06560 Valbonne, France
e-mail: Mariette.Yvinec@sophia.inria.fr

Received (received date)
Revised (revised date)
Communicated by Editor's name

ABSTRACT

A set of planar objects is said to be of type m if the convex hull of any two objects has its size bounded by $2m$. In this paper, we present an algorithm based on the marriage-before-conquest paradigm to compute the convex hull of a set of n planar convex objects of fixed type m. The algorithm is output-sensitive, i.e. its time complexity depends on the size h of the computed convex hull. The main ingredient of this algorithm is a linear method to find a *bridge*, i.e. a facet of the convex hull intersected by a given line. We obtain an $O(n\beta(h,m)\log h)$-time convex hull algorithm for planar objects. Here $\beta(h,2)=O(1)$ and $\beta(h,m)$ is an extremely slowly growing function. As a direct consequence, we can compute in optimal $\Theta(n\log h)$ time the convex hull of disks, convex homothets, non-overlapping objects. The method described in this paper also applies to compute lower envelopes of functions. In particular, we obtain an optimal $\Theta(n\log h)$-time algorithm to compute the upper envelope of line segments.

Keywords: Computational geometry, Convex hull, Upper Envelope, Output-sensitive algorithms, Marriage before conquest.

1. Introduction

Convex hull has been of main interest for years in computational geometry. Many articles have considered the case of points where general paradigms have been used or purposely developed. Worst-case optimal space and time algorithms have been established for sets of points in dimension d[1,2,3]. However, the convex hull of n points in general position in a d-dimensional space ranges from the d-simplex with

2^{d+1} faces to maximal polytopes of size $O(n^{\lfloor \frac{d}{2} \rfloor})$ (see Ref.4,5). We are interested in designing algorithms whose time complexity depends on both the input and output sizes: the so-called output-sensitive algorithms.

Optimal output-sensitive algorithms for points are known only in dimensions 2 and 3 by the time being. D.G. Kirkpatrick and R. Seidel[6,7] gave the first optimal output-sensitive algorithm in dimension 2. Their algorithm is based on a new paradigm: *marriage-before-conquest*. H. Edelsbrunner and W. Shi[8] gave an $O(n \log^2 h)$-time algorithm to compute the h facets of the convex hull of n points of $\mathbb{E}^3$ using the same paradigm. K.L. Clarkson and P.W. Shor[9] described an output-sensitive randomized algorithm for computing the convex hull of a set of points in dimension 3. The expected complexity of their algorithm is optimal. Their algorithm uses as a basic primitive the deterministic algorithm of D.G. Kirkpatrick and R. Seidel and was derandomized later on by B. Chazelle and J. Matoušek[10].

In higher dimensions ($d \geq 4$), for a long time the best known solution was the algorithm of R. Seidel[11] which after an $O(n^2)$-time preprocessing step , finds the facets of a convex hull in a shelling order at a logarithmic cost per facet. The preprocessing step was reduced later[12,13] on to $O\left(n^{2-\frac{2}{\lfloor \frac{d}{2} \rfloor+1}+\epsilon}\right)$ for any $\epsilon > 0$. Recently, T. Chan et al.[14] have investigated the case of points in four dimensions, achieving an $O((n+h)\log^2 h)$-time algorithm for computing the convex hull of a set of n points where h denotes the output-size. In higher dimensions, T. Chan[15] realized many improvements on the convex hull computations and related problems, combining the gift-wrapping method of D.R. Chand and S.S. Kapur[16] and G.F. Swart[17] with recent results on data structures for ray shooting queries in polytopes (developed by P.K. Agarwal et J. Matoušek[18] and refined by J. Matoušek and O. Schwarzkopf[19]).

Computing the convex hull of a set of curved objects has been much less investigated. Computing the convex hull of a single planar object bounded by curves has been carefully studied[20,21,22] and several authors have generalized linear-time algorithms for computing the convex hull of a simple planar polygon[23,24,25,26]. In the case of a family of n planar disks, optimal $\Theta(n \log n)$-time convex hull algorithms have been designed[27,28].

We consider the following problem: given a collection $\mathcal{O} = \{O_1, ..., O_n\}$ of n convex objects, compute in an *output-sensitive* manner the convex hull $\mathcal{CH}(\mathcal{O})$, i.e. the smallest convex object containing $\mathcal{O}$. In the general case, the usual way to compute the convex hull of $\mathcal{O}$ is to compute the lower and the upper envelopes of $\mathcal{O}$ and to consider the unique object bounded by these envelopes. Then, one can apply to this single planar object one of the convex hull algorithms mentioned above. A classical output-sensitive algorithm to compute the convex hull $\mathcal{CH}(\mathcal{O})$ is Jarvis's march[29] which runs in $O(nh)$ where h denotes the output-size. In this paper, we generalize the marriage-before-conquest approach of R. Seidel and D.G Kirkpatrick[7] in the case of planar objects.

Independently, T. Chan[15] gave a simple algorithm for computing the convex hull of a set of planar points. His algorithm can be adapted to handle the case of convex

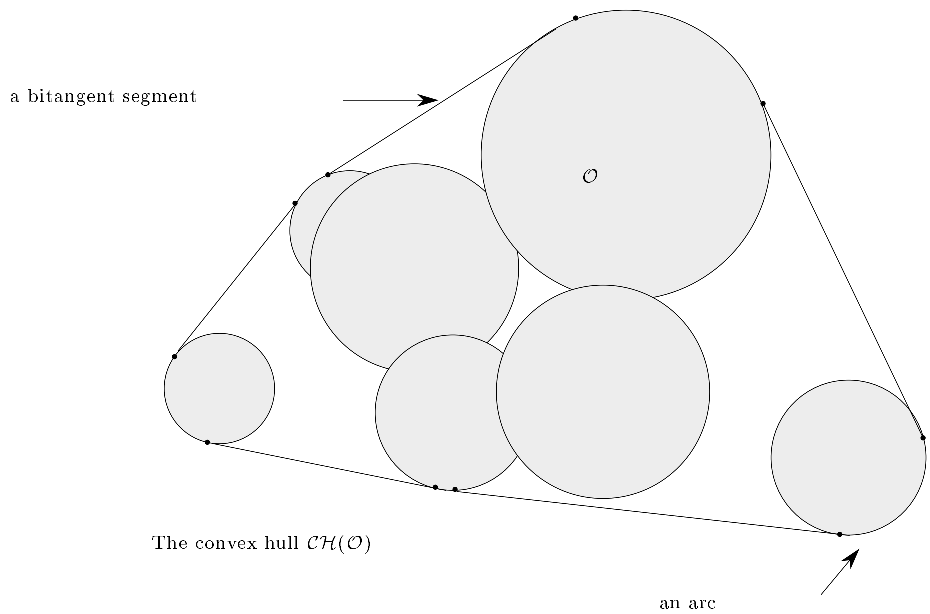

Figure 1: A convex hull of disks ($m = 2$).

objects (although this is not described in Ref.15) within the same time bounds. Nevertheless, our algorithm is different and is interesting in its own right. It relies on an $O(n \log h + \delta h)$-time algorithm to compute the convex hull of n objects of fixed type m such that any object can be colored with a value in $\{1, ..., \delta\}$ and objects of a same color do not intersect pairwise[a], where h denotes the output-size. Thus, we obtain immediately an optimal $\Theta(n \log h)$ algorithm if we consider that the objects satisfy the hard-sphere model[30] or have only a few intersections (in that case, our algorithm is simpler than T. Chan's one[15]). Moreover, we solve the problem of computing in linear time a bridge, i.e. a facet of the convex hull intersecting a given oriented line. In the general case, we first transform our original set of objects $\mathcal{O}$ into another set $\mathcal{T}$ such that $\mathcal{CH}(\mathcal{O}) = \mathcal{CH}(\mathcal{T})$ with the objects in $\mathcal{T}$ being colored with at most $\delta = \lceil \frac{n}{h} \rceil$ objects and apply our basic algorithm.

Computing the convex hull of general planar convex objects differs from the case of points because the convex hull of two points p_1 and p_2 is the straight segment $[p_1 p_2]$ whereas the complexity of the convex hull of two planar convex objects O_i and O_j depends on the nature of these objects. We call *arc* a maximal piece of the boundary of $\mathcal{CH}(\mathcal{O})$ that is included in the boundary ∂O_i of an object O_i of $\mathcal{O}$. The boundary of $\mathcal{CH}(\mathcal{O})$ is an alternating sequence of arcs and *bitangent segments* (Figure 1). In the following, the arcs of $\mathcal{CH}(\mathcal{O})$ and its bitangent segments are called *facets*. In this paper, we shall consider sets of convex objects with the property that the convex hull of any two objects has bounded complexity (if the objects are non-convex and have fixed descriptive complexity, we can first compute their

[a]In particular, if any object intersect at most γ others then $\delta \leq \gamma + 1$.

convex hulls in linear time). More precisely, a set of objects $\mathcal{O}$ is said to be of *type* m if the convex hull of any two objects of $\mathcal{O}$ has at most m arcs (or $2m$ facets). Let $|\mathcal{CH}(\mathcal{O})|$ denote the size of the convex hull $\mathcal{CH}(\mathcal{O})$, i.e. the number of facets (convex arcs and bitangent segments) of its boundary $\partial\mathcal{CH}(\mathcal{O})$. Then, $\mathcal{O}$ is of type m if $\forall\ i,j \in [1,n]$, $|\mathcal{CH}(O_i, O_j)| \leq 2m$. For example, points have type 1, disks, convex homothets and non-overlapping objects have type 2, ellipsis have type 4, etc. Note that if $\mathcal{O}$ is of type m then the boundaries of any two convex objects of $\mathcal{O}$ cannot intersect in more than m points. Moreover, if q denotes the maximum number of intersection points between the boundaries of two distinct convex objects of $\mathcal{O}$, then $m = \max\{2, q\}$. Moreover, if the objects are bounded by closed convex curves then m is even.

Throughout this paper, we suppose that the type of set $\mathcal{O}$ is fixed. Moreover, each object in $\mathcal{O}$ is assumed to have a bounded descriptive size (for instance, the boundary of each object is a curve of bounded degree) : in particular, this means that we can find in constant time the two supporting lines of an object with a given slope. Furthermore, we assume that the convex hull of two objects in $\mathcal{O}$ can be computed in constant time, where the constant depends on the type m.

This paper is organized as follows:

In section 2, we recall the complexity of the convex hull of n objects of type m.

In section 3, we first extend to the case of a set of convex objects of type m, the algorithm of D.G. Kirkpatrick and R. Seidel[6,7] to compute a *bridge*, i.e. the facet of the convex hull intersecting a given oriented line (subsection 3.1). Our algorithm is based on the searching-and-pruning paradigm and achieve an optimal $\Theta(n)$ time complexity to compute a bridge of a set of n convex objects of type m. Then, we present the scheme of the *marriage-before-conquest* approach (subsection 3.2). This scheme amounts to computing a bridge at a given oriented line, uses this bridge to filter the objects and to divide the problem into two independent sub-problems which are recursively solved. Finally, we refine the *marriage-before-conquest* algorithm in the case of a set partitionned into k subsets of non-overlapping objects, i.e. a set $\mathcal{O} = \cup_{i=1}^{k}\mathcal{P}_i$ where each $\mathcal{P}_i,\ i \in [1,k]$, is a collection of non-overlapping objects (subsection 3.3). The time complexity of the algorithm is $O(n\log h + hk)$. This algorithm is used as a basic primitive in the final algorithm. We also derive an $O(n\log h + \delta h)$-time algorithm to compute the convex hull of n objects of fixed type m where h denotes the output-size and δ is the maximal number of objects that an object can intersect.

In section 4, we describe the algorithm in the general case. We design an $O(n\beta(h,m)\log h)$-time convex hull algorithm where n is the number of objects, h denotes the output-size and $\beta(h,m)$ is a very slowly growing function related to the maximum length $\lambda(n,m)$ of a (n,m)-Davenport-Schinzel sequence[31,32,33]. More precisely, $\beta(h,2) = O(1)$ and $\beta(h,m) = O(2^{\alpha(h)^{c_m}})$ if $m > 2$, where c_m is an integer depending on m and $\alpha(\cdot)$ is the functional inverse of Ackermann's function. The algorithm is close to optimal with respect to both the input and output sizes since $\Omega(n\log h)$ is a lower bound[7].

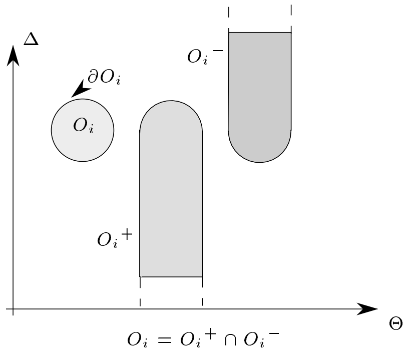

Figure 2: Defining ${O_i}^+$ and ${O_i}^-$.

In section 5, we adapt the method for computing upper envelopes of functions intersecting pairwise in at most m points and obtain an $O(n\beta(h, m+2)\log h)$-time algorithm. We improve slightly the algorithm in case of k-intersecting *generalized segments*, i.e. partially defined functions that intersect pairwise in at most k points. In that case, we obtain an $O(n\beta(h, k+1)\log h)$-time algorithm which is $\Theta(n\log h)$ for line segments.

Finally, in section 6, we conclude and give several guidelines for future research.

2. Complexity of the Convex Hull of Convex Objects of Type m

In this section, we first examine the complexity of the convex hull $\mathcal{CH}(\mathcal{O})$ where $\mathcal{O}$ is a set of planar convex objects of type m.

Let us consider p^+ the point with coordinates $(0, +\infty)$ (resp. p^- the point with coordinates $(0, -\infty)$). Let us call *upper convex hull* (respectively *lower convex hull*) of $\mathcal{O}$ the convex hull $\mathcal{CH}^+(\mathcal{O}) = \mathcal{CH}(\mathcal{O}, p^-)$ (resp. $\mathcal{CH}^-(\mathcal{O}) = \mathcal{CH}(\mathcal{O}, p^+)$). We denote by O_i^+ (resp. O_i^-) the object $\mathcal{CH}^+(O_i, p^-)$ (resp. $\mathcal{CH}^-(O_i, p^+)$) (see Figure 2). Let $\mathcal{O}^+ = \{O_i^+ | O_i \in \mathcal{O}\}$ and $\mathcal{O}^- = \{O_i^- | O_i \in \mathcal{O}\}$. Then, $\mathcal{CH}^+(\mathcal{O}) = \mathcal{CH}(\mathcal{O}^+)$, $\mathcal{CH}^-(\mathcal{O}) = \mathcal{CH}(\mathcal{O}^-)$ and $\mathcal{CH}(\mathcal{O}) = \mathcal{CH}(\mathcal{O}^+) \cap \mathcal{CH}(\mathcal{O}^-)$.

We bound the complexity of the convex hull of convex objects of type m as follows:

Theorem 1 *In the worst-case, the complexity of the convex hull of n planar convex objects of type m is bounded by $4\lambda(n, m)$ where $\lambda(n, m)$ is the maximum length of an (n, m)-Davenport-Schinzel sequence*[31,32,33].

Proof. Since the boundaries $\partial\mathcal{CH}(\mathcal{O}^+)$ and $\partial\mathcal{CH}(\mathcal{O}^-)$ of respectively $\mathcal{CH}(\mathcal{O}^+)$ and $\mathcal{CH}(\mathcal{O}^+)$ coincides at their extremities, the size $|\mathcal{CH}(\mathcal{O})|$ of the convex hull is at most $|\mathcal{CH}(\mathcal{O}^+)| + |\mathcal{CH}(\mathcal{O}^-)|$. We therefore focus on the upper bound of $|\mathcal{CH}(\mathcal{O}^+)|$. Since the convex hull $\mathcal{CH}(\mathcal{O}^+)$ is an alternating sequence of bitangent segments and arcs, we count the maximal number of arcs that can appear on the boundary of $\mathcal{CH}(\mathcal{O}^+)$, i.e. $\partial\mathcal{CH}(\mathcal{O}^+)$.

To each object O_i^+ of $\mathcal{O}^+$ we associate its *supporting* function $f_i(\cdot)$ defined as follows: $f_i(\theta)$ is defined over $[0, \pi]$ as the y-coordinate of the intersection point $p_i(\theta)$

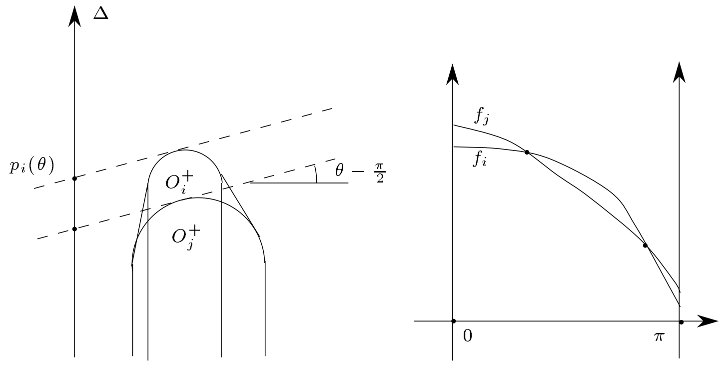

Figure 3: Duality between the boundary of the upper convex hull $\partial\mathcal{CH}^+(\mathcal{O})$ and the upper envelope $E_{\mathcal{F}}$ of $\mathcal{F}$.

Table 1: Lower and upper bounds of $\lambda(n,m)$. $\alpha(n)$ is the functional inverse of Ackermann's function.

$m \backslash \lambda(n,m)$	Lower Bound Ω	Upper Bound O
1	n	n
2	$2n-1$	$2n-1$
3	$\Omega(n \times \alpha(n))$	$O(n \times \alpha(n))$
4	$\Omega(n \times 2^{\alpha(n)})$	$O(n \times 2^{\alpha(n)})$
$m = 2s+1$	$\Omega(n \times 2^{O(\alpha(n)^{s-1})})$	$O(n \times \alpha(n)^{O(\alpha(n)^{s-1})})$
$m = 2s+2$	$\Omega(n \times 2^{O(\alpha(n)^{s})})$	$O(n \times 2^{O(\alpha(n)^{s})})$

of the y-axis Δ with the supporting line of O_i^+ with slope θ (see Figure 3). We get a collection $\mathcal{F} = \{f_i | 1 \leq i \leq n\}$ of n functions that are totally defined over the range $[0,\pi]$. As it is well known from duality, the upper convex hull $\mathcal{CH}(\mathcal{O}^+)$ is isomorph to the upper envelope $E_{\mathcal{F}}$ of $\mathcal{F}$, i.e. the pointwise maximum of the f_i's. An arc of $E_{\mathcal{F}}$ corresponds to an arc of $\partial\mathcal{CH}(\mathcal{O}^+)$, a vertex of $E_{\mathcal{F}}$ corresponds to a bitangent segment of $\partial\mathcal{CH}(\mathcal{O}^+)$.

Since $\mathcal{O}^+$ is of type m, two supporting functions f_i and f_j intersect in at most m points.

Therefore, the number of arcs of $\partial\mathcal{CH}(\mathcal{O}^+)$ is bounded by the maximal length $\lambda(n,m)$ of an (n,m)-Davenport-Schinzel sequence[31,32,33]. Since the upper convex hull is an alternating sequence of arcs and bitangent segments, we get $|\mathcal{CH}(\mathcal{O}^+)| \leq 2\lambda(n,m)$. It follows that $|\mathcal{CH}(\mathcal{O})| \leq 4\lambda(n,m)$. $\square$

Computing upper envelopes of real functions (defined over $\mathbb{R}$) that can mutually intersect in at most q points is a problem which has been extensively studied[34,35,36]. A divide-and-conquer approach yields an $O(\lambda(n,q)\log n)$-time complexity algorithm.

Therefore, one can compute the upper envelope of the 'dual' functions f_i's defined by objects in $\mathcal{O}^+$ in time $O(\lambda(n,m)\log n)$ if set $\mathcal{O}^+$ is of type m. Alternatively, the convex hull $\mathcal{CH}^+(\mathcal{O})$ can also be computed using the randomized incremental construction of Clarkson[37] and Clarkson and Shor[38] in expected running time $\tilde{O}(\lambda(n,m)\log n)$.

3. Computing the Convex Hull of Colored Families of Convex Objects

In this section, we first show in subsection 3.1 how to compute in linear time a bridge, i.e. a facet of the convex hull intersected by a given ray. Then, we present in subsection 3.2 the marriage-before-conquest paradigm applied to the case of objects. Finally, we refine the algorithm in subsection 3.3 in the case of colored families of convex objects, i.e. families that can be partitionned into monochromatic subsets of pairwise non-intersecting objects.

3.1. Bridge of a Convex Hull

3.1.1. Definition and notations

The bridge of $\mathcal{O}$ at Δ is the unique facet of $\mathcal{CH}^+(\mathcal{O})$ that is intersected by Δ. The *bridge* at Δ is either an arc or a bitangent segment of $\mathcal{CH}(\mathcal{O})$. This section is devoted to the computation of the bridge at an oriented line Δ of a set of planar convex objects of fixed type m. The bridge facet is easily determined if one knows the line which supports $\mathcal{CH}(\mathcal{O})$ at the point $\Delta \cap \partial\mathcal{CH}(\mathcal{O})$ where Δ intersects the boundary of $\mathcal{CH}(\mathcal{O})$. Indeed, if this line is a supporting line for at least two objects in $\mathcal{O}$ then the bridge is a bitangent segment whereas if this line is a supporting line of a single object $O_i \in \mathcal{O}$ then the bridge is an arc of $\mathcal{CH}(\mathcal{O})$ included in ∂O_i. In both cases, the two endpoints of the bridge can be found in linear time once this supporting line is known. Thus, we focus on the determination of the supporting line of $\mathcal{CH}(\mathcal{O})$ at point $\Delta \cap \partial\mathcal{CH}(\mathcal{O})$. Hereafter, this line is called the *supporting line* of the bridge at Δ.

Computing the supporting line of the bridge at Δ of n convex objects is a generalized linear program[39,40,41,42] and can therefore be computed by a randomized algorithm in expected $\tilde{O}(n)$ time. Moreover, we can use the derandomized algorithm of B. Chazelle and J. Matoušek[43] in order to obtain a linear deterministic algorithm. Hereafter we give a more direct algorithm to compute in linear time the bridge at Δ. D.G. Kirkpatrick and R. Seidel[7] gave a deterministic optimal $\Theta(n)$ algorithm that computes a bridge for a set of n points using a searching-and-pruning procedure. We extend this algorithm to convex objects of fixed type m.

In order to follow the steps of this searching-and-pruning method, we first extend the main theorem of D.G. Kirkpatrick and R. Seidel[7]'s algorithm for computing the bridge of points to the case of convex objects that can be separated by a line parallel to Δ. Then, we introduce the *vertical decomposition* in order to obtain convenient sets of convex objects. We finally give the overall algorithm and analyze its time complexity.

3.1.2. The case of convex objects

Without loss of generality, consider that the direction of Δ is the direction of the y-axis, called the *vertical* axis. We denote by $x(p)$ the abscissa of point p. Kirkpatrick and Seidel proved the following lemma for a set $\mathcal{O}$ of points:

Lemma 2 (3.2, pp. 291 Ref.7) *Let p, q be a pair of points of $\mathcal{O}$ with $x(p) < x(q)$, let s_b be the slope of the supporting line of the bridge of $\mathcal{O}$ at Δ and let s be the slope of the straight line through p and q. If $s > s_b$ then p cannot be a point of the bridge of $\mathcal{CH}(\mathcal{O})$ at Δ. If $s < s_b$ then q cannot be a point of the bridge at Δ.*

Two objects O_1 and O_2 are said to be *x-separated* if they can be separated by a line parallel to Δ. Note that x-separated objects can be ordered along the x-axis. In the following, we note $x(O)$ the x-range of an object O, i.e. the projection of O onto the x-axis. Let (O_1, O_2) be a pair of x-separated objects. If O_1 is to the left of an oriented vertical line separating O_1 and O_2 then we note $x(O_1) < x(O_2)$ and O_1 (resp. O_2) is called the left (resp. right) object of the pair (O_1, O_2). An object $O \in \mathcal{O}$ is said to participate to the bridge at Δ if the supporting line of $\mathcal{CH}^+(\mathcal{O})$ at $\Delta \cap \partial\mathcal{CH}^+(\mathcal{O})$ is a supporting line of object O. We extend lemma 2 to the case of x-separated objects. Observe that if O_1 and O_2 are a pair of x-separated object, the boundary of the upper convex hull $\mathcal{CH}^+(O_1, O_2)$ has a unique bitangent segment.

Lemma 3 *Let (O_1, O_2) be a pair of x-separated objects with $x(O_1) < x(O_2)$, let s be the slope of the unique bitangent segment of $\partial\mathcal{CH}^+(O_1, O_2)$ and let s_b be the slope of the bridge of $\mathcal{O}$ at Δ. If $s > s_b$ then the left object O_1 of the pair cannot participate to the bridge at Δ. If $s < s_b$ then the right object O_2 of the pair cannot participate to the bridge at Δ.*

Proof. We only give the proof in case of $s > s_b$ (the other case is obtained by symmetric considerations). Let I be the intersection point between a separating line Δ' parallel to Δ and the affine hull L of the unique bitangent segment of $\partial\mathcal{CH}^+(O_1, O_2)$ (see Figure 4). Let s be the slope of L and define L' as the line passing through I with slope s_b. Let $L_1(s_b)$ be the tangent line to O_1 with slope s_b. $L_1(s_b)$ and L' are parallel lines (and can therefore be ordered along Δ'). Because of the convexity of O_1, if $s > s_b$ then $y(L' \cap \Delta') = y(L \cap \Delta') \geq y(L_1(s_b) \cap \Delta')$ and $L_1(s_b)$ is below L'. But the contact point $T_2(s) = L \cap O_2$ is strictly above L' if $s > s_b$. Therefore, point $T_2(s)$ is above $L_1(s_b)$ so that O_1 cannot participate to the bridge at Δ $\square$

3.1.3. Vertical decomposition

The vertical decomposition will give rise to a set of x-separated convex objects.

Let $\mathcal{O} = \{O_1, ..., O_n\}$ be a set of n planar convex objects of type m and $\mathcal{CH}^+(\mathcal{O})$ its upper convex hull. We decompose this upper convex hull by striping $\mathcal{CH}^+(\mathcal{O})$. To stripe $\mathcal{CH}^+(\mathcal{O})$, we draw through each vertex of $\partial\mathcal{CH}^+(\mathcal{O})$ a line parallel to Δ. These parallel lines induce a decomposition of each object O_i of $\mathcal{O}$ into sub-objects, called *tiny* objects in the following (see Figure 5). We only keep the tiny objects whose boundary participates to the boundary of the convex hull $\partial\mathcal{O}$. Note that each tiny object is defined from a single object and two vertical lines, and that two

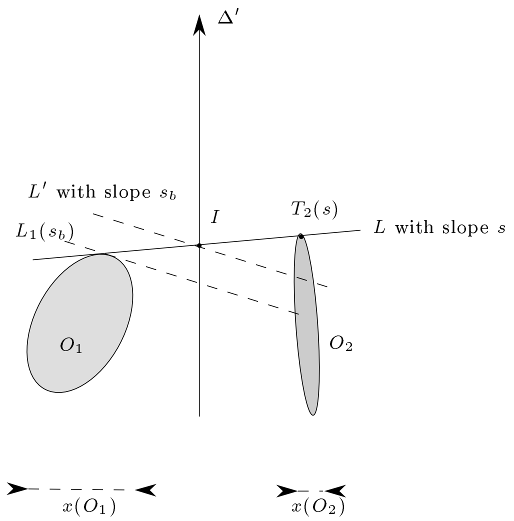

Figure 4: Discarding O_1 when $s > s_b$.

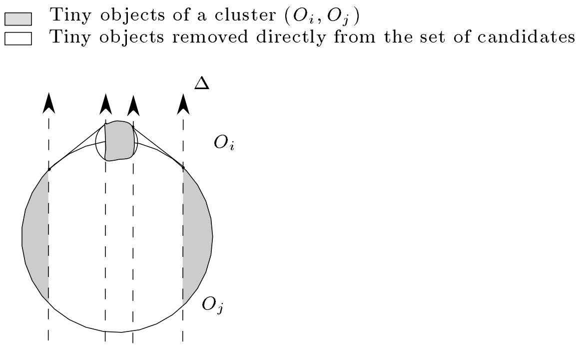

Figure 5: Vertical decomposition of a pair of convex objects of type 2.

tiny objects arising from this decomposition are x-separated.

3.1.4. Algorithm

Let L be the line which supports $\mathcal{CH}(\mathcal{O})$ at point $\Delta \cap \partial\mathcal{CH}^+(\mathcal{O})$. Line L is a supporting line for some of the objects in $\mathcal{O}$. Our goal is to select from $\mathcal{O}$ the objects that touch L.

We pair up the convex objects of $\mathcal{O}$ into pairs (O_i, O_j). For each pair (O_i, O_j), we first compute $\mathcal{CH}^+(O_i, O_j)$, the upper convex hull of O_i and O_j and apply the vertical decomposition to $\mathcal{CH}^+(O_i, O_j)$. We discard from this decomposition all the tiny objects that do not have an arc of $\partial\mathcal{CH}^+(O_i, O_j)$ on their boundaries. Indeed, as these tiny objects do not appear on the boundary of $\mathcal{CH}^+(O_i, O_j)$, they also do not appear on the boundary of $\mathcal{CH}^+(\mathcal{O})$ and therefore cannot participate to any bridge of $\mathcal{CH}^+(\mathcal{O})$ (Figure 1).

We call *cluster* the set of remaining tiny objects of a pair. As each pair is of type at most m since $\mathcal{O}$ is of type m, we can deduce that there are at most $(m+1)$ convex tiny objects in a cluster. Then, we pair up all the tiny objects within a cluster into at most $\lfloor \frac{m+1}{2} \rfloor$ tiny pairs. Note that we pair up only the tiny objects within a cluster since, as they are x-separated, they have type 2 whereas two tiny objects of different clusters have type m. In the following, we shall use lemma 3 to reduce the number of tiny objects in the clusters. As the slope s_b of the supporting line of $\mathcal{CH}(\mathcal{O})$ at $\Delta \cap \partial\mathcal{CH}^+(\mathcal{O})$ is of course unknown, the trick is to resolve tests like $s < s_b$ or $s > s_b$ using transitivity. A cluster is said to be *reduced* if it has only one remaining tiny object. The algorithm consists in an *initial step* where we pair up the objects in order to get non-reduced clusters and several rounds of *selecting* and *clustering* where we eliminate, round after round, the tiny objects. We describe the algorithm below:

Initial step. We pair up the objects and compute for each pair its vertical decomposition. This step gives rise to clusters of tiny objects. If an object O_i of a pair (O_i, O_j) is included in O_j then the cluster generated by this pair is reduced. In that case, we discard O_i and pair O_j again until all clusters are non-reduced.

Selecting and Clustering. A round:

- **Selecting.**
  - Pair up the tiny objects inside each cluster.
  - Compute the median s_m of the slopes of the bitangent segments of pairs of tiny objects (use the median algorithm of Blum et al.[44]).
  - Let $\mathcal{O}'$ be the subset of objects which contribute to the current collection of tiny objects. Then, find the supporting line $L(s_m)$ of $\mathcal{CH}^+(\mathcal{O}')$ with slope s_m and locate the contact points $L(s_m) \cap \mathcal{CH}^+(\mathcal{O}')$ with respect to Δ. To find the supporting line of $\mathcal{CH}^+(\mathcal{O}')$ with a given slope s_m, we find the object(s) which maximize $\max_{i=1..|\mathcal{O}'|}\{y_i - s_m x_i\}$ where point (x_i, y_i) is the contact point of the supporting line of O_i with slope s_m. In general, there is one or two such objects and therefore one or two contact points but there can be possibly more.
  - Discard tiny objects:
    * If there are contact points located in both sides of Δ then $s_m = s_b$. The supporting line of $\mathcal{CH}(\mathcal{O})$ at $\Delta \cap \partial\mathcal{CH}^+(\mathcal{O})$ is fully determined by a tiny pair whose bitangent segment has slope s_m.
    * If all the contact points are located at the left side of Δ then $s_m > s_b$. We consider all tiny pairs with slope $s \geq s_m$ and discard the left tiny object t_1 of these pairs.

* If all the contact points are located at the right side of Δ then $s_m < s_b$. We consider all tiny pairs with slope $s \leq s_m$ and discard the right tiny object t_2 of these pairs.

- **Clustering.** This stage is required in order to obtain a set of non-reduced clusters for the next round. For each reduced cluster, we consider the original object $O \in \mathcal{O}$ which gave rise to the single tiny object of this cluster. We pair up these objects and compute for each pair its vertical decomposition. This step gives rise to new clusters. If an object O_i of a pair (O_i, O_j) is included in O_j then the cluster generated by this pair is reduced. In that case, we discard O_i and pair O_j again until all clusters are non-reduced. In the next round, we consider these new clusters together with the non-reduced clusters remaining from the last selecting step.

The algorithm halts whenever it finds a tiny pair whose slope equals the slope s_b of the supporting line of the bridge at Δ or if it remains only one tiny object. In the former case, the bridge is a bitangent segment and we find its two endpoints in linear time. In the latter case, there are two subcases: either the remaining tiny object does not intersect Δ and $\mathcal{CH}(\mathcal{O}) \cap \Delta = \emptyset$ or it defines the object whose boundary contains the bridge arc. In the latter subcase, the endpoints of the arc can be found in linear time.

3.1.5. Complexity analysis

Theorem 4 *The above algorithm computes the bridge of a set of n planar convex objects of fixed type m in optimal $\Theta(n)$ time and storage.*

Proof. Once we know the supporting line of the bridge at Δ, we can determine, in linear time, the nature of the bridge (arc or segment) and compute its two endpoints in linear time for a fixed type m. We therefore focus on the analysis of the searching-and-pruning algorithm.

Let l and k be respectively the number of tiny objects and the number of clusters (they are all non-reduced) present at the beginning of some round of the selecting and clustering steps. Then, we denote by $c(l,k)$ the cost of the algorithm from that stage. Let l' and k' be respectively the number of tiny objects and the number of non-reduced clusters at the end of that round, i.e. after the clustering step. We have the following recursive equation:

$$c(l,k) = \begin{cases} O(1) & \text{if } k \leq 1 \\ \alpha l + \beta(k-k') + c(l',k') & \text{otherwise,} \end{cases} \tag{1}$$

where α and $\beta = \beta_m$ are some constants (β depending on m).

Let r denote the number of clusters reduced during the *selecting* phase. Since we pair up the r reduced clusters to create new non-reduced clusters, we have $k' \leq k - r + \lfloor \frac{r}{2} \rfloor$. In the second part of equation (1), αl is the cost of the *selecting* phase, $\beta \frac{r}{2} \leq \beta(k-k')$ the cost of the *clustering* phase of the round and $c(l',k')$ the total cost of the remaining rounds. Each vertical decomposition of the convex hull of two objects costs $\beta_m = \beta$ if type m is fixed.

If $k = 1$ there is only one cluster of tiny objects. We can compute the convex hull of the at most two objects which give rise to the set of its tiny objects in time $O(1) = \gamma_m = \gamma$ if type m is fixed.

Let $\mathcal{S}_1$ ($|\mathcal{S}_1| = l'_1$) be the set of remaining tiny objects after the selecting phase of the current round and $\mathcal{S}_2$ ($|\mathcal{S}_2| = l'_2$) the set of tiny objects created during the clustering phase. Let $\mathcal{S}'$ ($|\mathcal{S}'| = l'|$) be the set of tiny objects at the beginning of the next round, i.e. $\mathcal{S}' \subseteq \mathcal{S}_1 \cup \mathcal{S}_2$. Clearly, we have $l' \leq l'_1 + l'_2$.

We prove that $l'_1 \leq \frac{5}{6}l$:

Assume that among the k clusters present at the beginning of the current round, k_o clusters have an odd number of tiny objects (say the first k_o clusters) and thus remain with an unpaired tiny object after the pairing of tiny objects while the $(k - k_o)$ other clusters have all their tiny objects paired. Finally, denote by a_i the number of pairs of tiny objects in the i-th cluster. We have the following equation:

$$l = \sum_{i=1}^{k_o}(2a_i + 1) + \sum_{i=k_o+1}^{k} 2a_i = \big(\sum_{i=1}^{k} 2a_i\big) + k_o \tag{2}$$

The *selecting* process removes a tiny object from half of the tiny pairs, so that $l'_1 \leq l - \frac{1}{2}\sum_{i=1}^{k} a_i$. Using equation (2), we obtain $l'_1 \leq \frac{3}{4}l + \frac{k_o}{4}$. As the number of tiny objects l is at least $2k + k_o$ and k_o ranges over $[0, k]$, we have $l \geq 3k_o$. Thus, $l'_1 \leq \frac{5l}{6}$.

Now, consider the number of created tiny objects during the *clustering* step. Clearly, $l'_2 \leq (k - k')(m + 1)$. l' is therefore upper-bounded by $\frac{5}{6}l + (k - k')(m + 1)$.

Then, it follows by induction on vector (k, l) ordered lexicographically that $c(l, k) \leq \delta l + (\beta + \delta(m + 1))k + \gamma$ for any $\delta \geq 6\alpha$.

Initially, $l \leq \lceil \frac{n}{2} \rceil(m + 1)$ and $k \leq \lceil \frac{n}{2} \rceil$ so that the complexity of all the rounds of the selecting and clustering step is upper bounded by $O(n)$ for any fixed type m.

The cost of the initial step is also $O(n)$. Thus we obtain an $\Theta(n)$-time algorithm to compute the bridge $\square$

3.2. Marriage-Before-Conquest Algorithm

In this section, we present the marriage-before-conquest strategy to compute the convex hull $\mathcal{CH}(\mathcal{O})$ of a set of n convex objects $\mathcal{O}$. We consider w.l.o.g. the computation of the upper convex hull since the boundary of $\mathcal{CH}(\mathcal{O})$ is obtained in $O(1)$ time from the boundaries of $\mathcal{CH}^+(\mathcal{O})$ and $\mathcal{CH}^-(\mathcal{O})$. Each object in $\mathcal{O}$ has two supporting lines parallel to the y-axis, called *walls*. Each wall is oriented as the y-axis. Let $\mathcal{W}$ be the set of walls and denote by $|\mathcal{W}| = w = 2n$ its cardinality. Let $\mathcal{R}$ be a range, i.e. an interval on the x-axis. We define a *slab* as the portion of the euclidean plane $\mathbb{E}^2$ between two lines parallel to Δ. The upper convex hull $\mathcal{CH}^+(\mathcal{O})$ can be described as an x-ordered sequence of facets. The following algorithm $\mathsf{MarriageBeforeConquest}(\mathcal{W}, \mathcal{O}, \mathcal{R})$ computes a subsequence $\mathcal{MBC}(\mathcal{W}, \mathcal{O}, \mathcal{R})$ of the facets of $\mathcal{CH}^+(\mathcal{O})$ included in the slab $\mathcal{B} = \mathcal{R} \times (-\infty, +\infty)$.

Termination. If $w = 0$ then $\mathcal{MBC}(\mathcal{W}, \mathcal{O}, \mathcal{R}) = \emptyset$. Return $\mathcal{MBC}(\mathcal{W}, \mathcal{O}, \mathcal{R})$.

Divide. Find the median W_m of the walls $\mathcal{W}$. Split $\mathcal{W}$ into two balanced subsets $\mathcal{W}'_1 = \{W \in \mathcal{W} | x(W) \leq x(W_m)\}$ and $\mathcal{W}'_2 = \{W \in \mathcal{W} | x(W) \geq x(W_m)\}$.

Merge. Compute the bridge b at the median oriented line W_m.

Filter. Let $\mathcal{W}_1$ (respectively $\mathcal{W}_2$) be the subset of the walls of $\mathcal{W}'_1$ (resp. $\mathcal{W}'_2$) that do not intersect b. Let w_1 and w_2 denote respectively the cardinalities of sets $\mathcal{W}_1$ and $\mathcal{W}_2$. Let x_b^+ and x_b^- be respectively the abscissæ of the right and left endpoints of b. Let $\mathcal{R}_1 = \mathcal{R} \cap (-\infty, x_b^-)$, $\mathcal{B}_1 = \mathcal{R}_1 \times (-\infty, +\infty)$, $\mathcal{R}_2 = \mathcal{R} \cap (x_b^+, +\infty)$ and $\mathcal{B}_2 = \mathcal{R}_2 \times (-\infty, +\infty)$. Let $\mathcal{O}_1$ (resp. $\mathcal{O}_2$) be the set of objects in $\mathcal{O}$ that intersect slab $\mathcal{B}_1$ (resp. slab $\mathcal{B}_2$). Compute $\mathcal{W}_1$, $\mathcal{W}_2$, $\mathcal{R}_1$, $\mathcal{R}_2$, $\mathcal{B}_1$, $\mathcal{B}_2$, $\mathcal{O}_1$ and $\mathcal{O}_2$. Let $n_1 = |\mathcal{O}_1|$ and $n_2 = |\mathcal{O}_2|$.

Conquest. Call recursively $\mathsf{MarriageBeforeConquest}(\mathcal{W}_1, \mathcal{O}_1, \mathcal{R}_1)$ and $\mathsf{MarriageBeforeConquest}(\mathcal{W}_2, \mathcal{O}_2, \mathcal{R}_2)$ and return the ordered sequence of facets $\mathcal{MBC}(\mathcal{W}_1, \mathcal{O}_1, \mathcal{R}_1) \cup \{b\} \cup \mathcal{MBC}(\mathcal{W}_2, \mathcal{O}_2, \mathcal{R}_2)$.

We denote by $c(n, w, h)$ the complexity of the algorithm MarriageBeforeConquest running inside range $\mathcal{R}$ if there are w walls in $\mathcal{B}$, n objects intersecting $\mathcal{B}$ and h computed facets of $\mathcal{CH}^+(\mathcal{O})$ in $\mathcal{B}$. Each computed facet is intersected by at least one wall of $\mathcal{W}$, so that $h \leq |\mathcal{W}|$. We obtain the following equation:

$$c(n, w, h) = \begin{cases} O(n) & \text{if } h \leq 1 \\ c(n_1, w_1, h_1) + c(n_2, w_2, h_2) + O(n) & \text{otherwise} \end{cases} \tag{3}$$

The algorithm ensures that $w_1 + w_2 \leq w$ and $w_1, w_2 \leq \lceil \frac{w}{2} \rceil$ but it does not control n_1 nor n_2 ($n_1, n_2 \leq n$) so that its worst-case running time is $O(nh)$. At the end, we are left with an x-ordered alternating sequence of computed facets and *empty* slabs (i.e. slabs that do not contain any wall of $\mathcal{W}$). We can find the whole upper convex hull using Jarvis's algorithm inside each empty terminal slab. In the following section, we study a special case where we can bound the number of objects that participate to the upper convex hull inside a slab (parameter n_1 and n_2 of equation (3)). We will use this "special" case as a basic primitive in the final algorithm.

3.3. The Case of a Non-Overlapping Partition

Let $\mathcal{O}$ be a set of n objects of fixed type m. If we know that a partition $\uplus_{i=1}^k \mathcal{P}_i$ of set $\mathcal{O}$ of fixed type m into k subsets such that each subset $\mathcal{P}_i$, for $i \in [1, k]$, is a set of non-overlapping convex objects then we can derive an $O(n \log h + hk)$-time complexity algorithm to find the convex hull of $\mathcal{O}$. This result holds, for example if $\mathcal{O}$ is a set of non-overlapping convex objects, since in that case $k = 1$ and $m = 2$. Let $\mathcal{B}$ be a vertical slab where we want to compute the upper convex hull. Among the objects of $\mathcal{O}$ intersecting $\mathcal{B}$, we distinguish two mutually exclusive categories:

Category 1: The objects that have a wall inside $\mathcal{B}$.

Category 2: The objects that intersect $\mathcal{B}$ but do not have a wall inside $\mathcal{B}$: these objects are called the *spanning* objects hereafter.

Algorithm MarriageBeforeConquest is slightly modified, taking into account these two categories of objects inside each slab $\mathcal{B}$ (with associated range $\mathcal{R}$), as follows:

- We bound the number of objects to consider in slab $\mathcal{B}$ by selecting among the spanning objects, at most one object of each family $\mathcal{P}_i$. Indeed, $\mathcal{R}$ is included in the x-range of each spanning object. Thus, the spanning objects which belong to a given family $\mathcal{P}_i$ can be ordered along any vertical line included in $\mathcal{B}$ and only the topmost object can contribute to the upper convex hull in $\mathcal{B}$.
- We stop the recursive calls as soon as $w \leq k$ and run Jarvis's march in each resulting slab on the set of objects $\mathcal{O}_{\mathcal{B}}$ relevant for this slab. We have $|\mathcal{O}_{\mathcal{B}}| \leq 2k$ since there are at most k spanning objects and k objects of category 1. This Jarvis's march is initialized from the computed facet which intersects the rightmost vertical line limiting $\mathcal{B}$ and stopped when the leftmost vertical line limiting $\mathcal{B}$ is reached.

Theorem 5 *Let $\mathcal{O}$ be a set of n planar convex objects of fixed type m partitionned into k subsets of non-overlapping convex objects, then the convex hull of $\mathcal{O}$ can be computed in $O(n \log h + hk)$ time, where h is the size of the convex hull of $\mathcal{O}$.*

Proof. Let $c(n, w, h)$ denote the complexity of the above algorithm. We have:

$$c(n, w, h) = \begin{cases} O(hk) & \text{if } w \leq k \\ c(n_1, w_1, h_1) + c(n_2, w_2, h_2) + O(n) & \text{otherwise} \end{cases} \tag{4}$$

with $w_1 + w_2 \leq w$, $w_1, w_2 \leq \lceil \frac{w}{2} \rceil$, $n_1 \leq w_1 + k$ and $n_2 \leq w_2 + k$ since we keep, in each sub-slab $\mathcal{B}_1, \mathcal{B}_2$, at most k spanning objects and there are at most w_1 objects (resp. w_2 objects) that have a wall in slab $\mathcal{B}_1$ (resp. $\mathcal{B}_2$).

We consider the recursive time complexity equation (4) and link parameters n and w using the inequality: $n \leq w + k$; thus $c(n, w, h) \leq c(w + k, w, h)$ and from now on, we simply note $c(w, h)$ for $c(w + k, w, h)$. Bounding n by $w + k$ in equation (4), we obtain:

$$c(w, h) = \begin{cases} \alpha h k & \text{if } w \leq k \\ c(w_1, h_1) + c(w_2, h_2) + \beta(w + k) & \text{otherwise} \end{cases} \tag{5}$$

where α and β are some constants.

Note that $w_1 \leq \lceil \frac{w}{2} \rceil$, $w_2 \leq \lceil \frac{w}{2} \rceil$ and $h = h_1 + h_2 + 1$. We prove by induction on w that $c(w, h) \leq \gamma(w \log h + kh)$ for a suitable constant γ:

- If $w \leq k$ then $c(w, h) = \alpha k h$ by equation (5). So that $c(w, h) \leq \gamma(w \log h + kh)$ if $\gamma \geq \alpha$.
- Suppose that $c(w', h) = \gamma(w' \log h + kh)$ for $0 \leq w' < w$ and consider $c(w, h)$ with $w > k$. Using equation (5), it follows that:

$$c(w, h) = \gamma(w_1 \log h_1 + kh_1 + w_2 \log h_2 + kh_2) + \beta(w + k)$$

with $w_1, w_2 \leq \frac{w}{2}$ and $h_1 + h_2 + 1 = h$. Note that $\log(h_1 h_2)$ is maximized for $h_1 = h_2 = \frac{h-1}{2}$, thus:

$$c(w,h) \leq \gamma(\frac{w}{2} \log \frac{h^2}{4} + kh) + \beta(w+k),$$

$$c(w,h) \leq \gamma(w \log h + kh - w) + \beta(w+k).$$

But $k < w$ by hypothesis, so that

$$c(w,h) \leq \gamma(w \log h + kh) + (2\beta - \gamma)w,$$

and $c(w,h) \leq \gamma(w \log h + kh)$ for suitable $\gamma \geq 2\beta$.

This proves that $c(w,h) \leq \gamma(w \log h + kh)$ for constant $\gamma = \max\{2\beta, \alpha\}$. Initially, $w = 2n$ (each of the n initial objects has two walls) so that the complexity of the algorithm is $O(n \log h + kh)$. □

As a direct consequence, we obtain a $\Theta(n \log h)$-time algorithm for computing the convex hull of non-overlapping convex objects. Note that our algorithm requires to know the partition of $\mathcal{O}$ into subsets of non-overlapping objects. We can define for a family of n objects its *intersection graph* $\mathcal{G}$ as follows: for each object $O_i \in \mathcal{O}$ we create a node and two different nodes are linked iff their corresponding objects intersect. If δ is the maximum degree of the nodes of $\mathcal{G}$, we know from the graph theory that there exists a partition of $\mathcal{O}$ into p subsets of non-overlapping objects such that $p \leq \delta + 1$. We can slightly modify our algorithm in order to take into account the paramater δ without knowing a partition into subsets of non-overlapping objects: choose a vertical line inside the slab and select from the spanning objects the object O that has the uppermost intersection point with that line. Then, we discard all the spanning objects that do not intersect O (this means that we only keep the spanning objects intersecting O). It is trivial to prove that all the spanning objects that do not intersect O are below O and therefore cannot participate to the upper convex hull. Thus, we obtain an $O(n \log h + \delta h)$-time algorithm to compute the upper convex hull of n objects of fixed type m where δ is the maximal number of intersection of any object with the others. For example, we can compute the convex hull of n hard-disks[30] in $\Theta(n \log h)$ (a family of disks in the hard-sphere model has the property that each disk intersects at most $O(1)$ others, i.e. $\delta = O(1)$). We also obtain an optimal $\Theta(n \log h)$-time algorithm if $\delta \leq O(\frac{n \log n}{\lambda(n,m)})$.

Note that the above algorithm computes the upper convex hull inside each terminal slab using Jarvis's march. If we skip this last phase of the algorithm, we are left with a subsequence of the facets of the convex hull. There is a terminal slab intersecting at most $2k$ objects between each pair of consecutive facets in the subsequence. Then, the algorithm is called PartialMBC and its complexity is still $O(n \log h + kh)$ but h is, here, the number of computed bridges (and not the total number of facets of the upper convex hull).

4. The General Case

In this section, we first present a convex hull algorithm assuming we know a good estimate h_e of the output-size h. To obtain a good estimate of the output-size, we have to compare the size h of the convex hull with some given value p; we show in section 4.2 how to perform such comparisons. The final algorithm is given in section 4.3.

4.1. Given an Estimate of the Output-Size

Let h_e be an estimate of the output-size $h = |\mathcal{CH}^+(\mathcal{O})|$. The algorithm includes two steps: the first step computes from $\mathcal{O}$ a set $\mathcal{T}$ of objects partitionned into non-overlapping subsets such that $\mathcal{CH}^+(\mathcal{O}) = \mathcal{CH}^+(\mathcal{T})$. Then, in a second step, we apply the marriage-before-conquest algorithm of section 3.3 on $\mathcal{T}$. We describe the algorithm below:

Grouping. Group the n objects into $\lceil \frac{n}{h_e} \rceil$ groups of size h_e. For each group, we compute the vertical decomposition of the convex hull of its objects. Thus, we obtain from the groups a set $\mathcal{T}$ of $O(\lceil \frac{n}{h_e} \rceil \lambda(h_e, m))$ tiny objects partitionned into $\lceil \frac{n}{h_e} \rceil$ subsets of at most $\lambda(h_e, m)$ non-overlapping tiny objects.

Marriage-before-conquest. Let $\mathcal{W}$ be the set of walls corresponding to the tiny objects of $\mathcal{T}$. Let $\mathcal{R}$ be the x-range $(-\infty, +\infty)$.
Return $\mathsf{MarriageBeforeConquest}(\mathcal{W}, \mathcal{T}, \mathcal{R})$ (see algorithm section 5).

Let us now analyze the complexity of the two steps:

Grouping. Computing the vertical decomposition of the upper convex hull of a group of h_e objects requires $O(\lambda(h_e, m) \log h_e)$ time: we first compute the upper envelope of the h_e objects by a divide-and-conquer algorithm and then run a walk-like convex hull algorithm on the resulting upper envelope[20]. The upper envelope has worst-case size $\lambda(h_e, m)$. Thus, the time required to compute the vertical decomposition of a group is $O(\lambda(h_e, m) \log h_e)$. Since there are $\lceil \frac{n}{h_e} \rceil$ groups, the total time complexity of this first step is $O(n \frac{\lambda(h_e, m)}{h_e} \log h_e)$.

Marriage-before-conquest. We run the marriage-before-conquest algorithm of section 3.3 onto the set of $O(\frac{n\lambda(h_e, m)}{h_e})$ tiny objects partitionned into $\lceil \frac{n}{h_e} \rceil$ subsets of non-overlapping objects. From the complexity analysis of section 3.3, this step requires $O(n \frac{\lambda(h_e, m)}{h_e} \log h + \frac{nh}{h_e})$ time.

The total time complexity of the algorithm is therefore $O\left(n \frac{\lambda(h_e, m)}{h_e} (\log h_e + \log h) + \frac{nh}{h_e}\right)$. Thus, if $h_e = h$ then the time required to compute the convex hull $\mathcal{CH}^+(\mathcal{O})$ is $O(n \frac{\lambda(h, m)}{h} \log h)$.

4.2. Comparing the Output-Size with a Given Value

In order to find a good estimate of h, we will need to determine if our current estimate (say p) is good (this means that p roughly equals to h) or not, i.e. to answer tests like $p > h$, $p = h$ or $p < h$.

Lemma 6 *There exists a deterministic algorithm that given an integer p, answers whether $h > p$ or not in $O(n\frac{\lambda(p,m)}{p}\log p)$ time.*

Proof. We design an algorithm which does not differ too much from the algorithm of section 4.1: the idea is to group the objects into $\lceil \frac{n}{p} \rceil$ groups of p objects and then run the marriage-before-conquest algorithm PartialMBC on the $\lceil \frac{n}{p} \rceil$ subsets of non-overlapping tiny objects resulting from the vertical decomposition of each group. Finally, we bound the number of facets computed by Jarvis's marches inside the terminal slabs. More precisely, we run Jarvis's march inside each "terminal" slab (a slab with at most $\frac{n}{p}$ walls) until we have computed a total of $\min\{p, h\}$ facets. We describe the algorithm below:

Let $a = 0$ (a denotes the number of computed facets).

Grouping. Group the n objects into $\lceil \frac{n}{p} \rceil$ groups of size p and compute the vertical decompositions of their convex hull. We obtain a set $\mathcal{T}$ of $O(\frac{n\lambda(p,m)}{p})$ tiny objects partitionned into $\lceil \frac{n}{p} \rceil$ non-overlapping subsets.

Marriage-before-conquest. Apply algorithm PartialMBC on the set $\mathcal{T}$ until each slab has less than $\lceil \frac{n}{p} \rceil$ walls, incrementing a each time we compute a bridge. If $a > p$ stop and return YES, i.e. $h > p$.

Jarvis's march. Fill the terminal slabs by running Jarvis's march inside each slab on a set of $O(\frac{n}{p})$ objects (at most $\lceil \frac{n}{p} \rceil$ spanning objects and $\lceil \frac{n}{p} \rceil$ objects that have a wall inside the slab), incrementing a and testing if $a > p$ each time we compute a new facet. If $a > p$ at some step then we stop the algorithm and return YES, i.e. $h > p$.

Default case. At this stage, we have computed the whole upper convex hull and $a = h$, the number of computed facets is less or equal to p. We return NO.

The overall cost of the grouping step is $O(n\frac{\lambda(p,m)}{p}\log p)$ as in section 4.1. The cost of the marriage-before-conquest algorithm is bounded by $O(n\frac{\lambda(p,m)}{p}\log p)$ since we stop the recursion process if the slab has less than $\lceil \frac{n}{p} \rceil$ walls. Indeed, we split into two balanced parts the walls of the tiny objects of $\mathcal{T}$ at each recursive call of the procedure. So that dividing the number of walls inside each slab by a factor $\lambda(p,m)$ amounts to computing at most $\lambda(p,m)$ bridge facets. Thus, the cost of running PartialMBC is bounded by $O(n\frac{\lambda(p,m)}{p}\log\lambda(p,m)) + n\frac{\lambda(p,m)}{p} = O(n\frac{\lambda(p,m)}{p}\log p)$ since $\log\lambda(p,m) = O(\log p)$. Let $c(n,p)$ denote the time complexity of this algorithm. Then, $c(n,p) = O(n\frac{\lambda(p,m)}{p}\log p) + O(\frac{n}{p}a')$ where a' is the number of computed facets during the Jarvis's march ($a' \leq a$). Clearly, $a' \leq p$ so that $c(n,p) = O(n\frac{\lambda(p,m)}{p}\log p)$. This proves the lemma. □

4.3. The Overall Algorithm

The scheme of the algorithm is to find a good estimate h_e of h, that is an estimate such that $h \leq h_e < h^2$, and to run the algorithm of section 4.1 with that estimate. The final algorithm is described below:

Initializing. Let $i = 0$ and $p = 2^{2^0} = 2$.

Estimating. While $(p < h)$ do $p \leftarrow \min\{n, p^2\}$ (this means that $i \leftarrow i + 1$ and $p = 2^{2^i}$)

Computing. Compute the upper convex hull using $p = h_e = 2^{2^i}$ (note that $h^2 > p \geq h$).

Note that we use the algorithm of section 6.2 to perform tests like $p < h$ in the while-loop.

Let $c(n, h)$ be the cost of the algorithm, we obtain:

$$c(n,h) = O(1) + O\left(\sum_{i=0}^{\lceil \log\log h \rceil} n \frac{\lambda(2^{2^i}, m)}{2^{2^i}} 2^i\right) + O\left(n \frac{\lambda(h_e, m)}{h_e} \log h_e\right).$$

Let $\beta(p, m)$ be an upper bound of the ratio $\frac{\lambda(p,m)}{p}$ that satisfies $\beta(p^2, m) = O(\beta(p, m))$ like $\beta(p, m) \leq O(2^{\alpha(p)^{c_m}})$ with c_m an integer depending on m (this upper-bound is deduced from the maximal length of (n, s)-Davenport-Schinzel sequences, see Table 1). We bound $c(n, h)$ as follows:

$$c(n,h) \leq O\left(n\beta(h,m) \sum_{i=0}^{\lceil \log\log h \rceil} 2^i\right) + O\left(n\beta(h^2, m) \log h^2\right),$$

$$c(n,h) = O(n\beta(h,m)\log h).$$

This yields the desired upper-bound $c(n, h) = O(n\beta(h, m) \log h)$.

Theorem 7 *There exists a deterministic algorithm that computes the upper convex hull of n planar convex objects of fixed type m in time $O(n\beta(h, m) \log h)$ using $O(n\beta(h, m))$ storage.*

This bound is very close to optimal since $\Omega(n \log h)$ is a lower bound[7]. In case of convex objects of type 2 (like disks, convex homothets, non-overlapping objects, etc.), the algorithm is truly optimal since $\frac{\lambda(h,2)}{h} = O(1)$ (see Ref.45). If $m > 2$ we do not know if our algorithm is optimal. We cannot reach the $\Omega(n \log h)$ lower bound (proved in Ref.7) with this method. Indeed, when grouping the objects into groups and computing their vertical decomposition, we create a set of tiny objects which is slightly supra-linear with respect to the original set of objects. This remark gives rise to the problem of the lower bound as soon as $m > 2$. Is $\Omega(n \frac{\lambda(h,m)}{h} \log h)$ a better lower bound for the convex hull problem? Can we group the objects in a better way so that the number of tiny objects obtained from the convex decomposition of the groups is less than $O(n \frac{\lambda(p,m)}{p})$ for a p-grouping?

In the following section, we show how this method can be used to compute upper envelopes of functions and line segments. In the latter case, we can improve the grouping step of the inputs so that we achieve an optimal $\Theta(n \log h)$-time algorithm in the case of line segments.

5. Computing Upper Envelopes

Let $\mathcal{F} = \{f_1, ..., f_n\}$ be a collection of n mono-variate, possibly partially defined, functions, all algebraic of some constant maximum degree. We denote by $E_{\mathcal{F}}$ its upper envelope, i.e. the pointwise maximum of the f_i's:

$$E_{\mathcal{F}}(x) = \max_{i \in \{1,\ldots,n\}} \{f_i(x)\},$$

where $f_i(x)$ is the value of the function f_i at abscissa x or $-\infty$ if x does not belong to the domain of definition of f_i. Throughout this paper, we will use the term function for the mathematical object itself or its graph. Thus, in term of graph, the upper envelope of functions can be seen as the part of the graphs of the f_i's visible from viewpoint $(0, +\infty)$. If the functions are partially defined, then the observer (which stands at point $(0, +\infty)$) may see the point $(0, -\infty)$, i.e. there exists vertical rays emanating from $(0, +\infty)$ that do not collide with the function graphs. In order to unify the definition of the mathematical object *upper envelope* in case of partially defined functions, we add an extra function $f_{-\infty}(\cdot)$ such that $f_{-\infty}(x) = -\infty, \forall\, x \in \mathbb{R}$. Thus,

$$E_{\mathcal{F}}(x) = \max_{i \in \{1,\ldots,n\}} \{f_i(x), f_{-\infty}(x)\} = \max_{i \in \{1,\ldots,n\}} \{f_i(x), -\infty\}.$$

The upper envelope is a sequence of maximal visible portions of the original functions. Hereafter, we call *facet* of the upper envelope each maximal visible portion of the original functions. A facet is fully determined by the function whose graph coincides with that facet, and its two endpoints. The size of the upper envelope $E_{\mathcal{F}}$ of $\mathcal{F}$, denoted by $|E_{\mathcal{F}}|$, is the number of facets of the upper envelope.

Set $\mathcal{F}$ is said of type m if any two functions of $\mathcal{F}$ intersect in at most m points. Line segments are of type 1, parabolæ are of type 2, ... Since the functions have a bounded descriptive size (algebraic functions of fixed degree), $\mathcal{F}$ is of fixed type m.

We can use the theory of Davenport-Schinzel[31,32,33,46] to bound the complexity of the upper envelope $E_{\mathcal{F}}$ of $\mathcal{F}$. The maximal length $\lambda(n, m)$ of an (n, m)-Davenport-Schinzel sequence is almost linear in n for fixed m[31,32,33]. It is well-known that the size of the upper envelope of n functions totally defined over $\mathbb{R}$ (resp. partially defined over $\mathbb{R}$) is bounded by $\lambda(n, m+1)$ (resp. $\lambda(n, m+3)$).

For example, line segments are partially defined functions intersecting pairwise in at most one point. Thus, the size of the upper envelope of n line segments is $\lambda(n, 3) = O(n\alpha(n))$. Here $\alpha(n)$ is the extremely slowly growing functional inverse of Ackermann's function[36]. This bound is tight: M. Sharir and A. Wiernik[36] built a set of n line segments such that the size of their upper envelope is $\Omega(n\alpha(n))$. However for practical implementation, it is worth noting that $\alpha(n) \leq 4$ for $n \leq \text{tower}(65536)$ where tower(i) is a tower of 2 of length i, i.e. $\text{tower}(1) = 2$ and $\text{tower}(i+1) = 2^{\text{tower(i)}}$.

The methodology previously described for computing convex hulls can be applied for computing upper envelopes. We briefly recall the main steps. Computing the bridge at a given oriented line Δ, i.e. the facet of $E_{\mathcal{F}}$ intersected by Δ, can be done

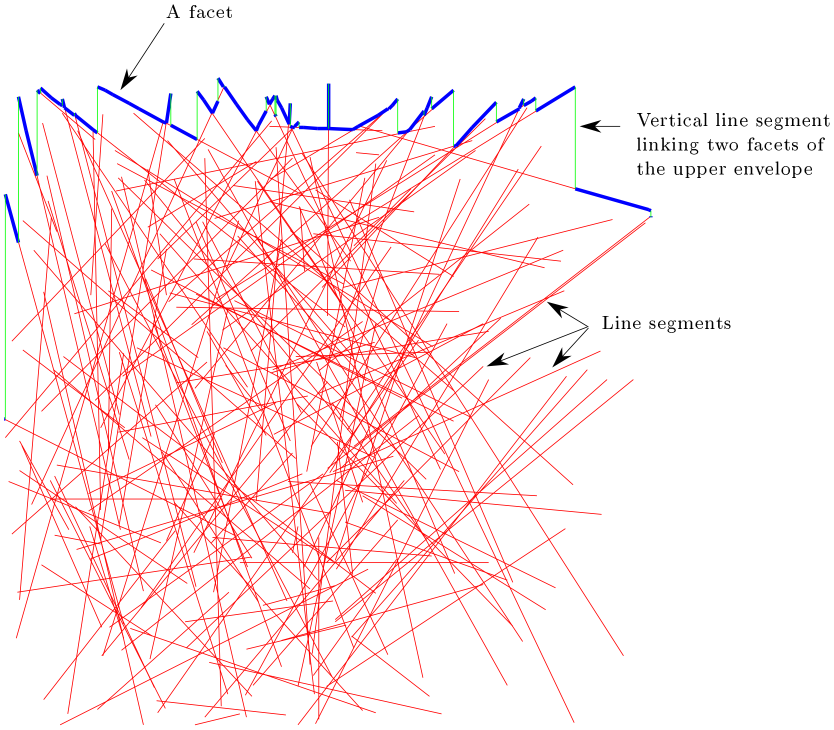

Figure 6: Upper envelope of 200 line segments. Facets are shown in bold.

almost trivially in linear time: first, we select the function which has the highest intersection point with Δ. Let f be that function. Then, in a second step, we find the two endpoints (to the left and right of Δ) limiting the bridge facet. As type m is fixed, we can compute the two endpoints in linear time. We can also design a linear-time-per-facet algorithm (an analogous algorithm of Jarvis's march). Then we consider the case of a set of functions partitionned into k subsets of pairwise nonintersecting subsets. We obtain a $O(n \log h + kh)$-time upper envelope algorithm.

We define the vertical decomposition of a group of functions as the partially defined functions induced by striping vertically the upper envelope. We follow the same steps as those of the convex hull algorithm and obtain an $O(n\beta(h,m)\log h)$-time $O(n\beta(h,m))$-storage algorithm with $\beta(h,m) = O(2^{\alpha(h)^{c_m}})$ where $c_m = \lceil \frac{m}{2} \rceil$ if the functions are partially defined and $c_m = \lceil \frac{m}{2} \rceil - 1$ otherwise (see Table 1). Note that the complexity of the upper envelope depends on both the number of intersection points and if the functions are partially or totally defined.

Thus, for the case of line segments we obtain an $O(n\alpha(h)\log h)$-time algorithm. We show in the following section how we can reach the optimal bound $\Omega(n \log h)$ by adapting the technique due to J. Hershberger[47]. The main idea is to group the the line segments efficiently. A family of functions is said to be *k-intersecting* if the functions are intersecting pairwise in at most k points. A set of *k-intersecting generalized segments* is a family of partially defined functions that are k-intersecting.

5.1. An Improved Algorithm for k-Intersecting Segments

W.l.o.g. we consider the case of line segments. The generalization of the result to k-intersecting generalized segments is straightforward. The main idea is to create groups so that the size of the vertical decomposition of each group remains linear. We first compute a *lazy* interval tree as follows: consider the $2n$ endpoints of the line segments and compute by recursive application of the median algorithm[44] a partition $\mathcal{P} = \{\mathcal{P}_1, ..., \mathcal{P}_p\}$ of the $2n$ endpoints so that each sheaf $\mathcal{P}_i$ has size $\frac{2n}{p}$ and the sheaves are x-ordered, i.e. $x(\mathcal{P}_i) < x(\mathcal{P}_j)$ for all $j > i$. We consider the following $p-1$ reference abscissæ and p x-ranges:

- For each sheaf $\mathcal{P}_i$, we associate the x-range X_i of the points $p_i \in \mathcal{P}_i$. Note that all the x-ranges of the sheaves are disjoint.
- Between two successive sheaves, we choose an abscissa a_i so that $X_i < a_i < X_{i+1}$, i.e an abscissa between two consecutive x-ranges of sheaves.

We build an interval tree $\mathcal{IT}$ as follows: each leaf of the interval tree corresponds to the x-range of a sheaf and each internal node to an abscissa separating the sheaves (see Figure 8). Then, we allocate the n line segments according to the lowest common ancestor of their two endpoints. At this step, all the segments are located into two kinds of sets:

- Those staying at a leaf of $\mathcal{IT}$. This means that the x-range of each of these line segments is included in the x-range of the sheaf. We say that these line segments are *unclassified.*

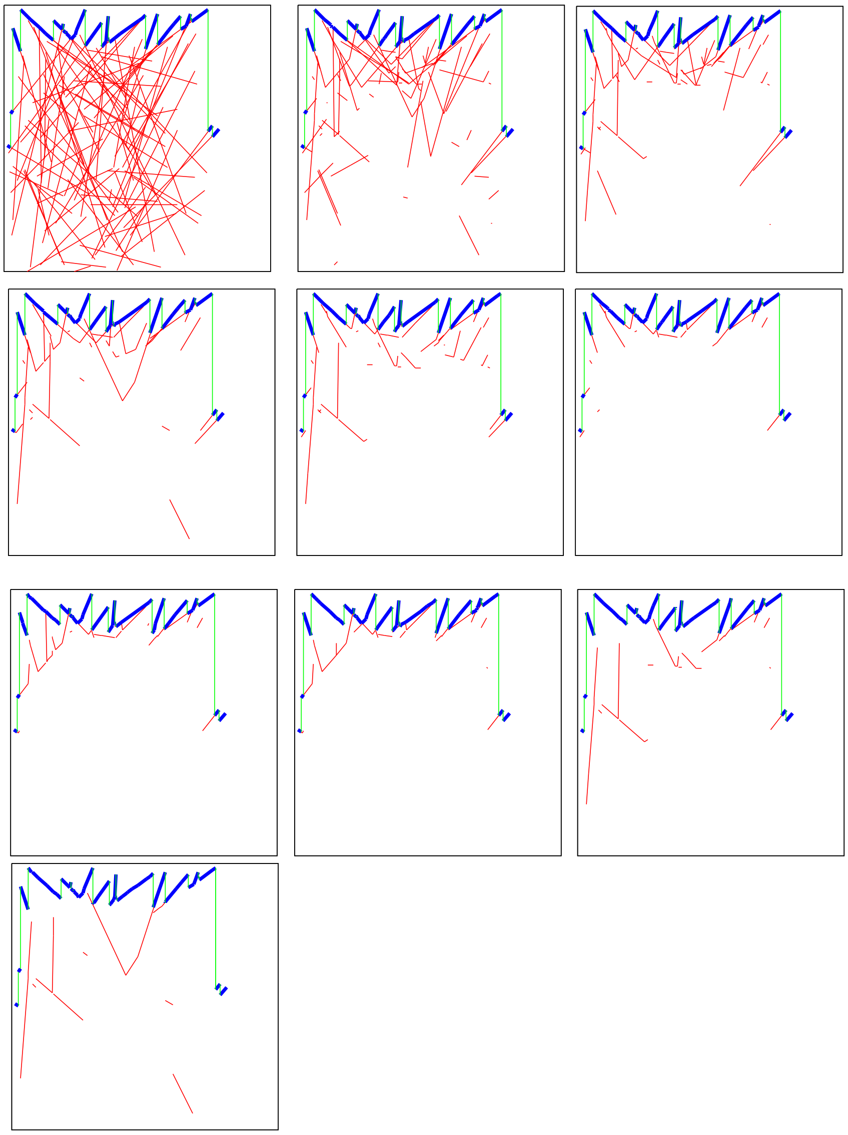

Figure 7: The upper envelope of 100 line segments. The upper left drawing depicts the upper envelope of 100 line segments. Then from left to right, and top to bottom, we first compute groups of size $10, 20, \ldots, 90$ and apply the marriage-before-conquest algorithm on the set of line segments resulting from the vertical decompositions of their upper envelopes.

- Those lying in an internal node of $\mathcal{IT}$. This means that all the line segments, whose lowest common ancestor of the abscissæ of their endpoints is the abscissa a_i, cross the vertical line $x = a_i$. Their upper envelope is linear in the number of line segments. We say that these segments are *classified*.

Following the communication of J. Hershberger[47], we notice that the upper envelope of the line segments allocated into a same internal level of $\mathcal{IT}$ is linear in the number of line segments. Indeed, the upper envelope of the segments allocated to a given internal node is linear (see Ref.47) because all these segments cross a vertical line and the segments of two nodes of a same internal level are separated by a vertical line by virtue of the interval tree. Let n_i denote the number of line segments at level i, $1 \leq i \leq \lceil \log p \rceil$. By grouping the line segments of each internal level of the interval tree into groups of size p and computing for each group the vertical decomposition of their upper envelope, we obtain an $O(\frac{n}{p} + \log p)$-coloration, i.e. a partition of the original set of n line segments into $\sum_{i=0}^{\lceil \log p \rceil} \lceil \frac{n_i}{p} \rceil = O(\frac{n}{p} + \log p)$ subsets of pairwise non-intersecting line segments resulting from the vertical decomposition of their upper envelope. We also color the unclassified line segments (those staying at a leaf of the interval tree) as follows: to the i-th line segment attached to a given leaf of the interval tree, we give it the color $(i, 2)$. Here, 2 means the unclassified line segments. Note that $i \leq \lceil \frac{n}{p} \rceil$. Moreover, two line segments with color $(i, 2)$ do not intersect since they belong to two different sheaves and are therefore x-separated.

Thus, globally, after an $O(n \log p)$-preprocessing time required for building the lazy interval tree, we obtain a $O(\frac{2n}{p} + \log p)$-coloration of a new created set of $O(n)$ line segments which has the same upper envelope as $\mathcal{O}$. We run the $O(n \log h + kh)$-time algorithm upon this new set. Since $k = \frac{2n}{p} + \log p$, we obtain an $O(n \log h + \frac{2n}{p} h + h \log p) = O((n + h) \log h)$-time algorithm with linear storage.

For the case of k-intersecting generalized segments, we note that the complexity of the upper envelope of the n_i k-intersecting segments at the i-th level of the interval tree is $O(\lambda(n_i, k+1))$[47]. It follows that the complexity of the upper envelopes (one upper envelope per group) of the n k-intersecting segments is $O(n\beta(h, k+1))$. Thus, we can compute the upper envelope of k-intersecting segments in time $O((n\beta(h, k+1) + h) \log h)$. The space requirement has also been reduced to $O(n\beta(h, k+1))$. A challenging problem is to design an algorithm that computes the upper envelope of n functions intersecting pairwise in at most m points in less than $O(\lambda(n, m+1) \log n)$ operations. Probably, if a better result is found, it may yield straightforwardly to a better output-sensitive algorithm since the crucial step of our method is to compute partitionned sets.

As a final remark, to underline the power of the grouping scheme, we show how in the case of line segments we can obtain again an $O(n \log h)$-time algorithm using ray shooting procedures. As before, we create $\lceil \frac{n}{p} \rceil$ groups of size p, compute their upper envelopes and preprocess these upper envelopes (which can be viewed as simple polygons, each of them of size $O(p\alpha(p)))$ for ray shooting. For each group, the time for computing its upper envelope and preprocess it for ray shooting is $O(p \log p)$[48,49]. Thus, the total time for the preprocessing step is $O(n \log p)$. The

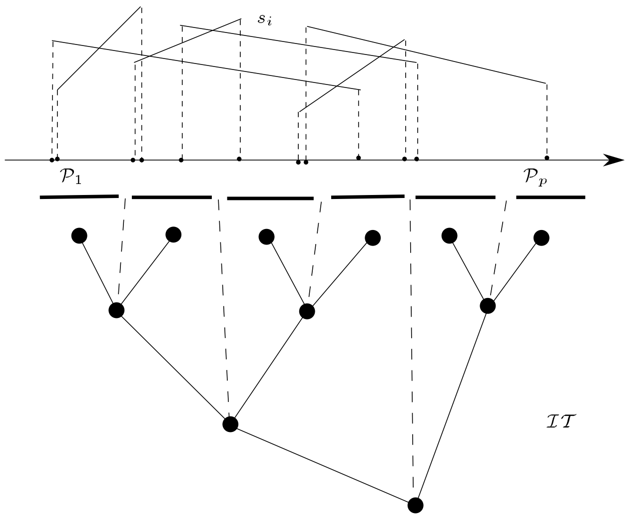

Figure 8: Building the lazy interval tree.

ray shooting query time of a group is $O(\log p)$. Then, the procedure walks from $x = -\infty$ to $x = \infty$ as follows: consider that the algorithm at some stage has found a portion of the upper envelope (a line segment) and therefore knows (by the rightmost endpoint e of that facet) which line segment s will support the following portion of the upper envelope. Then, for each group (in fact each simple polygon), we shoot a ray from the endpoint e following the direction of s. Finally, among the $\frac{n}{p}$ terminations, we choose the one that shorten the most the line segment s. The cost of this algorithm is $O(n \log p + (\frac{n}{p} \log p + \frac{n}{p})h)$. If $p = h$ then the algorithm has time complexity $O(n \log h)$. We use again the technic of approximation in order to achieve that bound.

6. Concluding Remarks

We have applied the marriage-before-conquest paradigm to the computation of the convex hull of n planar convex objects of fixed type m. We first described a linear-time algorithm to compute the bridge of the convex hull at a given oriented line. Then, we investigated the case where the family of objects consists of k subsets of non-overlapping objects. For that case, we designed an $O(n \log h + kh)$-time algorithm where h denotes the output-size. As a byproduct, we obtain an optimal $\Theta(n \log h)$-time algorithm for computing the convex hull of a set of non-overlapping objects. Moreover, if each object cannot intersect more than δ others then we design an $O(n \log h + \delta h)$-time algorithm. Finally, we transformed the problem of computing the convex hull of $\mathcal{O}$ to computing the convex hull of a set $\mathcal{T}$ partitioned into non-overlapping subsets such that $\mathcal{CH}(\mathcal{O}) = \mathcal{CH}(\mathcal{T})$. (We use nonoutput-sensitive algorithms in order to get $\mathcal{T}$.)

The size of the partition of $\mathcal{T}$, i.e. the number of non-overlapping subsets, depends on the size of the output. Since we do not know the output-size, we iteratively estimate it. We finally choose a good estimate to compute the convex hull of a set of n planar convex objects of fixed type m in $O(n\beta(h,m)\log h)$ time where $\beta(h,m)$ is an extremely slowly growing function. We can follow the same scheme for computing the upper envelope of possibly partially defined functions. In that case, the bridge at a given oriented line is the maximal piece of the lower envelope intersected by that line and can be computed trivially in linear time. Therefore, when computing the convex hull of objects, one might first compute the dual functions and apply the upper envelope algorithm. However, computing directly the convex hull remains interesting in the case of k-colored partitions because, in that case, one does not need to apply the grouping scheme.

All these algorithms can be easily parallelized onto EREW PRAM multi-computers, following the algorithm of S. Akl[50,51]. D.G. Kirkpatrick and R. Seidel[7] proved that $\Omega(n \log h)$ is a lower bound for computing the convex hull of a set of n points where h is the number of hull vertices. Can we improve that lower bound in the case of convex objects of fixed type m? It would also be interesting to find other applications of this method and to generalize it to higher dimensions.

Acknowledgements

We would like to thank Jean-Daniel Boissonnat for his careful reading of the manuscript and Jean-Pierre Merlet for supplying us with his interactive drawing preparation system JPdraw . We also thank an anonymous referee for his/her valuable comments. This work was supported in part by ESPRIT Basic Research Actions 7141 (ALCOMII).

References

1. K. L. Clarkson. New applications of random sampling in computational geometry. *Discrete Comput. Geom.*, 2:195–222, 1987.
2. B. Chazelle. An optimal convex hull algorithm for point sets in any fixed dimension. Technical Report CS-TR-336-91, Dept. Comput. Sci., Princeton Univ., Princeton, NJ, 1991.
3. H. Brönninman. *Derandomization of geometric algorithms*. Ph.D. thesis, Princeton Univ., 1995.
4. B. Grünbaum. *Convex Polytopes*. Wiley, New York, NY, 1967.
5. P. McMullen. The maximal number of faces of a convex polytope. *Mathematica*, 17:179–184, 1970.
6. D. G. Kirkpatrick and R. Seidel. The ultimate planar convex hull algorithm? In *Proc. 20th Allerton Conf. Commun. Control Comput.*, pages 35–42, Monticello, Illinois, 1982.
7. D. G. Kirkpatrick and R. Seidel. The ultimate planar convex hull algorithm? *SIAM J. Comput.*, 15:287–299, 1986.
8. H. Edelsbrunner and W. Shi. An $O(n \log^2 h)$ time algorithm for the three-dimensional convex hull problem. *SIAM J. Comput.*, 20:259–277, 1991.
9. K. L. Clarkson and P. W. Shor. Applications of random sampling in computational

geometry, II. *Discrete Comput. Geom.*, 4:387–421, 1989.

10. B. Chazelle and J. Matoušek. Derandomizing an output-sensitive convex hull algorithm in three dimensions. Technical report, Dept. Comput. Sci., Princeton Univ., 1992.
11. R. Seidel. Constructing higher-dimensional convex hulls at logarithmic cost per face. In *Proc. 18th Annu. ACM Sympos. Theory Comput.*, pages 404–413, 1986.
12. J. Matoušek. Linear optimization queries. *J. Algorithms*, 14:432–448, 1993. The results combined with results of O. Schwarzkopf also appear in *Proc. 8th ACM Sympos. Comput. Geom.*, 1992, pages 16–25.
13. J. Matoušek and O. Schwarzkopf. Linear optimization queries. In *Proc. 8th Annu. ACM Sympos. Comput. Geom.*, pages 16–25, 1992.
14. Timothy M.Y. Chan, Jack Snoeyink, and Chee-Keng Yap. Output-Sensitive Construction of Polytopes in Four Dimensions and Clipped Voronoi Diagrams in Three. In *Proc. 6th ACM-SIAM Sympos. Discrete Algorithms (SODA)*, pages 282 – 291. 1995.
15. M. Y. Chan. Output-Sensitive Results on Convex Hulls, Extreme Points, and Related Problems. *Proc. of the A.C.M. Symp. on Computational Geometry*, 1995.
16. D. R. Chand and S. S. Kapur. An algorithm for convex polytopes. *J. ACM*, 17:78–86, 1970.
17. G. F. Swart. Finding the convex hull facet by facet. *J. Algorithms*, 6:17–48, 1985.
18. P. K. Agarwal and J. Matoušek. Ray shooting and parametric search. *SIAM J. Comput.*, 22(4):794–806, 1993.
19. J. Matoušek and O. Schwarzkopf. On ray shooting in convex polytopes. *Discrete Comput. Geom.*, 10(2):215–232, 1993.
20. C. Bajaj and M. S. Kim. Convex hull of objects bounded by algebraic curves. *Algorithmica*, 6:533–553, 1991.
21. A. A. Schäffer and C. J. Van Wyk. Convex hulls of piecewise-smooth Jordan curves. *J. Algorithms*, 8:66–94, 1987.
22. D. P. Dobkin and D. L. Souvaine. Computational geometry in a curved world. *Algorithmica*, 5:421–457, 1990.
23. B. Bhattacharya and H. El-Gindy. A New Linear Convex Hull Algorithm for Simple Polygons. *IEEE Transactions on Information Theory*, 30, 84.
24. D. McCallum and D. Avis. A linear algorithm for finding the convex hull of a simple polygon. *Inform. Process. Lett.*, 9:201–206, 1979.
25. S. Y. Shin and T. C. Woo. Finding the convex hull of a simple polygon in linear time. *Pattern Recogn.*, 19:453–458, 1986.
26. R. L. Graham and F. F. Yao. Finding the convex hull of a simple polygon. *J. Algorithms*, 4:324–331, 1983.
27. D. Rappaport. A convex hull algorithm for discs, and applications. *Comput. Geom. Theory Appl.*, 1(3):171–181, 1992.
28. J. D. Boissonnat, A. Cérézo, O. Devillers, J. Duquesne, and M. Yvinec. An algorithm for constructing the convex hull of a set of spheres in dimension d. In *Proc. 4th Canad. Conf. Comput. Geom.*, pages 269–273, 1992.
29. R. A. Jarvis. On the identification of the convex hull of a finite set of points in the plane. *Inform. Process. Lett.*, 2:18–21, 1973.
30. D. Halperin and M. H. Overmars. Spheres, molecules, and hidden surface removal.

In *Proc. 10th Annu. ACM Sympos. Comput. Geom.*, pages 113–122, 1994.

31. P. K. Agarwal, M. Sharir, and P. Shor. Sharp upper and lower bounds on the length of general Davenport-Schinzel sequences. *J. Combin. Theory Ser. A*, 52(2):228–274, 1989.
32. M. Sharir. Almost linear upper bounds for the length of general Davenport-Schinzel sequences. *Combinatorica*, 7:131–143, 1987.
33. M. Sharir. Improved lower bounds on the length of Davenport-Schinzel sequences. *Combinatorica*, 8:117–124, 1988.
34. M. Sharir, R. Cole, K. Kedem, D. Leven, R. Pollack, and S. Sifrony. Geometric applications of Davenport-Schinzel sequences. In *Proc. 27th Annu. IEEE Sympos. Found. Comput. Sci.*, pages 77–86, 1986.
35. H. Edelsbrunner, L. Guibas, and M. Sharir. The upper envelope of piecewise linear functions: algorithms and applications. *Discrete Comput. Geom.*, 4:311–336, 1989.
36. A. Wiernik and M. Sharir. Planar realizations of nonlinear Davenport-Schinzel sequences by segments. *Discrete Comput. Geom.*, 3:15–47, 1988.
37. K. L. Clarkson. Applications of random sampling in computational geometry, II. In *Proc. 4th Annu. ACM Sympos. Comput. Geom.*, pages 1–11, 1988.
38. K. L. Clarkson and P. W. Shor. Algorithms for diametral pairs and convex hulls that are optimal, randomized, and incremental. In *Proc. 4th Annu. ACM Sympos. Comput. Geom.*, pages 12–17, 1988.
39. M. Sharir and E. Welzl. A combinatorial bound for linear programming and related problems. In *Proc. 9th Sympos. Theoret. Aspects Comput. Sci.*, volume 577 of *Lecture Notes in Computer Science*, pages 569–579. Springer-Verlag, 1992.
40. J. Matoušek, M. Sharir, and E. Welzl. A subexponential bound for linear programming. In *Proc. 8th Annu. ACM Sympos. Comput. Geom.*, pages 1–8, 1992.
41. N. Amenta. Helly theorems and generalized linear programming. In *Proc. 9th Annu. ACM Sympos. Comput. Geom.*, pages 63–72, 1993.
42. N. Amenta. Helly-type theorems and generalized linear programming. *Discrete Comput. Geom.*, 12:241–261, 1994.
43. B. Chazelle and J. Matoušek. On linear-time deterministic algorithms for optimization problems in fixed dimension. In *Proc. 4th ACM-SIAM Sympos. Discrete Algorithms*, pages 281–290, 1993.
44. M. Blum, R.W. Floyd, V.R. Pratt, R.L. Rivest, and Tarjan. Time bounds for selection. *J. Comput. Syst. Sci.*, 7, 1972.
45. K. Kedem, R. Livne, J. Pach, and M. Sharir. On the union of Jordan regions and collision-free translational motion amidst polygonal obstacles. *Discrete Comput. Geom.*, 1:59–71, 1986.
46. M. Sharir and P. K. Agarwal. *Davenport-Schinzel Sequences and Their Geometric Applications.* Cambridge University Press, New York, 1995.
47. J. Hershberger. Finding the upper envelope of n line segments in $O(n \log n)$ time. *Inform. Process. Lett.*, 33:169–174, 1989.
48. B. Chazelle, H. Edelsbrunner, M. Grigni, L. Guibas, J. Hershberger, M. Sharir, and J. Snoeyink. Ray shooting in polygons using geodesic triangulations. In *Proc. 18th Internat. Colloq. Automata Lang. Program.*, volume 510 of *Lecture Notes in Computer Science*, pages 661–673. Springer-Verlag, 1991.
49. J. Hershberger and S. Suri. A pedestrian approach to ray shooting: Shoot a ray, take a walk. In *Proc. 4th ACM-SIAM Sympos. Discrete Algorithms*, pages 54–63,

1993.

50. S. G. Akl. Optimal algorithms for computing convex hulls and for sorting. *Computing*, 33:1–11, 1984.

51. S. G. Akl. Optimal parallel algorithms for selection, sorting and computing convex hulls. In G. T. Toussaint, editor, *Computational Geometry*, pages 1–22. North-Holland, Amsterdam, Netherlands, 1985.