VERIFICATION OF PARTITIONS OF 2d AND 3d OBJECTS

L. Palios

23 – 2001

Preprint, no 23 – 01 / 2001

Department of Computer Science
University of Ioannina
45110 Ioannina, Greece

Verification of Partitions of 2d and 3d Objects

Leonidas Palios
Dept. of Computer Science, Univ. of Ioannina, Greece
palios@cs.uoi.gr

Abstract: We consider the problem of deciding whether a given collection of polygons forms (i) a partition or (ii) a cell complex decomposition of a given polygon. We also consider the 3-dimensional counterpart of the above problem, i.e., deciding whether a given collection of polyhedra forms (i) a partition or (ii) a cell complex decomposition of a given polyhedron. We describe simple $O(n \log n)$-time and $O(n)$-space algorithms for these problems, where n is the total size of the input. If, in the input, vertices are referenced by means of indices to a global list of distinct vertices, then our cell complex decomposition verification algorithms run in $O(n)$ time.

Keywords: checker, verification, partition, cell complex, polygon, polyhedron.

1. Introduction

In recent years, there has been growing interest in algorithms for the verification of geometric structures, which is the result of both the obvious benefits from such algorithms, and the fact that the newer and more efficient algorithms are more and more complicated which makes it easier for programming errors to occur in their implementation. Therefore, verification procedures are needed in order to check the programs' results. Since the verification procedures are programs themselves, they must be simple enough so that their correct implementation can be easily verified.

Usually, the authors of an algorithm for computing some geometric object describe properties of the object which can be used to verify the correctness of the computation (see e.g. [1] and [4] for 2-dimensional and 3-dimensional triangulations of point sets). Often, however, more machinery is needed. The first verification algorithms ("checkers") were provided by Mehlhorn *et al.* who defined the properties that a checker should have (correctness, simplicity, efficiency) and described checkers for convex polyhedra and convex hulls [5]; they have also studied checkers for trapezoidal decompositions, planar point location structures for trapezoidal decompositions, and Voronoi and Delaunay diagrams. Devillers *et al.* extended the notion of checkers characterizing them with respect to whether they need a certificate to help them in the verification task and to the kind of the certificate needed (arbitrary certificate, topology certificate, no certificate), and provided checkers for convex polytopes in two and higher dimensions, and for various types of planar subdivisions, such as triangulations, Delaunay triangulations, and convex subdivisions [3]. Their algorithms

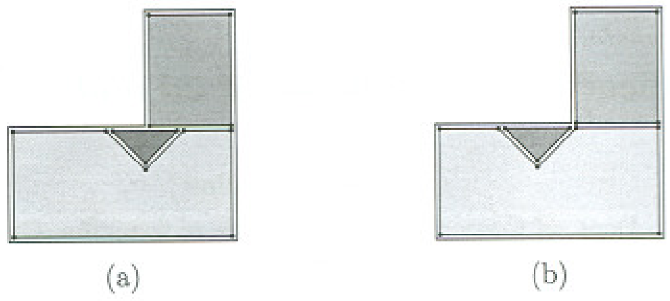

Figure 1: (a) a partition which is not a cell complex decomposition; (b) a cell complex decomposition.

run in linear time, assuming that the input is a 2-dimensional or 3-dimensional geometric graph with the necessary ordering of the graph.

In this paper, we present simple and efficient verification algorithms for partitions and cell complex decompositions of simple polygons and polyhedra. The algorithms rely on appropriately matching the edges of the two-dimensional decompositions and the facets of the three-dimensional decompositions, and run in $O(n \log n)$ time using $O(n)$ space, where n is the total size of the input. If, in the input, vertices are referenced by means of indices to a global list of distinct vertices, then our cell complex decomposition verification algorithms run in optimal $O(n)$ time.

The paper is structured as follows. In Section 2, we review the terminology that we will be using. In Section 3, we present verification algorithms for a partition and a cell complex decomposition of a polygon and in Section 4 verification algorithms for the three-dimensional counterpart, i.e., for a partition and a cell complex decomposition of a polyhedron. Section 5 concludes the paper with a summary of our results and open problems.

2. Terminology

A *simple polygon* is the part of the plane bounded by one closed non-self-intersecting polygonal line. This polygonal line is the *boundary* of the polygon which separates the *interior* of the polygon from its *complement*. The boundary of a polygon P is denoted by ∂P and consists of a collection of relatively open sets, the *faces* of P, which are called *vertices* and *edges* if their affine closures have dimension 0 and 1 respectively. The notion of a *simple polyhedron* is defined in the three-dimensional space in a fashion similar to that of a simple polygon in two-dimensional space. The boundary of a polyhedron consists of *vertices*, *edges*, and *facets* if their affine closures have dimension 0, 1, and 2 respectively.

Let P be a simple polygon (polyhedron) and $\mathcal{C}$ be a collection of simple polygons (polyhedra resp.). Then, $\mathcal{C}$ defines a *partition* of P if and only if the union of the elements of $\mathcal{C}$ equals P and any two elements of $\mathcal{C}$ either do not intersect or their intersection is a subset of their boundaries. The collection $\mathcal{C}$ defines a *cell complex decomposition* of P if and only if $\mathcal{C}$ is a partition of P and additionally the intersection of any two elements A and B of $\mathcal{C}$ is either the empty set or it is a face of both A and B. Figure 1(a) depicts a partition of an L-shaped polygon which is not a cell complex decomposition, while Figure 1(b) depicts a cell complex decomposition of the same polygon.

Note: In the following, we will deal with simple polygons and simple polyhedra only. Therefore, we will be using the terms polygon and polyhedron to mean simple polygon and simple polyhedron.

3. The Two-dimensional Case

We assume that each polygon in the input is described by the sequence of its vertices along its boundary, where each vertex is given by its coordinates.

3.1. Verification of a partition of a polygon.

Our verification algorithm relies on the following lemma.

Lemma 3.1. *Let P and $\mathcal{C}$ be a polygon and a collection of polygons respectively. Additionally, let $E_I = \{e - \partial P \mid e$ is an edge of a polygon in $\mathcal{C}\}$ and $E_B = \{e \cap \partial P \mid e$ is an edge of a polygon in $\mathcal{C}\}$. Then, the collection $\mathcal{C}$ forms a partition of P if and only if*

(i) the set E_I of (parts of) edges of the polygons in $\mathcal{C}$ can be partitioned into two sets S and S' such that

- $\bigcup_{s \in S} closure(s) = \bigcup_{s \in S'} closure(s)$,
- *the closures of the segments in S and S' define a partition of $\bigcup_{s \in S} closure(s)$, and*
- *for each (part of) edge e in S (resp. S'), if $S'[e]$ (resp. $S[e]$) is the set of segments of S' (resp. S) that are collinear with and intersect e, then, locally around e, the interior of the polygon in $\mathcal{C}$ with edge e and the interiors of the polygons in $\mathcal{C}$ whose edges contributed the elements of $S'[e]$ (resp. $S[e]$) lie on opposite sides of the line supporting e;*

(ii) the closures of the segments in E_B form a partition of the boundary ∂P of P, and for each (part of) edge e in E_B, locally around e, the interior of the polygon in $\mathcal{C}$ with e as an edge and the interior of P lie on the same side with respect to the line supporting e.

Proof: If $\mathcal{C}$ is a partition of the polygon P, then the above two conditions clearly hold (see Figure 1(a) for an example). Let us now prove the converse; we assume that the above two conditions hold and it suffices to show that no point in the complement of P belongs to any polygon in $\mathcal{C}$ and that each point in $interior(P) - \left(\bigcup_{Q \in \mathcal{C}} \partial Q\right)$ (i.e., p is a point of the interior of P that does not belong to the boundary of any polygon in $\mathcal{C}$) belongs to exactly one polygon in $\mathcal{C}$.

Condition (i) implies that if we overlay the two partitions defined by S and S', each subsegment s of the overlay belongs to exactly two edges, say, e and d, such that, locally around s, the interiors of the polygons in $\mathcal{C}$ with edges e and d (let them be P_e and P_d respectively) lie on opposite sides of the line supporting s. Let p be a point of s, and let $N(p)$ be a disc centered at p which is small enough that it does not intersect any other edge except for e and d. Then, the segment s partitions $N(p)$ into two half-discs, one belonging to P_e and the other belonging to P_d. Consider two points q and q' such that $q \in interior(P_e) \cap N(p)$ and $q' \in interior(P_d) \cap N(p)$. Then, points q and q' belong to the same number of polygons in $\mathcal{C}$: q belongs to P_e and some k other polygons, q' belongs to P_d and the same k other polygons. In a similar fashion, condition (ii) implies that for any point p belonging to a segment s in E_B (note that $s \subseteq \partial P$), any point $q \in N(p) - P$ belongs to 1 fewer polygons in $\mathcal{C}$ than any point $q' \in N(p) \cap interior(P)$.

We first show that no point in the complement of P belongs to any polygon in $\mathcal{C}$. Let p be such a point; then there exists a non-self-intersecting path from p to infinity which does not intersect ∂P. Moreover, we can assume without loss of generality that the path does not go through any vertex of the given polygons. Since the path does not intersect ∂P, it intersects, if any, only segments in E_I. The fact that the polygons in $\mathcal{C}$ do not extend to infinity (i.e., the point at infinity does not belong to any polygons in $\mathcal{C}$) and the earlier observation regarding segments of E_I imply that the point p does not belong to any polygon in $\mathcal{C}$. Thus, the point in the complement of P do not belong to any polygon in $\mathcal{C}$.

Next, consider a point p in the interior of P which does not belong to the boundary of any polygon in $\mathcal{C}$. Then, there exists a non-self-intersecting path from p to a point q in the complement of P which avoids all vertices and crosses the boundary of P exactly once. As the path crosses segments of E_I then the number of polygons to which points of the path on either side of the segment belong remains the same, whereas as the path crosses the boundary of P then the number of polygons to which a point of the path in the interior of P belongs is 1 more than the corresponding number for a point of the path in the complement of P, which is 0 as we showed earlier. Hence, point p belongs to exactly one polygon in $\mathcal{C}$. Therefore, we have shown that the polygons in $\mathcal{C}$ define a partition of P. ∎

The partition verification algorithm applies Lemma 3.1. It uses an array A of size equal to the total number of edges of the polygons in $\mathcal{C}$ and of the polygon P; the array is initially empty. The algorithm works as follows:

2d Partition Verification Algorithm

1. We orient the boundary of polygon P and the boundaries of the polygons in $\mathcal{C}$ in a compatible fashion (e.g., the order of vertices corresponds to a counterclockwise traversal of the boundary).

2. For each polygon Q in $\mathcal{C}$ do
```
        for each edge $\overrightarrow{uv}$ of $Q$ do
            if $u$ is lexicographically smaller than $v$
            then we add the entry $(u, v, +1)$ to $A$;
            else we add the entry $(v, u, -1)$ to $A$;
```

3. For each edge $\overrightarrow{uv}$ of P do
```
        if $u$ is lexicographically smaller than $v$
        then we add the entry $(u, v, -1)$ to $A$;
        else we add the entry $(v, u, +1)$ to $A$;
```

4. We sort the entries $(u_i, v_i, \pm 1)$ of the array A by slope of the line $u_i v_i$ and, in case of ties, lexicographically taking into account the coordinates of the vertices u_i and v_i.

5. For each slope separately, we traverse the related entries of A verifying that those with "+1" and those with "−1" partition the same set of segments. If this terminates successfully, then the given collection $\mathcal{C}$ of polygons defines a partition of the given polygon P, otherwise it does not.

Time complexity. Clearly, Steps 1, 2, and 3 can be completed in time linear in the total size n of the input, while Step 4 requires $O(n \log n)$ time. Step 5 requires linear time: for each slope, the sorting of the entries implies that each new entry with "+1" is either the extension of the latest entry with "+1" or starts a new segment of the union of the "+1"-entries, and similarly for the "-1"-entries. The algorithm uses linear space. Thus, we have the following result:

Theorem 3.1. *Let P and $\mathcal{C}$ be a polygon and a collection of polygons respectively of total size n. Then, it can be determined whether the collection $\mathcal{C}$ forms a partition of P in $O(n \log n)$ time using $O(n)$ space.*

3.2. Verification of a cell complex decomposition of a polygon.

Our verification algorithm relies on the following lemma.

Lemma 3.2. *Let P and $\mathcal{C}$ be a polygon and a collection of polygons respectively. Then, the collection $\mathcal{C}$ forms a cell complex decomposition of P if and only if the set of edges of the polygons in $\mathcal{C}$ can be partitioned into two sets E_I and E_B such that*

(i) for each edge e in E_I there exists exactly one other edge, say d, in E_I such that $e = d$ and, locally around e, the interiors of the polygons in $\mathcal{C}$ with edges e and d lie on opposite sides of the line supporting e;

(ii) the edges in E_B form a partition of the boundary ∂P of P and for each edge e in E_B, locally around e, the interior of the polygon in $\mathcal{C}$ with edge e and the interior of P lie on the same side with respect to the line supporting e.

Proof: If $\mathcal{C}$ is a cell complex decomposition of P then the above two conditions clearly hold (see Figure 1(b)) provided that we show that in a cell complex there cannot be an edge of a polygon in $\mathcal{C}$ that has non-empty intersections with both the boundary ∂P and the interior of P. Let e be such an edge (of a polygon Q in $\mathcal{C}$) and suppose that a segment au of e belongs to ∂P whereas a segment ub belongs to the interior of P; clearly, u is a vertex of P. Since ub belongs to the interior of P and because $\mathcal{C}$ defines a cell complex decomposition, there exists a polygon Q' in $\mathcal{C}$ other than Q which intersects Q along ub. But then, their intersection is a proper subset of the edge e, in contradiction to $\mathcal{C}$ defining a cell complex decomposition.

Let us now prove the converse. It is easy to see that if the two conditions of the lemma hold, then so do the conditions of Lemma 3.1. Therefore, $\mathcal{C}$ defines a partition of P. The proof will be complete if we show that the intersection of any two polygons in $\mathcal{C}$ is either empty or a face of both polygons. If the intersection of the two polygons is a single point, then clearly it is a vertex of both of them. (Suppose for contradiction that two polygons A and B of $\mathcal{C}$ intersect at a point t of the edge e of A. Clearly, the intersections of A and B with a small enough disc $N(t)$ centered at t share only t. Then, condition (i) implies that there exists another polygon in $\mathcal{C}$, let it be B', with an edge coinciding with e; then, $A \cap N(t)$ and $B' \cap N(t)$ partition $N(t)$, which implies that B and B' share interior points, a contradiction.) Suppose now that the intersection is a segment; then it must be a subset of an edge of either polygon. Then, condition (ii) implies that this segment cannot belong to an edge in E_B; equivalently, the segment belongs to the interior of P. Then, condition (i) implies that the segment has to be an edge of both polygons, as desired. ▮

The cell complex verification algorithm applies Lemma 3.2. It too uses an array A of size equal to the total number of edges of the polygons in $\mathcal{C}$, which is initially empty. The algorithm works as follows:

2d Cell Complex Verification Algorithm

1. We collect all the vertices and assign to each one of them a distinct positive integer $id(\ )$.
2. We orient the boundary of polygon P and the boundaries of the polygons in $\mathcal{C}$ in a compatible fashion.
3. For each polygon Q in $\mathcal{C}$ do

```
for each edge $\overrightarrow{uv}$ of $Q$ do
    if $u$ is lexicographically smaller than $v$
    then we add the entry $(id(u), id(v), +1)$ to $A$;
    else we add the entry $(id(v), id(u), -1)$ to $A$;
```

4. We sort the array A lexicographically.
5. We traverse the sorted array A deleting the matching entries (i.e., $(k, k', -1)$ and $(k, k', +1)$), which now appear next to each other.
6. We collect the unmatched entries and by applying a depth-first traversal we form the graph that these entries form. If this graph is a closed path identical to the boundary of P, then the collection $\mathcal{C}$ of polygons defines a cell complex decomposition of the given polygon P, otherwise it does not.

Time complexity. Step 1 can be executed in $O(n \log n)$ time: for each vertex in turn, we check if it coincides with any of the previously processed vertices by means of a balanced binary search tree; if it does, then it is assigned the integer associated with the vertex found in the tree, otherwise it is inserted in the tree and is assigned the next available positive integer. In this way, the integers assigned to the vertices range from 1 to the number of distinct vertices, which does not exceed n. Steps 2 and 3 take $O(n)$ time. Step 4 as well can be completed in linear time by using radix sorting [2] (recall that vertex indices range from 1 to n), and so do Steps 5 and 6. Thus, the algorithm requires a total of $O(n \log n)$ time. The space required by the algorithm is $O(n)$.

Thus, we have the following theorem:

Theorem 3.2. *Let P and $\mathcal{C}$ be a polygon and a collection of polygons respectively of total size n. Then, it can be determined whether the collection $\mathcal{C}$ forms a cell complex decomposition of P in $O(n \log n)$ time using $O(n)$ space.*

It is important to observe that all the steps of the above algorithm but Step 1 take linear time. In fact, if the input consists of the list V of distinct vertices of the polygon P and of the polygons in the collection $\mathcal{C}$, followed by the vertices of each polygon, where each vertex is referenced by its index to the list V, then we do not need to execute Step 1. Such a representation is commonly used, and is easily produced by partitioning programs. Hence, we have:

Corollary 3.1. *Let P and $\mathcal{C}$ be a polygon and a collection of polygons respectively of total size n, where each polygon is described by a sequence of indices to a global list of distinct vertices which indicates the order of the vertices around the boundary of the polygon. Then, it can be determined whether the collection $\mathcal{C}$ forms a cell complex decomposition of P in $O(n)$ time using $O(n)$ space.*

4. The Three-dimensional Case

We assume that each polyhedron in the input is described by a list of its facets, each represented by the sequence of its vertices along its boundary, where each vertex is given by its coordinates.

Two lemmata similar to Lemmata 3.1 and 3.2 hold in the three-dimensional case and are the basis of our verification algorithms.

4.1. Verification of a partition of a polyhedron.

Lemma 4.1. *Let P and $\mathcal{C}$ be a polyhedron and a collection of polyhedra respectively. Additionally, let $F_I = \{f - \partial P \mid f$ is a facet of a polyhedron in $\mathcal{C}\}$ and $F_B = \{f \cap \partial P \mid f$ is a facet of a polyhedron in $\mathcal{C}\}$. Then, the collection $\mathcal{C}$ forms a partition of P if and only if*

(i) the set F_I of (parts of) facets of the polyhedra in $\mathcal{C}$ can be partitioned into two sets S and S' such that the closures of the polygons in each of these sets defines a partition of $\bigcup_{s \in S}$ closure(s), and for each (part of) facet f in S (S' resp.), the interior of the polyhedron of $\mathcal{C}$ with facet f and the interiors of the polyhedra of $\mathcal{C}$ with facets the polygons of S' (S resp.) which intersect with f lie (locally around f) on opposite sides of the plane supporting f;

(ii) the closures of the polygons in F_B form a partition of the boundary ∂P of P, and for each (part of) facet f in F_B, locally around f, the interior of the polyhedron of $\mathcal{C}$ with facet f and the interior of P lie on the same side with respect to the plane supporting f.

Proof: The proof is similar to that of Lemma 3.1. ∎

In light of Lemma 4.1, the partition verification algorithm is as follows:

3d Partition Verification Algorithm

1. We orient the facets of the polyhedron P and of the polyhedra in $\mathcal{C}$ in a compatible fashion (i.e., in counterclockwise order as seen from outside the polyhedron).

2. For each polyhedron Q in $\mathcal{C}$ do

```
For each polyhedron $Q$ in $\mathcal{C}$ do
    for each facet $f$ of $Q$ do
        we find the lexicographically smallest vertex of $f$; let it be $u$;
        if the boundary of $f$ forms a left turn at $u$
        then we add the entry $(f, +1)$ to $A$;        {$A$ is an initially empty array}
        else we add the entry $(f, -1)$ to $A$;
```

```
3. For each facet $f$ of $P$ do
       we find the lexicographically smallest vertex of $f$; let it be $u$;
       if the boundary of $f$ forms a left turn at $u$
       then we add the entry $(f, -1)$ to $A$;
       else we add the entry $(f, +1)$ to $A$;
```

4. We sort the entries $(f_i, \pm 1)$ of the array A by slope of the plane supporting f_i and, in case of ties, lexicographically on the second field of the entry.

5. For each slope separately, we verify that the related facet records with "+1" and those with "−1" partition the same polygonal regions. If this matching terminates successfully, then the given collection $\mathcal{C}$ of polyhedra defines a partition of the given polyhedron P, otherwise it does not.

Time complexity. Clearly, Steps 1, 2, and 3 can be completed in time linear in the total size n of the input. Step 4 requires $O(n \log n)$ time. Step 5 can also be completed in $O(n \log n)$ time: for each different plane slope, we collect the entries with "+1" as second field, and apply on the associated facets Steps 2 and 4 of the 2d partition verification algorithm (see Section 3.1); Step 5 of that algorithm is applied next, except that the difference of the union of the "+1" and the "−1" entries is computed and used to form a graph; the same procedure is applied to the entries of the array A with "−1" as second field; finally, the two resulting graphs are checked to see whether they are identical. Thus, the algorithm requires a total of $O(n \log n)$ time. The algorithm uses linear space. Thus, we have the following result:

Theorem 4.1. *Let P and $\mathcal{C}$ be a polyhedron and a collection of polyhedra respectively of total size n. Then, it can be determined whether the collection $\mathcal{C}$ forms a partition of P in $O(n \log n)$ time using $O(n)$ space.*

4.2. Verification of a cell complex decomposition of a polyhedron.

Lemma 4.2. *Let P and $\mathcal{C}$ be a polyhedron and a collection of polyhedra respectively. Then, the collection $\mathcal{C}$ forms a cell complex decomposition of P if and only if the set of facets of the polyhedra in $\mathcal{C}$ can be partitioned into two sets F_I and F_B such that*

(i) for each facet f in F_I there exists exactly one other facet, say f', in F_I such that $f = f'$ and, locally around f, the interiors of the polyhedra in $\mathcal{C}$ with edges f and f' lie on opposite sides of the plane supporting f;

(ii) the edges in F_B form a partition of the boundary ∂P of P and for each facet f in F_B, locally around f, the interior of the polyhedron of $\mathcal{C}$ with facet f and the interior of P lie on the same side with respect to the plane supporting f.

Proof: The proof is similar to that of Lemma 3.2. ∎

The cell complex verification algorithm applies Lemma 4.2. It uses two auxiliary arrays A and B, which are initially empty. The algorithm works as follows:

3d Cell Complex Verification Algorithm

```
1. We collect all the vertices and assign to each one of them a distinct positive integer
   $id(\ )$.

2. We orient the facets of the polyhedron $P$ and of the polyhedra in $\mathcal{C}$ in a compatible
   fashion.

3. For each polyhedron $Q$ in $\mathcal{C}$ do
       for each facet $f$ of $Q$ do
           $a \leftarrow$ the lexicographically smallest vertex of $f$;
           $b \leftarrow$ the lexicographically largest vertex of $f$;
           $c \leftarrow$ $f$'s vertex farthest away from the line $\overline{ab}$ (and lex-smallest, if ties);
           if $f$'s boundary is directed from $a$ to $c$ to $b$
           then we add the entry $((id(a), id(b), id(c)),\ +1)$ to $A$;
           else we add the entry $((id(a), id(b), id(c)),\ -1)$ to $A$;

4. We sort the array $A$ lexicographically.

5. We traverse the sorted array $A$, we verify that matched entries (which now appear
   next to each other) correspond to matching facets and we delete them.

6. For each facet $f$ whose entry in $A$ has not been matched do
       for each edge $\overrightarrow{uv}$ of $f$ do
           if $u$ is lexicographically smaller than $v$
           then we add the entry $(id(u), id(v), +1)$ to $B$;
           else we add the entry $(id(v), id(u), -1)$ to $B$;

7. We sort the array $B$ lexicographically.

8. If the entries in $B$ do not appear in matching pairs (i.e., $(k, k', -1)$ and $(k, k', +1)$),
   then the collection $\mathcal{C}$ of polyhedra does not define a cell complex decomposition of the
   given polyhedron $P$.

9. For each matching pair $(id(u), id(v), \pm 1)$ in $B$ do
       if the two facets incident upon this pair of edges are not coplanar
       then make $u$ and $v$ adjacent;

10. We apply a depth-first traversal and we check that the graph formed matches the edge
    skeleton of $P$. If it matches, then the collection $\mathcal{C}$ of polyhedra defines a cell complex
    decomposition of the given polyhedron $P$, otherwise it does not.
```

Time complexity. Step 1 takes $O(n \log n)$ time, as described in Section 3.2. Steps 2, 3, 5, and 6 take $O(n)$ time. Steps 4 and 7 can be completed in linear time as well by using radix sorting. Finally, Steps 8, 9, and 10 also take linear time. Thus, the algorithm requires a total of $O(n \log n)$ time. The space required by the algorithm is $O(n)$.

Thus, we have the following theorem:

Theorem 4.2. *Let P and $\mathcal{C}$ be a polyhedron and a collection of polyhedra respectively of total size n. Then, it can be determined whether the collection $\mathcal{C}$ forms a cell complex decomposition of P in $O(n \log n)$ time using $O(n)$ space.*

As in the two-dimensional case, all the steps of the above algorithm but Step 1 take linear time. Hence, we have:

Corollary 4.1. *Let P and C be a polyhedron and a collection of polyhedra respectively of total size n, where each polyhedron is represented by a list of facets, each described by a sequence of indices to a global list of distinct vertices which indicates the order of the vertices around the boundary of the facet. Then, it can be determined whether the collection C forms a cell complex decomposition of P in $O(n)$ time using $O(n)$ space.*

5. Concluding Remarks

In this paper, we present simple $O(n \log n)$-time and $O(n)$-space verification algorithms for partitions and cell complex decompositions of polygons and polyhedra. We note that if the input is so that vertices are referenced by means of indices to a global list of distinct vertices (something which can be easily obtained from a partitioning program), then the cell complex verification algorithms take linear time.

The obvious open question is whether linear time verification algorithms for these problems can be constructed. We conjecture that linear partition verification algorithms may be difficult to obtain under the problem's specifications because we assume that vertices are referenced by their coordinates. Another open question is whether these verification algorithms extend to higher than three dimensions.

6. References

[1] D. Avis and H. El Gindy, Triangulating point sets in space, *Discrete and Computational Geometry* **2**, 99–111, 1987.

[2] T.H. Cormen, C.E. Leiserson, R.L. Rivest, and C. Stein, *Introduction to Algorithms*, 2nd edition, MIT Press & McGraw-Hill, 2001.

[3] O. Devillers, G. Liotta, F.P. Preparata, and R. Tamassia, Checking the convexity of polytopes and the planarity of subdivisions, *Computational Geometry: Theory and Applications* **11(3-4)**, 187–208, 1998.

[4] H. Edelsbrunner, F. Preparata, and D. West, Tetrahedralizing point sets in 3 dimensions, *Journal of Symbolic Computation*, **10**, 335–347, 1990.

[5] K. Mehlhorn, S. Näher, T. Schilz, S. Schirra, M. Seel, R. Seidel, and C. Uhrig, Checking geometric programs or Verification of geometric structures, *Proc. 12th Symposium on Computational Geometry*, 159–165, 1996.