Distance-Based Subdivision for Translational LP Containment

Karen Daniels [*] Victor J. Milenkovic[†]

Abstract

The two-dimensional translational *containment* problem is to find translations for a collection of polygons which place them inside a polygonal container without overlapping. The polygons and container may be nonconvex. Our linear-programming-based (LP) translational containment algorithm uses our restrict/evaluate/subdivide paradigm which operates on two-dimensional configuration spaces. The following *distance-based subdivision* problem arises during the subdivision step: given a polygon U and a point t which is outside U but inside the convex hull of U, find the line L through t which partitions U into two pieces U^- and U^+ on opposite sides of L such that:

$$\min(\Delta(t, \mathrm{CH}(U^-)),\ \Delta(t, \mathrm{CH}(U^+)))$$

is maximized, where $\mathrm{CH}(U)$ is the convex hull of U and $\Delta(a, B)$ is the Euclidean distance from a point a to a point set B. We show that if U is connected, then the distance-based subdivision problem can be solved in $O(|U|)$ time in a real arithmetic model and in $O(|U| + \log(\log \epsilon^{-1}))$ time in a rational arithmetic model, where the input parameter ϵ is the desired accuracy.

1 Introduction.

A number of industries generate new parts by cutting them from stock material: cloth, leather (hides), sheet metal, *etc.* These industries require good solutions to *containment* problems. *Containment* is the question of whether a given set of part shapes can be fit, without overlapping, into a given container. The shapes and container are represented as polygons and may be nonconvex. Material such as cloth has a grain, and it sometimes has a "nap" (*e.g.* velvet or corduroy) or a colored pattern (*e.g.* stripes or plaid). A part cut out of such material has only one, two, four, or possibly eight allowed orientations, not arbitrary rotation. For such materials, containment can be decomposed into two NP-complete subproblems [11]: 1) selection of discrete orientations and 2) *translational* containment. *Translational containment*, containment with fixed orientations, is clearly an important subproblem.

In previous work [11, 12], we introduced an algorithm for translational containment based entirely on the principles of mathematical programming. **Restriction** establishes lower bounds through relaxation and the solution of linear programs. **Evaluation** establishes upper bounds by finding potential solutions. **Subdivision** branches, when necessary, by introducing a cutting plane. The objective that the containment algorithm minimizes is the overlap among the placed polygons. Naturally, we seek a layout with zero overlap. The algorithm is called the *linear-programming-based containment* algorithm or *LP containment* algorithm. Restriction, evaluation, and subdivision operate on two-dimensional configuration spaces. Each configuration space is a polygonal region U_{ij} representing the non-overlapping positions for polygon j relative to polygon i. U_{ij} may have multiple components and holes. The evaluation algorithm iteratively builds and solves an *overlap linear program* (OLP) whose constraints are derived from the edges of the U_{ij}s and their convex hulls. The evaluation algorithm terminates at a local overlap minimum. If this minimum yields an overlapping layout, then the U_{ij} corresponding to the "most overlapped" pair of polygons in the layout is subdivided. Denote this U_{ij} by U. Because polygons i and

[*] Harvard University, Division of Applied Sciences, and University of Miami. Email: daniels@das.harvard.edu. This research was funded by the Textile/Clothing Technology Corporation from funds awarded by the Alfred P. Sloan Foundation and by NSF grants CCR-91-157993 and CCR-90-09272.

[†] University of Miami, Department of Math and Computer Science. Email: vjm@cs.miami.edu. This research was funded by the Textile/Clothing Technology Corporation from funds awarded by the Alfred P. Sloan Foundation, by NSF grant CCR-91-157993, and by a subcontract of a National Textile Center grant to Auburn University, Department of Consumer Affairs.

j overlap, the relative position of polygon j with respect to polygon i is a point t which is outside U. The OLP constraints force t to be inside the convex hull of U.

1.1 Distance-Based Subdivision

We introduced *distance-based subdivision* (DBS) [1, 12] for use within LP containment. The DBS technique eliminates a "false" (local but not global) overlap minimum and all layouts near that minimum, allowing the containment algorithm to make progress towards the global overlap minimum with each subdivision. There are three cases: 1) **U has multiple components.** It sets U^- equal to the component of U which is closest to t. It sets U^+ equal to the union of the other components. 2) **U is connected and every ray out of t intersects U.** It selects the vertex u of U which is nearest to t. It splits U into U^-, the closure of $H^- \cap U$, and U^+, the closure of $H^+ \cap U$, where H^- and H^+ are the open half-planes to the left and right of LINE(tu), respectively.[1] 3) **U is connected and t can "see to infinity" along some ray.** It selects a line L through t which splits U into U^- and U^+. The line is chosen to maximize

$$d(L) = \min(\Delta(t, \mathrm{CH}(U^-)),\ \Delta(t, \mathrm{CH}(U^+))),$$

the minimum Euclidean distance from t to the convex hulls of the two "halves" of U. For cases 1 and 2, the closest component or vertex of U to t can clearly be located in $\mathrm{O}(|U|)$ time. Two tasks remain: 1) determine whether t can "see to infinity" when U is connected (case 2 or 3) and, if so, 2) compute the line described in case 3, which we denote by $L_{\max}$.

The DBS algorithm which we implemented within LP containment[2] solves these tasks in $\mathrm{O}(|U| \log \epsilon^{-1})$ time, where ϵ is the accuracy. In our experiments [1] on infeasible containment problems[3], using overlap minimization for the evaluation step followed by distance-based subdivision produces fewer subdivisions than our earlier "naive" LP-based containment algorithm. The naive LP algorithm [2, 10] solves a single linear program during the evaluation step. Its subdivision line is the extension of an edge of some U_{ij}, and the choice of U_{ij} is not based on the results of evaluation. Tight coupling of evaluation and subdivision within LP containment appears to produce good results in practice. We are therefore interested in the fastest possible DBS algorithm, which is the subject of this paper. Here we present an algorithm[4] which shows how to determine in $\mathrm{O}(|U|)$ time whether t can "see to infinity" when U is connected and, if so, how to compute $L_{\max}$ in $\mathrm{O}(|U|)$ time in a real arithmetic model and in $\mathrm{O}(|U| + \log(\log \epsilon^{-1}))$ time in a rational arithmetic model, where ϵ is the accuracy.

1.2 Overview.

To maximize $d(L)$, it suffices to consider lines in the set $\mathcal{L} = \{L | d(L) \neq 0\}$. Each $L \in \mathcal{L}$ contains a ray out of t along which t can "see to infinity." Section 2 shows how to find $\mathcal{L}$ in $\mathrm{O}(|U|)$ time. If $\mathcal{L} = \emptyset$, then U and t satisfy case 2; otherwise they satisfy case 3. Section 2 also shows how to construct in $\mathrm{O}(|U|)$ time an ordered list E of $\mathrm{O}(|U|)$ subedges of the boundary of U, where E is the part of the boundary of U which is visible to t. It also shows how to find a sublist G of E such that $\mathrm{CLOSURE}(\mathcal{L}) = \{\mathrm{LINE}(tu) | u \in e \text{ for } e \in G\}$.[5] Section 3 shows that binary search on the edge index in G can determine the edge in the list which contains $u_{\max}$ such that $L_{\max} = \mathrm{LINE}(tu_{\max})$. It also observes that binary search can be used on any partition G' of G. Section 4 creates a partition G' of G with the property that, with $\mathrm{O}(|U|)$ preprocessing time, we can evaluate a given vertex of any edge in G' in $\mathrm{O}(1)$ time within the binary search. It shows that G' contains $\mathrm{O}(|U|)$ edges and can be constructed in $\mathrm{O}(|U|)$ time. Section 4 also shows that, if the binary search determines that $u_{\max}$ is in the interior of an edge in G', then $\mathrm{O}(|U|)$ preprocessing allows us to retrieve, in $\mathrm{O}(1)$ time, the information necessary to calculate $u_{\max}$. Section 5 shows that, in this case, $u_{\max}$ can be calculated in $\mathrm{O}(1)$ time in a real arithmetic model and in $\mathrm{O}(\log(\log \epsilon^{-1}))$ time in a rational arithmetic model, where ϵ is the accuracy. This demonstrates

[1]A point p is on the left of LINE(ab) if $(b-a) \times (p-a) > 0$: *left* is from the point of view of an observer standing at a and facing toward b.

[2]This implementation of our LP containment algorithm has been licensed by Gerber Garment Technologies, the largest provider of textile CAD/CAM software in the U.S., and they are incorporating it into an existing CAD/CAM software product.

[3]Data sets are from the apparel industry.

[4]The algorithm also appears in our technical report [11].

[5]The set of lines through t is a metric space under angle, so closure is well-defined. Taking the closure adds the two lines which "bound" the set of lines in $\mathcal{L}$.

that $L_{\max}$ can be computed in $O(|U|)$ time in a real arithmetic model and in $O(|U| + \log(\log \epsilon^{-1}))$ time in a rational arithmetic model.

2 Finding $\mathcal{L}$.

Lemma 2.1 *If U is connected, then in $O(|U|)$ time we can find $\mathcal{L} = \{L | d(L) \neq 0\}$.*

Proof: $L \in \mathcal{L}$ if and only if it contains a ray out of t which does not intersect U. The set of rays $\text{RAYS}(t, U)$ out of t that hit U is connected[6] because U is connected. Therefore, the set of rays $\overline{\text{RAYS}(t,U)} = \text{RAYS}(t) \setminus \text{RAYS}(t,U)$ that do not intersect U is also connected; these are the rays we need. Since $t \in \text{CH}(U)$, each ray in $\overline{\text{RAYS}(t,U)}$ must pass through the boundary of $\text{CH}(U)$. The (connected) set $\{\rho \in \overline{\text{RAYS}(t,U)} | (\rho \cap \text{BOUNDARY}(\text{CH}(U))) \neq \emptyset\}$ must be a subset S of a *single* edge e of $\text{CH}(U)$: if the set contained a vertex of $\text{CH}(U)$, then we could move this vertex towards t and make $\text{CH}(U)$ smaller. The edge e is not an edge of U. Furthermore, let H be the component of $\text{CH}(U) \cap \overline{U}$ which contains t, and let V be the visibility polygon of H with respect to t. If V has an edge which is not collinear with t and which is not a subset of an edge of U, then the interior of that edge is S and $\mathcal{L} = \{\text{LINE}(st) | s \in S\}$. Otherwise, S does not exist and $\mathcal{L} = \emptyset$. Now we establish the running time. The convex hull of the outer boundary equals $\text{CH}(U)$ and can be computed in linear time [9, 3, 5]. The boundary of H can be determined in linear time. Since U is connected, it has no "islands" inside H, and thus H is a simple polygon. The visibility polygon V for a simple polygon H can be found in $O(|H|)$ ($\leq O(|U|)$) time [4, 8, 7]. ■

Lemma 2.2 *If U and t satisfy case 3, then in $O(|U|)$ time we can construct: 1) an ordered list E of $O(|U|)$ subedges of the boundary of U, where E is the part of the boundary of U that is visible to t and 2) a sublist G of E such that* $\text{CLOSURE}(\mathcal{L}) = \{\text{LINE}(tu) | u \in e \textit{ for } e \in G\}$.

Proof: Let V be the visibility polygon from the proof of Lemma 2.1. To form E, remove the single edge $\text{CLOSURE}(S)$ of V which is a subset of the boundary of $\text{CH}(U)$. Remove edges of V which are collinear with t. The result is a list of edges E which we can assume are ordered clockwise with respect to t. The list E can be represented as: $A_1A_2, A_3A_4, \ldots, A_{2n-1}A_{2n}$, where A_1 and A_{2n} are the endpoints of S and $\text{RAY}(tA_{2i})$ equals $\text{RAY}(tA_{2i+1})$, $1 \leq i \leq n-1$. Because U is connected and $t \in \text{CH}(U)$, both $\text{LINE}(A_1t)$ and $\text{LINE}(A_{2n}t)$ intersect E beyond t at least once (and at most twice if A_1t (or $A_{2n}t$) passes through A_{2i} and A_{2i+1} and these are not the same point). If an intersection point is in the interior of an edge in E, split that edge and reindex E. Let the intersection points of $\text{LINE}(A_{2n}t)$ with E (beyond t) be $A_{2\alpha-1}$ and $A_{2\alpha-2}$ and the intersection points of $\text{LINE}(A_1t)$ with E (beyond t) be $A_{2\beta}$ and $A_{2\beta+1}$. Let G be the following sublist of E: $A_{2\alpha-1}A_{2\alpha}, \ldots, A_{2\beta-1}A_{2\beta}$. Because $\mathcal{L}$ is connected[7] and, from the proof of Lemma 2.1, $\mathcal{L} = \{\text{LINE}(st) | s \in S\}$, we conclude that $\text{CLOSURE}(\mathcal{L}) = \{\text{LINE}(tu) | u \in e \text{ for } e \in G\}$, as required. The construction time is $O(|U|)$ for two reasons: 1) Lemma 2.1 guarantees that V can be constructed in $O(|U|)$ time and 2) V, being the visibility polygon of the simple polygon H of size $O(|U|)$, has $O(|U|)$ edges. The second reason also guarantees that E has $O(|U|)$ edges. ■

3 Binary Search.

Here we show that binary search on the value of the edge index i in G, $\alpha \leq i \leq \beta$, can determine the edge in G which contains $u_{\max}$ such that $L_{\max} = \text{LINE}(tu_{\max})$. This result also holds for any partition of G. Let $U^-(L)$ denote U^- when U is split by L, and define $U^+(L)$ analogously. Let $d^-(L) = \Delta(t, \text{CH}(U^-(L)))$ and $d^+(L) = \Delta(t, \text{CH}(U^+(L)))$. The functions $d^-(L)$ and $d^+(L)$ are discontinuous where L intersects both A_{2i} and A_{2i+1} but $A_{2i} \neq A_{2i+1}$ (this occurs where L intersects an edge of V which is collinear with t). Unfortunately, $d^-(\text{LINE}(tA_{2i})) = d^-(\text{LINE}(tA_{2i+1}))$ and $d^+(\text{LINE}(tA_{2i})) = d^+(\text{LINE}(tA_{2i+1}))$, so $d^-(L)$ and $d^+(L)$ cannot reveal if $L_{\max}$ occurs at a discontinuity. To overcome this problem we define $d^-(u)$ and $d^+(u)$, for $u \in A_{2i-1}A_{2i}$,

[6] The union of the rays is a single wedge.

[7] The union of lines in $\mathcal{L}$ is a double wedge through t, bounded by two lines.

as follows: $d^-(u) = \Delta(t, \mathrm{CH}(A_1, A_2, \ldots, A_{2i-1}, u))$ and $d^+(u) = \Delta(t, \mathrm{CH}(A_{2n}, A_{2n-1}, \ldots, A_{2i}, u))$. To evaluate each value of i within the binary search, perform the following tests. If $d^-(A_{2i-1}) > d^+(A_{2i-1})$ and $d^-(A_{2i}) < d^+(A_{2i})$, then return edge $A_{2i-1}A_{2i}$. If $d^-(A_{2i}) \geq d^+(A_{2i})$ and $d^-(A_{2i+1}) \leq d^+(A_{2i+1})$, then return vertex A_{2i}. If $d^-(A_{2i-2}) \geq d^+(A_{2i-2})$ and $d^-(A_{2i-1}) \leq d^+(A_{2i-1})$, then return vertex A_{2i-1}. Otherwise, return NULL. The search initializes i_{lower} to α and i_{upper} to β. For each value of i, if evaluation returns NULL, one of the bounds on i is updated based on the test $d^-(A_{2i-1}) > d^+(A_{2i-1})$.

Lemma 3.1 *This binary search on the value of the edge index i in G, $\alpha \leq i \leq \beta$, yields either the vertex A_j of G such that $d(A_j)$ maximizes $d(L)$ for $L \in \mathcal{L}$ or the edge $A_{2i-1}A_{2i}$ of G such that $d(u)$ maximizes $d(L)$ for $L \in \mathcal{L}$ for some $u \in$ INTERIOR$(A_{2i-1}A_{2i})$.*

Proof: We claim that the search maintains the following invariant: $d^-(A_{2i_{\text{lower}}-1}) > d^+(A_{2i_{\text{lower}}-1})$ and $d^-(A_{2i_{\text{upper}}}) < d^+(A_{2i_{\text{upper}}})$ and that the "ordering"[8] of $d^+(u)$ and $d^-(u)$ changes exactly once for $i_{\text{lower}} \leq i \leq i_{\text{upper}}$. The first part of the invariant is true at the start of the search because $d^-(A_{2\alpha-1}) > d^+(A_{2\alpha-1}) = 0$ and $0 = d^-(A_{2\beta}) < d^+(A_{2\beta})$. The updates of i_{lower} and i_{upper} for each i maintain this part of the invariant. The second part of the invariant relies on the following monotonicity result by Daniels [1]. Parameterize the line segment A_1A_{2n} by the linear function $\gamma(\tau)$, for $\tau \in [0,1]$, such that $\gamma(0) = A_{2n}$ and $\gamma(1) = A_1$. Daniels shows that the distance function $d^-(\mathrm{LINE}(\gamma(\tau)t))$ is non-increasing for τ over $[0,1]$. For $\tau, \tau' \in [0,1]$ and $\tau' > \tau$, LINE$(\gamma(\tau')t)$ is rotated clockwise from LINE$(\gamma(\tau)t)$. Her proof shows that, as a consequence, "more" of U^- lies to the left of LINE$(\gamma(\tau')t)$ than LINE$(\gamma(\tau)t)$, and thus $\mathrm{CH}(U^-(\mathrm{LINE}(\gamma(\tau)t))) \subseteq \mathrm{CH}(U^-(\mathrm{LINE}(\gamma(\tau')t)))$, which implies that $d^-(\mathrm{LINE}(\gamma(\tau)t)) \geq d^-(\mathrm{LINE}(\gamma(\tau')t))$ and therefore establishes that $d^-(\mathrm{LINE}(\gamma(\tau)t))$ is non-increasing. A similar argument shows that $d^+(\mathrm{LINE}(\gamma(\tau)t))$ is non-decreasing for τ over $[0,1]$. The full paper shows that $d^-(u)$ $(d^+(u))$ has the same monotonicity property as $d^-(\mathrm{LINE}(tu))$ $(d^+(\mathrm{LINE}(tu)))$. Monotonicity guarantees that the minimum of $d^-(u)$ and $d^+(u)$ is maximized where the ordering changes. The full paper shows that the minimum of $d^-(L)$ and $d^+(L)$ is maximized where the minimum of $d^-(u)$ and $d^+(u)$ is maximized. ■

4 Preprocessing.

Lemma 3.1 allows us to find the vertex or edge in G associated with $L_{\max}$ using binary search on the value of the index of G. The running time of the binary search is dominated by $O(\log(|G|)X) = O(\log(|U|)X)$, where X is the time required to calculate $d^+(A_j)$ and $d^-(A_j)$ for a given vertex A_j of G. This allows $L_{\max}$ to be found in time $O(|U| + \log(|U|)X + Y)$, where Y is the time to find $u \in$ INTERIOR$(A_{2i-1}A_{2i})$ such that $d(u)$ maximizes $d(L)$ for $L \in \mathcal{L}$ (if the binary search returns an edge instead of a vertex). This section shows how to preprocess G in $O(|U|)$ time so that $X = O(1)$ and so that $Y = O(1)$ in a real arithmetic model and $Y = O(\log(\log \epsilon^{-1}))$ in a rational arithmetic model with accuracy ϵ. (Finding $u \in$ INTERIOR$(A_{2i-1}A_{2i})$ such that $d(u)$ maximizes $d(L)$ for $L \in \mathcal{L}$ is addressed in Section 5.) This allows us to find $L_{\max}$ in $O(|U|)$ time in a real arithmetic model and in $O(|U| + \log(\log \epsilon^{-1}))$ time in a rational arithmetic model. Some of the preprocessing involves partitioning G in $O(|U|)$ time into G' which has $O(|U|)$ edges. Clearly, any partition G' of G also has the binary search property and, since G' has $O(|U|)$ edges and is constructed in $O(|U|)$ time, the asymptotic running time of the overall algorithm is unaffected by the partitioning. Before describing the preprocessing, we introduce more notation. If u is in edge $A_{2i-1}A_{2i}$ of G, $(\alpha \leq i \leq \beta)$, then denote by $C^-(u)$ the portion of $\mathrm{CH}(A_1, A_2, \ldots, A_{2i-1}, u)$ which is visible from t. Let $c^-(u)$ be the closest point of $C^-(u)$ to t. Similarly, define $C^+(u)$ to be the portion of $\mathrm{CH}(A_{2n}, A_{2n-1}, \ldots, A_{2i}, u)$ which is visible from t and define $c^+(u)$ be the closest point of $C^-(u)$ to t. Lemma 4.1 below allows us to use $C^-(u)$ and $C^+(u)$ instead of $\mathrm{CH}(U^-(\mathrm{LINE}(tu)))$ and $\mathrm{CH}(U^+(\mathrm{LINE}(tu)))$ in the remainder of the paper.

Lemma 4.1 *$d^-(u) = \Delta(t, C^-(u))$, and $d^+(u) = \Delta(t, C^+(u))$.*

Consider $u \in A_{2i-1}A_{2i}, \alpha \leq i \leq \beta$. The vertices of $C^-(u)$ are a subsequence of $A_1, A_2, \ldots, A_{2i-1}$ plus the last (clockwise most)[9] vertex is u. Similarly, the vertices of $C^+(u)$ are a subsequence of $A_{2n}, A_{2n-1}, \ldots, A_{2i}$ plus the first (counter-clockwise most) vertex is u. Define the *fixed part* $R^-(u)$ of $C^-(u)$ to be the chain which is a

[8] We cannot guarantee intersection of $d^-(u)$ and $d^+(u)$ because $d^-(u)$ and $d^+(u)$ are discontinuous.

[9] This is clockwise most from the point of view of t.

subset of $C^-(u)$ and whose vertices form the set VERTICES$(C^-(u)) \setminus \{u\}$. Let $r^-(u)$ be the closest point of $R^-(u)$ to t. Let $s^-(u)$ be the last vertex of $R^-(u)$. Define the *fixed part* $R^+(u)$ of $C^+(u)$, $r^+(u)$, and $s^+(u)$ analogously. We say that $C^-(p)$ and $C^-(q)$ are *combinatorially equivalent* if $R^-(p) = R^-(q)$. The analogous definition holds for $C^+(p)$ and $C^+(q)$. We say that two points p and q are *hull equivalent* if $C^-(p)$ and $C^-(q)$ are combinatorially equivalent and $C^+(p)$ and $C^+(q)$ are combinatorially equivalent.

Here we describe a clockwise scan of $E = A_1A_2, \ldots, A_{2n-1}A_{2n}$ which is a modification of Graham's scan [6]. Graham's scan constructs the convex hull of a set of points in linear time once they are in sorted order about an internal point O. As observed in [13], in Graham's scan, "if a point is not a vertex of the convex hull, it is internal to some triangle Opq where p and q are consecutive hull vertices." If points are ordered clockwise about O, then this implies that the point is in the wedge which is on the right of Op and on the left of Oq and the point is on the right of pq. When the scan encounters a triple of consecutive points prq in the ordering about O such that prq is a left turn, it pops vertices off the current hull until convexity is restored. We modify Graham's scan so that it correctly constructs the visible part of a convex chain when O (t in our case) is *external* to the chain. We need only change the left turn test for a triple prq to a right turn test in order to perform a clockwise scan about t. A counter-clockwise scan about t from A_{2n} to A_1 requires only minor modifications to the scan. In both cases, the modified scan has the same asymptotic running time of $O(|U|)$ as Graham's scan. To simplify the remainder of our discussion, we insist that the chain which is built during the modified scan is convex at all times. Hence, we do not add a new point to the chain until all vertices which must be removed have been removed. This does not affect the running time or correctness of the algorithm.

During a given step of one direction of the modified Graham scan, denote by C the current state of the convex chain which is maintained by the scan. We describe a partitioning process which produces a partition G' of G, where the new index of $A_{2\beta}$ is $2\beta'$. The partition can be generated during the modified scan of E by marking edges of $A_{2\alpha-1}A_{2\alpha}, \ldots, A_{2\beta-1}A_{2\beta}$ as vertices are removed from C. In the clockwise scan, suppose we are processing A_{2i}, $\alpha \leq i \leq \beta$, and that $A_gA_hA_{2i}$ is a right turn, where A_gA_h is the last edge of C. Before deleting A_h, extend ray A_gA_h. If it intersects $A_{2i-1}A_{2i}$, then "mark" $A_{2i-1}A_{2i}$ at the intersection point. Note that no marking need be done when processing A_{2i+1} because there is no edge between A_{2i} and A_{2i+1}. Use the same marking process during the counterclockwise scan of $A_{2n}A_{2n-1}, \ldots, A_2A_1$. When the second scan is complete, split the edges of G at the marks to produce the partition $G' = A'_{2\alpha-1}A'_{2\alpha}, \ldots, A'_{2\beta'-1}A'_{2\beta'}$. The full paper shows that G' has $O(|U|)$ edges and can be constructed in $O(|U|)$ time.

Lemma 4.2 *Immediately after the clockwise (counterclockwise) partitioning/modified Graham scan processes vertex A'_j of edge $A'_jA'_{j+1}$ of G', we can obtain, in* $O(1)$ *time: 1) $C^-(A'_j)$ $(C^+(A'_j))$ from C and 2) $R^-(u)$ $(R^+(u))$ and $s^-(u)$ $(s^+(u))$ from C, for $u \in$* INTERIOR$(A'_jA'_{j+1})$ *if j is odd. Furthermore, given $p, q \in$* INTERIOR$(A'_{2i-1}A'_{2i})$ *for $\alpha \leq i \leq \beta'$, p and q are hull equivalent.*

In the full paper, we show that, because the modified Graham scan repeatedly either adds or deletes a single edge from C, Lemma 4.3 below implies that we can update the nearest point in $O(1)$ time, which yields the result below in Lemma 4.4.

Lemma 4.3 *Let Q be a convex polygon and t be a point outside Q. Parameterize the portion of the boundary of Q which is visible to t by the piecewise linear function $\gamma(\tau)$, $\tau \in [0, 1]$, where τ increases clockwise about t. The distance function $d_Q(\tau) = \Delta(t, Q(\gamma(\tau)))$ is unimodal.*

Lemma 4.4 *In* $O(|U|)$ *time it is possible to precompute $r^-(u)$ and $r^+(u)$ for $u \in$* INTERIOR$(A'_{2i-1}A'_{2i})$ *of G', $\alpha \leq i \leq \beta'$, as well as $c^-(u)$ and $c^+(u)$ for each vertex u of G'.*

5 Locating the Intersection Point along an Edge.

Lemma 5.1 *If $L_{\max}$ intersects* INTERIOR$(A'_{2i-1}A'_{2i})$, *then we can find the intersection point in* $O(1)$ *time in a real arithmetic model, or in* $O(\log(\log \epsilon^{-1}))$ *time in a rational arithmetic model with accuracy ϵ.*

Proof: Consider $u \in$ INTERIOR$(A'_{2i-1}A'_{2i})$. The point $c^-(u)$ is either: 1) on $R^-(u)$ (and therefore equal to $r^-(u)$), 2) in the interior of $s^-(u)u$, or 3) equal to u. A similar statement holds for $c^+(u)$. We show that

INTERIOR($A'_{2i-1}A'_{2i}$) contains four subsegments, each having constant $c^-(u)$ and $c^+(u)$ type. Consider $c^-(u)$, w.l.o.g. In the full paper, we show that if type (1) applies to any point in INTERIOR($A'_{2i-1}A'_{2i}$), then either $c^-(u)$ is the same for all points u in INTERIOR($A'_{2i-1}A'_{2i}$) or else there is one point w along it where $c^-(u)$ changes from $s^-(u)$ to a point on $(s^-(u)u]$. The point w, if it exists, is the intersection of INTERIOR($A'_{2i-1}A'_{2i}$) with the line through $s^-(u)$ perpendicular to $s^-(u)t$. The nearest point of $(s^-(u)u]$ to t becomes u when tu is perpendicular to $ts^-(u)$. The locus of points for which tu is perpendicular to $ts^-(u)$ is the circle with diameter $ts^-(u)$. This circle has up to two intersections with INTERIOR($A'_{2i-1}A'_{2i}$), and $c^-(u)$ is at u in between the two points of intersection. Lemma 4.2 and Lemma 4.4 allow us to retrieve $s^-(u)$, $s^+(u)$, $r^-(u)$ and $r^+(u)$ for a given i in O(1) time. From these, the endpoints of the subsegments can be located in O(1) time for a given i, because they require only a constant number of algebraic operations, such as intersecting a line and/or circle with a line.

To find the intersection point on a constant-type subsegment, parameterize the subsegment by the linear function $\gamma(\tau)$, for $\tau \in [0,1]$. Consider $c^-(\gamma(\tau))$. By Lemma 4.1, $d^-(\gamma(\tau))$ is the distance from $c^-(\gamma(\tau))$ to t. In case (2),

$$d^-(\gamma(\tau)) = \frac{(t-q)\times(\gamma(\tau)-q)}{|\gamma(\tau)-q|},$$

a continuous algebraic function. In case (3), $d^-(\gamma(\tau)) = |\gamma(\tau) - t|$; this is also continuous. Analogous formulas hold for $d^+(\gamma(\tau))$. We can intersect two pieces using algebraic operations. The case which dominates the running time occurs when both $c^-(\gamma(\tau))$ and $c^+(\gamma(\tau))$ are of type (2). In this case, one can find the intersection point by proceeding algebraically at first and then numerically. Setting $d^+(\gamma(\tau)) = d^-(\gamma(\tau))$, squaring, and clearing fractions produces a fourth-degree polynomial equation in τ. The intersection point for $\tau \in [0,1]$ is the root (in this interval) of the fourth-degree polynomial. Under a real arithmetic model, we can represent the root *exactly* as an algebraic number. This requires manipulating algebraic numbers. To avoid this, one can operate on a nearby rational approximation to the root. This can be obtained using, for example, Newton's method. The quadratic convergence of Newton's method adds only $\log(\log \epsilon^{-1})$ to the running time of the algorithm, where ϵ is the accuracy. ■

Theorem 5.2 *The line $L_{\max}$ can be computed in* $O(|U|)$ *time in a real arithmetic model and in* $O(|U| + \log(\log \epsilon^{-1}))$ *time in a rational arithmetic model, where ϵ is the accuracy.*

References

[1] K. Daniels. *Containment Algorithms for Nonconvex Polygons with Applications to Layout.* PhD thesis, Harvard University, 1995.

[2] K. Daniels and V. J. Milenkovic. Multiple Translational Containment: Approximate and Exact Algorithms. In *Proceedings of the 6th Annual ACM-SIAM Symposium on Discrete Algorithms,* pages 205–214, 1995.

[3] D.T.Lee. On finding the convex hull of a simple polygon. *Int'l J. Comput. and Infor. Sci.*, 12(2):87–98, 1983.

[4] H. El Gindy and D. Avis. A Linear Algorithm for Computing the Visibility Polygon from a Point. *Journal of Algorithms*, 2:186–197, 1981.

[5] R. Graham and F. Yao. Finding the convex hull of a simple polygon. *Journal of Algorithms*, 4(4):324–331, 1983.

[6] R. L. Graham. An efficient algorithm for determining the convex hull of a finite planar set. *Info. Proc. Lett.*, 1:132–133, 1972.

[7] B. Joe and R. B. Simpson. Corrections to Lee's Visibility Polygon Algorithm. *BIT*, 27:458–473, 1987.

[8] D. T. Lee. Visibility of a Simple Polygon. *Computer Vision, Graphics, and Image Processing*, 22:207–221, 1983.

[9] D. McCallum and D. Avis. A linear algorithm for finding the convex hull of a simple polygon. *Info. Proc. Lett.*, 9:201–206, 1979.

[10] V. J. Milenkovic. Multiple Translational Containment, Part II: Exact Algorithms. *Algorithmica, special issue on Computational Geometry in Manufacturing, accepted, subject to revisions.*

[11] V. J. Milenkovic and K. Daniels. Translational Polygon Containment and Minimal Enclosure using Geometric Algorithms and Mathematical Programming. Technical Report 25–95, Center for Research in Computing Technology, Division of Applied Sciences, Harvard University, 1995.

[12] V.J. Milenkovic and K. Daniels. Translational Polygon Containment and Minimal Enclosure using Mathematical Programming. *International Transactions in Operational Research (submitted).*

[13] F. Preparata and M. Shamos. *Computational Geometry: An Introduction.* Springer-Verlag, New York, 1985.