Computing the Union of 3-Colored Triangles

Jean-Daniel Boissonnat, Olivier Devillers, Franco P. Preparata

published in International Journal of Computational Geometry & Applications, 1:187-196, 1991

Computing the Union of 3-Colored Triangles

Jean-Daniel.Boissonnats* Olivier Devillers†
Franco P. Preparata†

Abstract

Given is a set $\mathcal{S}$ of n points, each colored with one of $k \geq 3$ colours. We say that a triangle defined by three points of $\mathcal{S}$ is 3-colored if its vertices have distinct colours. We prove in this paper that the problem of constructing the boundary of the union $T(\mathcal{S})$ of all such 3-colored triangles can be done in optimal $O(n \log n)$ time.
Keywords: Convex hull, Legged robot, Motion planning, Stability

1 Introduction

Given is a set $\mathcal{S}$ of n points, each colored with one of $k \geq 3$ colours. We say that a triangle defined by three points of $\mathcal{S}$ is 3-colored if its vertices have distinct colours. We consider in this paper the problem of constructing the boundary of the union $T(\mathcal{S})$ of all such 3-colored triangles.

This problem is motivated by the analysis of the stability of a legged robot [1]. Let k be the number of legs of the robot. We associate to each leg a colour and assume that each leg can reach a finite number of foothold points, colored accordingly. For a given configuration there exists a stable assignment of the legs to the footholds if and only if the center of mass lies inside the union of all 3-colored triangles.

The problem is a subtle generalization of the classic convex-hull problem. Indeed, $T(\mathcal{S})$ is exactly the convex hull $CH(\mathcal{S})$ if, for example, all the points of $\mathcal{S}$ have distinct colours ($k = n$). In spite of the more complicated structure of the underlying geometry, we present an optimal $O(n \log n)$-time algorithm for this problem.

*http://www-sop.inria.fr/geometrica/ INRIA, BP93, 06902 Sophia Antipolis, France.
†Department of Computer Science, Brown University,Providence, RI 02912 (USA)
[0]INRIA, BP 93, 06902 Sophia Antipolis cedex, France. E-mail: First-name.Lastname@sophia.inria.fr . This work was partially supported by ESPRIT Basic Research Action Nr. 3075 (ALCOM), by CNES, and by NSF Grant CCR-89-06419.

The distinct colours are referred to by means of the integers $1, 2, \ldots, k$. In the sequel, we denote $\mathcal{S}_i$ the subset of $\mathcal{S}$ of points with colour i and $\Gamma_i = CH\left(\bigcup_{j \neq i} \mathcal{S}_j\right)$, the convex hull of the points of $\mathcal{S}$ whose colours are distinct from color i.

2 Geometric Preliminaries

2.1 Some geometric properties of $T(\mathcal{S})$

Properties 1-5 are self-evident, and are therefore stated without proof.

Property 1 *Every line segment joining two points with distinct colours (2-colored segment) is an edge of a 3-colored triangle and so it belongs to $T(\mathcal{S})$.*

Property 2 *$T(\mathcal{S})$ is connected.*

Property 3 *The convex vertices of $T(\mathcal{S})$ are original points of $\mathcal{S}$ (vertices of triangles) and its concave vertices are intersections points between pairs of 2-colored segments (edges of triangles).*

Property 4 *If $p \in \mathcal{S}_i$ is a convex vertex of $T(\mathcal{S})$, the two edges of $T(\mathcal{S})$ incident to p are contained in the supporting lines to Γ_i issuing from p.*

Notice that the points where the supporting lines issuing from p intersect Γ_i are not of colour i, so the line segments joining p to these points are 2-colored and thus contained in $T(\mathcal{S})$. Moreover, any other 2-colored line segment between p and a point inside Γ_i is contained in the wedge defined by the two supporting lines to Γ_i.

Property 5 *The vertices of the convex hull $CH(\mathcal{S})$ of $\mathcal{S}$ form a subset of the convex vertices of $T(\mathcal{S})$. They appear in the same order along the boundary of $T(\mathcal{S})$ as along that of $CH(\mathcal{S})$.*

Property 6 *Along the boundary of $T(\mathcal{S})$ there is at most one concave vertex between two consecutive convex vertices.*

Proof: Let p and q be two consecutive convex vertices. If p and q have distinct colours, the line segment pq is included in $T(\mathcal{S})$ and there are no concave vertices between p and q.

Suppose now that p and q have the same colour. Let r be the first concave vertex encountered when marching along the boundary of $T(\mathcal{S})$ from p to q (see Figure 1). If r is not adjacent to q then r belongs also to a 2-colored segment uv whose endpoint u is chosen to belong to the half plane limited by the line defined by p and r and not containing q. Clearly $u \neq p$ and $v \neq q$. If either of the two line segments pv or qu were contained in $T(\mathcal{S})$, then r would not be a

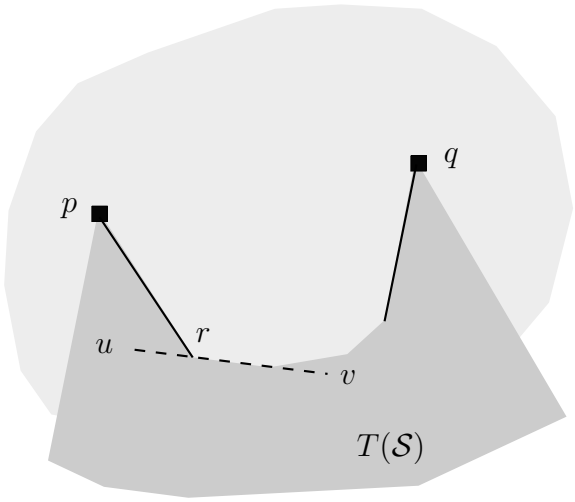

Figure 1: It is impossible to have two consecutive concave vertices.

vertex of $T(\mathcal{S})$. But because u and v have distinct colours, at least one of these two line segments is 2-colored and belongs to $T(\mathcal{S})$. This contradicts Property 1 and proves the property. ◇

The following property is an immediate consequence of Property 6.

Property 7 *The number of edges of $T(\mathcal{S})$ is at most $2n - k$.*

This bound is tight and can be achieved if all n points are vertices of $CH(\mathcal{S})$ and the boundary consists of k monochromatic sequences of vertices.

2.2 Extremal 2-colored segments

A 2-colored segment l is said to be *extremal and anchored at p* if:

(i) it is incident to p;

(ii) the line containing l leaves only points with the same colour as p – if any – in one of the half-planes it defines.

It is immediate that an extremal segment anchored at $p \in S_j$ belongs to a supporting line from p to Γ_j. It follows that $p \in S_j$ has anchored extremal segments if and only if p is external to Γ_j and that if p has an anchored extremal segment, it has two such segments, called *left* and *right* (with respect to an observer standing at p and facing Γ_j) and respectively denoted $L(p)$ and $R(p)$.

It follows from Properties 1 and 2 that the boundary of $T(\mathcal{S})$ is also the boundary of the unbounded cell of the arrangement of the extremal 2-colored line segments anchored at points of $\mathcal{S}$.

Let $\mathcal{S}'$ be the set of vertices of $CH(\mathcal{S})$. We denote $B(\mathcal{S}')$ the boundary of the unbounded cell of the arrangement of the extremal 2-colored line segments of $\mathcal{S}$ anchored at points of $\mathcal{S}'$.

Property 8 *$B(\mathcal{S}')$ is a simple polygon with at most $2n-k$ edges. Its vertices appear in a cyclic order consistent with the order along the boundary of $CH(\mathcal{S})$.*

Proof: We first prove that $B(\mathcal{S}')$ is connected. Indeed, if p and q are two successive points of $\mathcal{S}'$ in clockwise order, then $L(p) \cap R(q) \neq \emptyset$ (if p and q are of different colours $L(p) = R(q) = pq$).

Next we observe that the convex vertices of $B(\mathcal{S}')$ are points of $\mathcal{S}'$ and its concave vertices are intersection points between pairs of extremal 2-colored segments. The proof of Property 6 and thus of Property 7 apply to $B(\mathcal{S}')$ without change. This achieves the proof. ◇

We conclude with the following important property.

Property 9 *The polar order of the color-i convex vertices of $T(\mathcal{S})$ with respect to any point of $\mathcal{S}$ of colour $j \neq i$ is consistent with their order along the boundary of $T(\mathcal{S})$.*

Proof: The property trivially holds for the convex vertices of $T(\mathcal{S})$ belonging to $CH(\mathcal{S})$. Therefore, let us consider a subsequence $C = (p_1, \ldots, p_r)$ of convex vertices of $T(\mathcal{S})$ occurring between two convex vertices p and q consecutive along $CH(\mathcal{S})$. All vertices of C have the same colour, say i, coincident with the common colour of p and q (otherwise such vertices would be linked to p and q by 2-colored segments and would not be convex). Consider two other points, say u_1 and u_2, of respective colours $j_1 \neq i$ and $j_2 \neq i$, and assume for a contradiction that p_s precedes $p_t, 1 \leq s \neq t \leq r$ with respect to u_1 and p_s follows p_t with respect to u_2 (in polar order). This implies that u_1 and u_2 belongs to distinct halfplanes defined by the line containing $p_s p_t$, so that p_s and p_t cannot be convex vertices of $T(\mathcal{S})$. ◇

3 Algorithm

We observe that p is a convex vertex of $T(\mathcal{S})$ only if p has anchored extremal segments. Thus, the set $\mathcal{S}^* \subseteq \mathcal{S}$ of points with anchored extremal segments is a superset of the set of convex vertices $C(\mathcal{S})$ of $T(\mathcal{S})$ and the latter is a superset of the set $\mathcal{S}'$ of vertices of $CH(\mathcal{S})$.

$$\mathcal{S}' \subseteq C(\mathcal{S}) \subseteq \mathcal{S}^*$$

Our algorithm will identify a subset of $\mathcal{S}^*$, which is itself a superset of $C(\mathcal{S})$, and subsequently filter it to obtain $C(\mathcal{S})$.

To gain efficiency, the algorithm computes, in Step 1 a data structure that organizes the set of colours. Then it computes in Step 2, the extremal segments

of each $p \in \mathcal{S}'$; next, in Steps 3 and 4, the algorithm obtains the points of $\mathcal{S}^* - \mathcal{S}'$ lying outside $B(\mathcal{S}')$, and determines their anchored extremal segments. Finally, Step 5 will obtain $C(\mathcal{S})$ and construct the boundary of $T(\mathcal{S})$.

We now describe in detail the five steps of the algorithm.

Step 1 : Building the data structure The data structure is a balanced binary tree $\mathcal{T}$ that organizes the set of colours. Each leaf of the tree is associated with a distinct colour and each internal node is associated with the union of the colours of its descendant nodes.

A point p of $\mathcal{S}$ is assigned to all the nodes associated with its colour. As a consequence, each point is assigned to exactly one node in each level of the tree and thus appears in at most $\lceil \log_2 k \rceil$ nodes of the tree. The "left (resp. right) subtree of $\mathcal{T}$" means the "left (resp. right) subtree of the root of $\mathcal{T}$". In the sequel $\mathcal{S}(V)$ will denote the set of points assigned to node V of $\mathcal{T}$.

This data structure has some similarity with the segment tree: here the binary tree serves the purpose of splitting the set of colours into some canonical subsets corresponding to nodes of the tree.

The main fact we will use is that the set of points $\bigcup_{j \neq i} \mathcal{S}_j$ is partitioned into $O(\log k)$ sets, associated with the sibling nodes of the nodes pertaining to the colour i. (These nodes form a path from the root to a leaf of $\mathcal{T}$.)

For each node of the tree, we compute the convex hull of the set of points assigned to that node (for short, the "hull of the node"). For the leaf associated to colour i, $CH(\mathcal{S}_i)$ is computed using any classical algorithm in time $O(n_i \log n_i)$ [2]. This is done for all $i \in \{1, \ldots, k\}$, which takes time $\sum_{i=1}^{k} O(n_i \log n_i) = O(n \log n)$.

The hull of an internal node is obtained by merging the hulls of its children, which can be done in time proportional to the sum of the sizes of the hulls of the children. The internal nodes of $\mathcal{T}$ are thus computed in $O(n)$ time per level of the tree. Since $k \leq n$, we conclude that Step 1 takes $O(n \log n)$ time.

Step 2 : Computing $B(\mathcal{S}')$ For each $p \in \mathcal{S}' \cap \mathcal{S}_j, j = 1, \ldots k$, we wish to compute its two supporting lines to Γ_j. Since the points of $\cup_{i \neq j} \mathcal{S}_i$ are partitioned into $O(\log k)$ sets, we compute the pair of supporting lines from p to the convex hulls of each such set, and determine the supporting lines (i.e., the extremal segments) by a simple tournament among these $O(\log k)$ competing pairs.

The $|\mathcal{S}'| O(\log k)$ competing pairs of supporting lines are collectively computed as follows. For each pair of sibling nodes in $\mathcal{T}$, say V and W, we determine for each $q \in \mathcal{S}' \cap \mathcal{S}(V)$ the supporting lines to $CH(\mathcal{S}(W))$ (and *vice versa*) by a forward march along the boundaries of $CH(\mathcal{S}(V))$ and $CH(\mathcal{S}(W))$. We claim that as we march forward, say clockwise, along the boundary of $CH(\mathcal{S}(V))$, visiting $\mathcal{S}' \cap \mathcal{S}(V)$, the corresponding clockwise march along the boundary of $CH(\mathcal{S}(W))$ never steps backward. This is simply due to the fact that $CH(\mathcal{S}(W)) \subset CH(\mathcal{S})$ and that we only consider points of $\mathcal{S}(V)$ which also belongs to $CH(\mathcal{S})$ (see Figure 2). Thus the determination of the supporting

lines from each $q \in \mathcal{S}' \cap \mathcal{S}(V)$ to $CH(\mathcal{S}(W))$ takes time proportional to the sum of the number of vertices of the two hulls.

Scanning the pairs of sibling nodes takes $O(n)$ time per level, thus $O(n \log k)$ time in total. Selecting the actual pair of extremal lines of a vertex p of $CH(\mathcal{S})$ among the $O(\log k)$ competing pairs associated to the distinct occurrences of p in $\mathcal{T}$ can be done in $O(\log k)$ time.

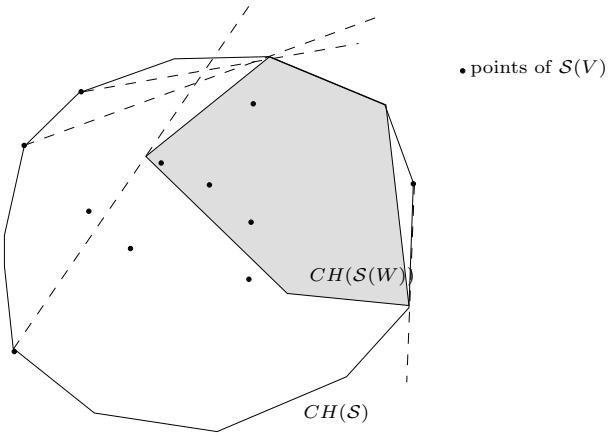

Figure 2: The scan cannot go backward.

In that way we identify the extremal segments $(L(p), R(p))$ of each $p \in \mathcal{S}'$. Because any two consecutive vertices of $CH(\mathcal{S})$ are consecutive convex vertices of $B(\mathcal{S}')$ with at most one concave vertex between them (Property 8), a simple march along $CH(\mathcal{S})$ achieves the construction of $B(\mathcal{S}')$ in $O(n)$ time. Thus Step 2 uses $O(n \log k)$ time.

Step 3 : Obtaining a superset of $C(\mathcal{S})$ Due to Property 9, the order along a subsequence of convex vertices of $T(\mathcal{S})$ of a given colour is consistent with the polar order of these vertices with respect to a point of $\mathcal{S}$ of a different colour. Therefore, we choose a point $r^\star$ originally assigned to the right subtree of $\mathcal{T}$ as a reference point for all the points (colored differently from the reference point) assigned to the left subtree of $\mathcal{T}$.

In order to sort the points according to the polar order around $r^\star$, we proceed as follows. For each leaf L of the left subtree, we sort the points assigned to L around $r^\star$, and by a simple scan we identify the vertices of $B(\mathcal{S}')$ and the points lying outside $B(\mathcal{S}')$: these points are potential vertices of $T(\mathcal{S})$. This can be done in total time $O(n \log n)$.

This computation is done symmetrically for the right subtree of $\mathcal{T}$. So we get for each $\mathcal{S}_j$ an ordered set of potential vertices of colour j. As mentioned earlier, these lists contain all the vertices of $T(\mathcal{S})$ but, in general, also some additional points that will be filtered out in Steps 4 and 5. We let $\mathcal{S}''$ denote the set of potential vertices obtained by this step.

By recursively merging the lists of sibling nodes we get, for each internal node (except the root), an ordered list of potential vertices. This is done in $O(n)$ time per level of the tree. Thus Step 3 takes $O(n \log n)$ time in total.

Step 4 : Computing the extremal segments of the potential vertices

We now compute the extremal segments anchored at all the points in $\mathcal{S}''$. This is done in a way analogous to the working of Step 2. The main difference between Steps 2 and 4 is that set $\mathcal{S}''$ rather than $\mathcal{S}'$ is being considered.

Specifically, consider two sibling nodes V and W of $\mathcal{T}$. Let p and q be two consecutive points of $\mathcal{S}(V) \cap \mathcal{S}'$. If p and q have distinct colours, pq is an extremal segment anchored at both p and q.

Otherwise, the points of $\mathcal{S}(V) \cap \mathcal{S}''$ occurring between p and q, say $p_1, p_2, \ldots, p_r$, have all the same colour as the common colour of p and q. The four supporting lines from p and q to $CH(\mathcal{S}(W))$ have been already computed in Step 2. We now scan the sequence $p_1, \ldots, p_r$ and compute the supporting lines of these points. Differently from Step 2, since the points of $\mathcal{S}(V) \cap \mathcal{S}''$ are not necessarily on the boundary of a convex polygon, the march along $CH(\mathcal{S}(W))$ is not guaranteed to be always forward. The situation is illustrated in Figure 4, where the point of support u_3 occurs between points of support u_1 and u_2; however, since p_1, p_2, and p_3 are in the correct polar order, p_2 belongs to the wedge defined by the two supporting lines $L(p_3)$ and $R(p_3)$ of p_3, and can be discarded from the set of potential vertices of $T(\mathcal{S})$.

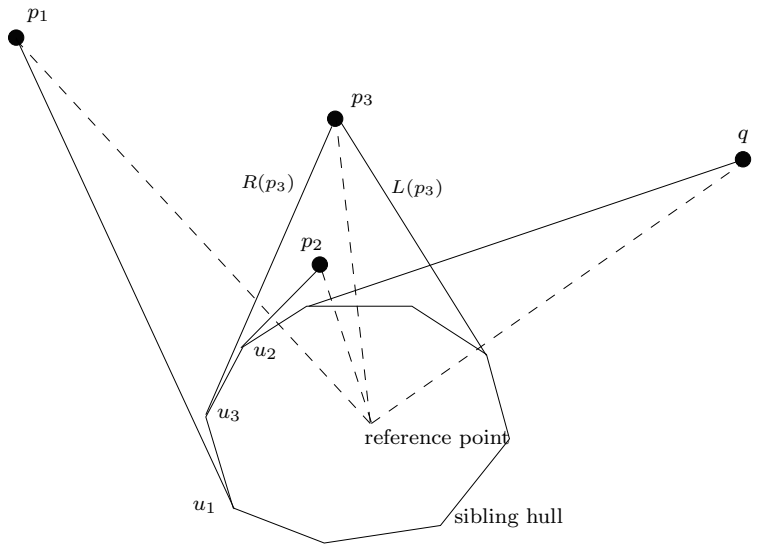

Figure 3: The scan can go backward.

As a consequence, at each step of the scan, either we proceed forward and process a new point or we delete one point from set $\mathcal{S}''$. Since $|\mathcal{S}''| < n$, it

follows that the scan takes time proportional to $|\mathcal{S}(V) \cap \mathcal{S}''|$ (a similar argument is used in the analysis of the well-known Graham scan for computing convex hulls) [3, 2]. Thus Step 4 costs $O(n)$ time per level which is $O(n \log k)$ in total.

Step 5 : Constructing $T(\mathcal{S})$ At this stage, we know for each $p \in \mathcal{S}' \cup \mathcal{S}''$ the pair $(L(p), R(p))$ of extremal anchored segments. We also know, for each subset of $\mathcal{S}' \cup \mathcal{S}''$, pertaining respectively to the left or right subtree of $\mathcal{T}$, a cyclic order, say clockwise, around a chosen reference point.

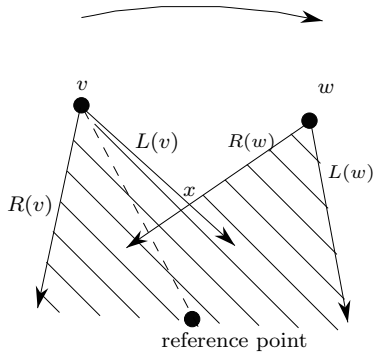

Figure 4: For the illustration of step 5.

Let p and q be two consecutive vertices of $CH(\mathcal{S})$. If p and q have distinct colours, segment pq is extremal and anchored at both p and q. Otherwise, the ordered set of potential vertices between p and q, $p_1, p_2, \ldots, p_r$, have all the same colour. We now consider the construction of the portion of the boundary of $T(\mathcal{S})$ contained between p and q. The technique, again akin to the Graham scan, scans $p_1, \ldots p_r$ clockwise. Let C be a doubly-connected list of points initialized with $p_1, \ldots, p_r$. At the end, C will represent the sequence of vertices of $T(\mathcal{S})$ occurring between p and q.

Let v be the current point of C under consideration and w its successor in the list. At the beginning, $v = p_1$ and $w = p_2$. It is to be noticed that both w and $L(v)$ necessarily lie in the half-plane to the left of the line containing v and the reference point $r^\star$, oriented from v to $r^\star$.

If w belongs to the right half plane limited by $L(v)$ directed away from v, then w cannot be a vertex of $T(\mathcal{S})$ and is deleted from C. Otherwise, if $R(w)$ intersects $L(v)$ at some point x, we insert x immediately after v in C and advance to w ; else, $R(w)$ must intersect the constructed polygonal chain between p and v. We delete v from C and regress, so that the point now preceding w becomes the current point.

The scan halts when the whole chain $p_1, \ldots, p_r$ has been scanned, i.e., when $v = q$. By repeating this procedure for each pair of consecutive vertices of $CH(\mathcal{S})$, we obtain $T(\mathcal{S})$.

At each step of the scan, either we advance and process a new point or we

regress but, in that case, we delete one point from C. As the total size of all lists is $O(n)$, Step 5 takes $O(n)$ time in total.

4 Analysis

The correctness of the algorithm is readily proved by observing that all the convex vertices of $T(\mathcal{S})$ have been computed as well as their extremal segments.

We deduce from the analysis of the different steps of the algorithm that $T(\mathcal{S})$ can be computed in $O(n \log n)$ time.

We prove the optimality of the time bound by transforming "ordered convex hull" to the present problem. Given a set of points with $S = \{p_1, p_2, \ldots, p_n\}$, $n \geq 3$, we give colour 1 to p_1, colour 2 to p_2, and colour 3 to all other points. We compute $T(\mathcal{S})$ for $\mathcal{S}$ so colored, and obtain a simple polygon, star-shaped with respect to p_1 and p_2, so that its convex hull is computed trivially in time $O(n)$. Since the transformation takes $O(n)$ time, the $\Omega(n \log n)$ lower bound for computing $T(\mathcal{S})$ is immediate.

For simplicity of presentation, the implementation discussed above assumes that for each node V of $\mathcal{T}$ we actually store the corresponding convex hull, and that for each point of $\mathcal{S}' \cup \mathcal{S}''$ we store the $O(\log k)$ competing pairs of supporting lines. This implementation would correspond to a storage requirement $O(n \log k)$. In reality, storage can be reduced to $O(n)$ by maintaining the sets $\mathcal{S}(V)$ and their convex hulls only for the leaves of $\mathcal{T}$ and by processing pairs of sibling nodes, level by level of $\mathcal{T}$, when required, without maintaining the structures pertaining to the heretofore processed levels of $\mathcal{T}$.

More precisely, in Step 1, we now only store convex hulls of the leaves and compute $\mathcal{S}'$. The construction of the convex hulls of the internal nodes (which was done once for all in Step 1) is now performed on the fly during Step 2 and again during Step 4. Similarly the computation of the competing pairs of supporting lines (which was done once for all in Step 3) is now performed on the fly during Step 4 at the same time as the construction of the convex hulls.

We can then summarize the preceding results as follows.

Theorem 1 *The union of 3-colored triangles of a set of n points colored with $k \geq 3$ colours can be computed in optimal $\Theta(n \log n)$ time using $\Theta(n)$ space.*

5 Conclusion

We have proved that the union of the 3-colored triangles of a set of n points colored with k colours is a simple polygon with at most $2n - k$ edges that can be computed in optimal time $\Theta(n \log n)$ using $\Theta(n)$ space.

Several further extensions remain to be considered. In particular, what is the complexity of the union of $(d+1)$-colored simplices in d space and the time required to construct their union?

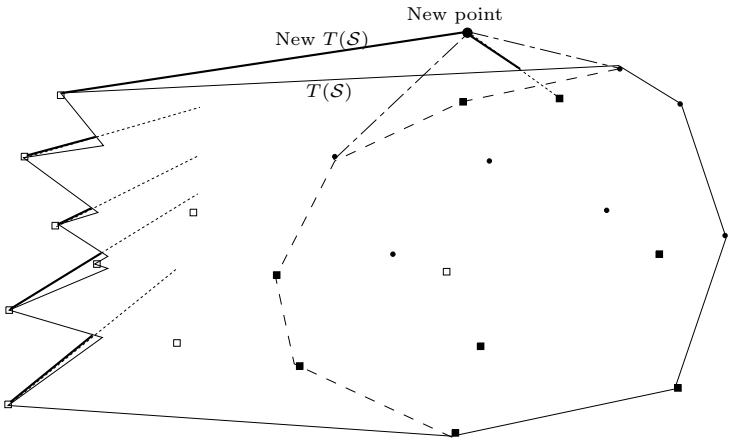

Figure 5: The insertion of a new point.

The dynamic version of the problem would also be of great practical interest : when the robot moves the footholds change. Unfortunately, as shown in Figure 5 the changes induced by the insertion of the deletion of a point can be as large as $\Omega(n)$ and not local. We let as an open question whether there exist a dynamic algorithm sensitive to the size of the change.

Acknowledgements

The authors thank the anonymous referee whose comments helped improving the presentation of the paper.

References

[1] J.D. Boissonnat and O. Devillers. *Motion Planning of Legged Robots*. Technical Report, Institut National de Recherche en Informatique et Automatique, (France), 1991. In preparation.

[2] F.P. Preparata and M.I. Shamos. *Computational Geometry : an Introduction*. Springer-Verlag, 1985.

[3] R. Graham. An efficient algorithm for determining the convex hull of a finite planar set. *Information Processing Letters*, 1:132–133, 1972.