1990

Zone Theorem and Polyhedral Decompositions

Chandrajit L. Bajaj

Tamal K. Dey

Report Number:
90-1002
ZONE THEOREM
AND
POLYHEDRAL DECOMPOSITIONS

Chanderjit L. Bajaj*
and
Tamal K. Dey†

Computer Sciences Department
Purdue University
Technical Report CSD-TR-1002
CAPO Report CER-90-31
August, 1990

* Supported in part by NSF grant DMS 88-16286, NSF grant CCR 90-02228 and ONR contract N00014-88-K-0402.
† Supported in part by David-Ross Fellowship.
Zone Theorem and Polyhedral Decompositions

Chanderjit L. Bajaj*  Tamal K. Dey†

Department of Computer Science
Purdue University
West Lafayette, IN 47907

Abstract

We use the zone theorem for a better analysis of an algorithm for worst-case optimal convex decomposition of non-convex simple polyhedra. This new analysis reveals that the simple algorithm using repeated cutting and splitting through notches which guarantees a worst-case optimal convex decomposition is quite efficient. We establish an $O(nr + sr^2)$ time and an $O(nr)$ space bound for the algorithm where the input polyhedron has $n$ edges of which $r$ are reflex and $s = \min(n, r^2)$. For most cases $r << n$ in practice. Thus, the second term in the time complexity is dominated by the first one in most of the cases in practice. However, our goal, here, is not to present an asymptotically better algorithm for the problem but to establish the fact that a simple algorithm for the problem is actually quite efficient in practice.

1 Introduction

The main purpose behind decomposition operations is to simplify a problem for complex objects into a number of subproblems dealing with simple objects. Convex decompositions lead to efficient algorithms, for example, in geometric point location and intersection detection, motion planning, see [6]. Our motivation stems from the use of geometric models in SHILP, a solid model creation, editing and display system being developed at Purdue [1].

In simple polyhedra which are homeomorphic to a 2-sphere, non-convexities are caused by the reflex edges where the inner dihedral angle is greater than $180^\circ$. These edges are called notches. The non-convexity caused by a notch can be removed by cutting the polyhedron with a plane through the notch which resolves the inner reflex angle. In general, there are infinite choices for the plane which removes a notch. The chosen plane for eliminating a notch is called the notch plane for that notch. A notch plane may intersect other notches and thus may create subnotches. All subnotches of a notch can be eliminated by cutting and splitting the polyhedra containing those subnotches with a single notch plane. This simple method of convex decomposition was first suggested by Chazelle in [4]. He analyzed the algorithm and presented an $O(nr^3)$ time and $O(nr^2)$ space complexity for it. It was thought that due to repeated visiting of the edges of the polyhedral pieces, this algorithm is doomed to have a high complexity. Recently, in [3], we modified the algorithm given in [4] to have an $O(nr^2)$ time and $O(nr)$ space complexities. Though, better algorithms for the subproblems (e.g. polygon nesting [2]) helped to reduce the complexities, it was the use of the zone theorem for line arrangements on plane which led to the better analysis. Extending that approach, we use the zone theorem for hyperplane arrangements in 3-dimension to analyze our algorithm which gives $O(nr + sr^2)$ time and $O(nr)$ space bounds where $s = \min(n, r^2)$. Chazelle and Palios gave an algorithm to tetrahedralize simple polyhedra which runs in $O(nr + r^2 \log r)$ time and $O(n + r^3)$ space. Since in most cases, $r << n$, the first term in the

---

*Supported in part by ARO Contract DAAG29-85-C0018 under Cornell MSI, NSF grant DMS 88-16286 and ONR contract N00014-88-K-0402.
†Supported in part by David-Ross fellowship.
time complexities of both the algorithms dominate the second term. In that respect, both algorithms are comparable. Moreover, when it comes to simplicity, the algorithm described here, outperforms others with considerably low cost. The algorithm of Chazelle and Palios works for simple polyhedra which are homeomorphic to a 2-sphere. These restricted class of polyhedra cannot have holes and shells (internal voids). The input polyhedra of our algorithm have one more restriction that no facet can have holes. In what follows, we use the term simple polyhedra to refer to the input polyhedra of our algorithm.

2 Preliminaries

2.1 Arrangements of Hyperplanes

A finite set of \( r \) hyperplanes decomposes the \( d \)-dimensional space \( E^d \) into connected components of various dimensions. This decomposition is called an arrangement of \( n \) hyperplanes in \( d \)-dimension. For example, in 2-dimension, a finite set of lines decompose the plane into 0-dimensional points, 1-dimensional line segments and 2-dimensional facets. In particular, a \( d \)-dimensional connected component of a \( d \)-dimensional arrangement is called a cell. For \( 0 \leq k \leq d \), a \( k \)-flat is the affine hull of \( k+1 \) affinely independent points. Thus, a point is a 0-flat, a line is a 1-flat and a plane is a 2-flat. The connected component of a \( k \)-flat which is adjacent to a cell is called a \( k \)-face of the arrangement. The number of \( (d-1) \)-faces adjacent to a cell defines its combinatorial complexity. Since each cell is convex, by Euler’s relation, the number of \( k \)-faces for a particular \( k \) is within a constant factor of the number of \( k \)-faces for other values of \( k \). Careful counting of the faces which are adjacent to the cells, satisfying certain conditions, gives not so obvious bounds on their complexities. Zone theorem is such a wonderful result. Collection of all the cells adjacent to a hyperplane is called its zone.

Zone Theorem: In an arrangement of \( r \) hyperplanes in \( d \)-dimension, the total complexity of all the cells adjacent to a hyperplane is \( \Theta(n^{d-1}) \).

Corollary 1: In a 3-dimensional arrangement of \( r \) planes, the complexity of the cells, adjacent to a line which lies on a single plane of the arrangement, is \( O(r^2) \).

Proof: Follows directly from the zone theorem applied to 3-dimension.

2.2 Data Structure

Let \( S \) be a simple polyhedron, with no holes on the facets, and having \( s \) vertices : \( \{v_1, v_2, ..., v_s\} \), \( n \) edges : \( \{e_1, e_2, ..., e_n\} \) and \( q \) facets : \( \{f_1, f_2, ..., f_q\} \).

The polyhedron \( S \), is represented by a collection of vertices, edges, and facets, each of which is maintained as structures similar to the representations of [8].

Vertices: Each vertex is represented with two fields.

1. vertex.coordinates: contains the three dimensional coordinates of the vertex.
2. vertex.adjacencies: contains pointers to the edges incident on the vertex.

Edges: Each edge is represented with two fields.

1. edge.vertices: contains pointers to the incident vertices.
2. edge.orientededges: contains pointers to the structures called orientededges which represent different orientations of an edge on each facet incident on it. The orientation of an edge on a facet \( f_i \) is such that a traversal of the oriented edge has facet \( f_i \) to its right.

Orientededges: Each Orientededge is represented with four fields.

1. orientededge.edge: Contains pointer to the corresponding edge.
2. orientededge.facet: Contains pointer to the facet on which the orientededge is incident.
3. orientededge.orientation: Contains information about the orientation of the edge on the facet.
4. orientededge.nextorientededge: Contains pointer to the next orientededge on the oriented edge cycle of a facet as described below.

Facets: Each facet is represented with two fields.

2
Figure 1: notch plane, cross sectional map, cut

1. facet equation: contains the equation of the plane supporting the facet.

2. facet cycle: contains a pointer to the oriented edge cycle bounding the facet. The traversal of the oriented edge cycle always has the facet to the right. An oriented edge cycle is represented as a circular linked list of oriented edges on the cycle.

2.3 Useful Lemmas

Let $G$ be a simple polygon with vertices $v_1, v_2, ..., v_n$ in clockwise order, a vertex $v_i$ is a reflex vertex of $G$ if the inner angle between the edge $(v_{i-1}, v_i)$ and $(v_i, v_{i+1})$ is $> 180^\circ$. In the following Lemmas, the line segments of a line which are interior to a polygon or a polyhedron are called chords.

Lemma 2.1: Let $G$ be a simple polygon (possibly with holes) with $r$ reflex vertices. No line can intersect $G$ in more than $r + 1$ chords.

Proof: See [3]

Lemma 2.2: Let $S$ be a simple polyhedron with $r$ notches. A line intersects $S$ in $O(r)$ chords.

Proof: Consider a cross-section of $S$ on a plane passing through the line. Let there be $k$ polygons on the cross-section. Certainly, $k = O(r)$. The total number of chords present in these polygons is equal to the number of chords present in $S$. Using Lemma 2.2 for each polygon on the cross-section gives $O(\sum_{i=1}^{k} (r_i + 1)) = O(r)$ bounds for the number chords present in $S$.

3 Convex Decomposition

3.1 Sketch of the Algorithm

A reflex edge of a polyhedron is removed by cutting it with a notch plane. This notch plane may intersect other notches to create subnotches of that notch. At any generic step of the algorithm, all subnotches of a notch, present possibly in different polyhedra, are eliminated by a single notch plane. We assume that the notch plane does not pass through any vertex of the polyhedron being cut. This avoids many complications that would arise otherwise. For details see [3].

Algorithm ConvDecomp($S$)

Step 1: Assign a notch plane for each notch in $S$.

Step 2: repeat

Let $g_1, g_2, ..., g_k$ be the subnotches of a notch $g$ present in the polyhedra $S_1, S_2, ..., S_k$. Let $P_g$ be the notch plane assigned to $g$. Remove $g_1, g_2, ..., g_k$ from $S_1, S_2, ..., S_k$ by the notch plane $P_g$.

until all notches are eliminated.
Step 1 can be performed trivially in $O(r)$ time. The crucial step of the algorithm is Step 2 which we detail below.

### 3.2 Intersecting a Simple Polyhedron with a Notch Plane

Let $S$ be a simple polyhedron with $r$ notches and $p$ edges which is encountered at a generic step of the algorithm ConvDecomp. The notch plane $P_g$: $ax + by + cz + d = 0$ defines two half spaces $P_g^+: ax + by + cz + d \geq 0$ and $P_g^-: ax + by + cz + d \leq 0$. To cut a polyhedron $S$ with the plane $P_g$, it is essential to compute

$$GP^1_g = P_g \cap cl(int(P_g^+) \cap int(S))$$
$$GP^2_g = P_g \cap cl(int(P_g^-) \cap int(S))$$

where $cl(O)$ and $int(O)$ denote the closure and interior of the geometric object $O$. We refer to $GP^1_g$ and $GP^2_g$ as cross sectional maps. Note that $GP^1_g$ and $GP^2_g$ may be different. See for example, Figure 1. However, if $P_g$ does not pass through any vertex of $S$, $GP^1_g$ and $GP^2_g$ are congruent and we denote both of them as $GP_g$. In general, $GP_g$ consists of a set of isolated points, segments and polygons, possibly with holes. The unique polygons $Q_g$ on $GP_g$ containing the notch $g$ on its boundary is called the cut. Note that to remove a notch $g$, it is sufficient to split $S$ along only the cut instead of the entire cross sectional map.

The algorithm to cut a polyhedron $S$ with a notch plane $P_g$ consists of two basic steps.

- Computing the cut $Q_g$.
- Splitting the polyhedron $S$ along $Q_g$.

### 3.3 Computation of cut $Q_g$

To compute the boundary $B_g$ of $Q_g$, we compute the intersection points of a facet with the notch plane $P_g$. Let $a_1^g, a_2^g, \ldots, a_i^g$ denote the intersection points on the edges $e_1^g, e_2^g, \ldots, e_i^g$ respectively of a facet $f_i$. These intersection points are computed on the fly as we go along computing the boundary $B_g$ as follows. Let $a_1^g$ be an initial point on $B_g$. Later, we will see how this initial point is chosen. Let $a_1^g, a_2^g, \ldots, a_i^g$ be the intersection points sorted along the line of intersection $P_g \cap f_i$. This sorted sequence can be computed at a cost, linear in number of edges present in the facet $f_i$ using the algorithm of [7]. We keep this sorted sequence of intersection points associated with $f_i$. Further, with each intersection point $a_i^g$, a pointer to the edge $e_i^g$ is kept associated. We continue to construct the boundary $B_g$ containing $a_i^g$ as follows. See Figure 2. One of the segments $a_{i-1}^g a_i^g$ and $a_i^g a_{i+1}^g$ lies inside $f_i$. Without loss of generality, assume $a_{i-1}^g a_i^g$ lies inside $f_i$. We join $a_{i-1}^g$ and $a_{i+1}^g$ and continue from $a_{i+1}^g$ to determine other edges of the boundary $B_g$. Let $f_{i+1}$ be the other facet, adjacent to the edge $e_{i+1}^g$. The facet $f_{i+1}$ can be retrieved in constant time in our data structure. The intersection points on $f_{i+1}$ sorted along the line $P_g \cap f_{i+1}$ is computed and kept associated with $f_{i+1}$ for further use. We determine the intersection point $a_{m+1}^g$ adjacent to $a_{i+1}^g$ on $f_{i+1}$ such that the segment $a_{i+1}^g a_{m+1}^g$ lies inside $f_{i+1}$. Without loss of generality, assume $a_{m+1}^g$ is ordered after $a_{i+1}^g$ in the sorted sequence of intersection points associated with $f_{i+1}$. Note that the points $a_{m-1}^g$ and $a_{i+1}^g$ are same. We proceed from $a_{m+1}^g$ and continue the above procedure until the initial point $a_1^g$ is reached. At any generic step, we use the list of sorted intersection points on a facet if it has been computed earlier. Any vertex of $g$ can be taken as an initial point for this computation. By this method, computation of the boundary $B_g$ takes at most $O(t)$ time where $t$ is the number of edges on the facets intersected by $P_g$. This is mainly due to the fact that the sorted list of intersection points on a facet can be determined at a cost, linear in number of edges of the facet using the algorithm of [7].

### 3.3.1 Splitting $S$

Splitting $S$ along the cut $Q_g$ is carried out by splitting facets which are intersected by $Q_g$. Suppose $f_i$ is such a facet which is to be split at $a_1^g, a_2^g, \ldots, a_i^g$ which are on the edges $e_1^g, e_2^g, \ldots, e_i^g$. The splitting
of $f_i$ is carried out by splitting the edges on which $(a_1^i, a_2^i, ..., a_k^i)$ lies. To do this, we visit only the intersection points on each facet intersected by $P_g$ and for each such intersection point spend constant time for setting relevant pointers to carry out the split operation. We create two oppositely oriented facets at the same geometric location corresponding to the cut $Q_g$ and adjust all the modified incidences properly. This, in effect, splits $S$ into two pieces. Observe that the modifications, carried out in the data structures, are restricted only to the edges incident on the facets intersected by $P_g$. This cannot take more than $O(t)$ time.

3.4 Worst Case Complexity

Combining the costs of "cut computation" in section 3.2.1 and "splitting operation" in section 3.2.2, we get the following Lemma.

Lemma 3.1. A simple polyhedron $S$, having $p$ edges can be partitioned with a notch plane $P_g$ of a notch $g$ in $O(t)$ time and in $O(p)$ space where $t$ is the number of edges present on the facets of $S$ intersected by $P_g$.

Lemma 3.2. Let $S_1, S_2, ..., S_k$ be the polyhedra in the current decomposition, where each $S_i$ contains a subnotch $g_i$ of a notch $g$ of a simple polyhedron $S$ with $n$ edges and $r$ notches. Let $m_i$ and $u_i$ be the number of edges and vertices on $Q_{g_i}$ respectively. Then $m$ and $u$, the total number of edges and vertices on all the cuts supported by the subnotches of the notch $g$ are given as $m = \sum_{i=1}^{k} m_i = O(n)$ and $u = \sum_{i=1}^{k} u_i = O(n)$.

Proof: Consider the cut $Q_g$ produced by the intersection of $S$ with $P_g$. As an effect of cutting the polyhedral pieces with notch planes, the region in $Q_g$ is divided into smaller facets by notch lines produced by the intersection of other notch planes with $P_g$. We focus on the facets $Q_{g_1}, Q_{g_2}, ..., Q_{g_k}$ adjacent to the subnotches $g_1, g_2, ..., g_k$ of the notch $g$.

Consider the set of notch lines which divides $Q_g$ and the line $L_g$ corresponding to the notch $g$. They produce a line arrangement [6] on the notch plane $P_g$. Consider the facets adjacent to the line $L_g$ in this arrangement. They form the so-called zone of $L_g$. In Figure 3(a) we have shown the zone of $L_g$ only on one of its side. Let us denote this zone by $Z_g$ and the set of its vertices and edges by $V_g$ and $E_g$ respectively. By zone theorem, $|V_g| = O(l)$ and $|E_g| = O(l)$ if there are $l$ lines in the arrangement. Overlaying $Q_g$ on $Z_g$ produces $Q_{g_1}, Q_{g_2}, ..., Q_{g_k}$. Let $V'_g$ and $E'_g$ denote the set of vertices and edges in $Q_{g_1}, Q_{g_2}, ..., Q_{g_k}$. The vertices in $V'_g$ can be partitioned into three disjoint sets, namely, $T_1, T_2, T_3$. The set $T_1$ consists of vertices formed by the intersection of two notch lines, $T_2$ consists of vertices formed by the intersection of two edges of $Q_g$ and $T_3$ consists of vertices formed by the intersections of a notch line and an edge of $Q_g$. Certainly, $|T_1| \leq |V'_g| = O(l)$, since overlaying of $Q_g$ on $Z_g$ cannot introduce any vertices in $T_1$. If $Q_g$ has $u'$ edges, $|T_2| \leq u'$.
To count the number of vertices in $T_a$, consider an edge $e$ in $E_g$ which contributes one or more segments to $E'_g$ as a result of intersections with $Q_g$. There must be at least one reflex vertex of $Q_g$, present between two successive edge segments of $e$. Charge a cost of 1 to this reflex vertex which lies to the left (or, right) of each segment and charge a cost of 1 to $e$ itself for the leftmost (or, rightmost) segment. We claim that each reflex vertex of $Q_g$ is charged at most once by this method. Suppose, on the contrary, a reflex vertex is charged twice by this procedure. Then, that reflex vertex must appear between two segments of two edges in $E_g$ as shown in Figure 3(b). As can be easily observed, all four edge segments cannot be adjacent to the regions incident on the edge $g$ of $Q_g$. This contradicts our assumption that all of these four segments are present in $E_g$. Hence, the total charge incurred upon the reflex vertices of $Q_g$ and the edges of $E_g$ can be at most $r_g + O(1)$ where $r_g$ is the number of reflex vertices present in $Q_g$. This implies that as a result of intersections with $Q_g$, at most $r_g + O(1)$ segments of edges in $E_g$ are contributed to $E_g$. Hence, $I_{T_a} = O(r_g + 1)$.

**Lemma 3.3:** The total number of edges in the final decomposition of the polyhedron $S$ with $r$ notches and $n$ edges is $O(nr)$.  

**Proof:** Total number of edges in the final decomposition consists of newly generated edges by the cuts, and the edges of $S$ which are not intersected by any notch plane. Since the total number of edges present in all the cuts corresponding to a notch is $O(n)$, the total number of newly generated edges by each notch plane is $O(n)$. Thus $r$ notch planes generate $O(nr)$ new edges. Hence, the total number of edges in the final decomposition is $O(nr + n) = O(nr)$.

**Lemma 3.4:** Let $S_1, S_2, ..., S_k$ be the polyhedra in the current decomposition, where each $S_i$ contains a subnotch $g_i$ of a notch $g$ of a simple polyhedron $S$ with $n$ edges and $r$ notches. Let $t_i$ be the number of edges on the facets of $S_i$ which are intersected by $P_g$. Then, $t = \sum_{i=1}^{k} t_i = O(n + sr)$ where $s = \min(n, r^2)$.  

**Proof:** Consider the set $\mathcal{P}$ of notch planes which produce the pieces $S_1, S_2, ..., S_k$ before eliminating the notch $g$. These planes and the notch plane $P_g$ form a plane arrangement $A(\mathcal{P})$ in 3-dimensions. By proper choice of notch planes, it can be guaranteed that no plane other than $P_g$ passes through the notch $g$. Consider all cells adjacent to the line $L_g$ corresponding to the notch $g$. Let us call them as zones of $L_g$ or $Z_g$ in short. We are interested in counting the number of facets bounding these cells. By Corollary 1, the complexity of $Z_g$ in $A(\mathcal{P})$ is $O(r^2)$, since there are at most $r$ notch planes.

This implies that there are only $O(r^2)$ facets in $Z_g$. Superimpose $S$ on $A(\mathcal{P})$. Arrangement $A(\mathcal{P})$ produces a decomposition of $S$. Focus only on the regions which are adjacent to $L_g$. Polyhedra $S_1, S_2, ..., S_k$ define precisely these regions which are also portions of $Z_g$. The complexity of the facets of $S_1, S_2, ..., S_k$ which are intersected by $P_g$ can be determined by carefully analyzing the effect of the superimposition.
of $S$ over $Z_g$. The set $T$ of edges on those facets can be partitioned into three disjoint sets, namely, $T_1, T_2, T_3$. The set $T_1$ consists of the edges formed by the intersections of a facet of $Z_g$ and a facet of $S$. The set $T_2$ consists of the edges formed by the intersections of two facets of $S$. The set $T_3$ consists of the edges formed by the intersections of two facets of $Z_g$. The edges of $S$ along with their intersections with the facets of $Z_g$ give rise to the edges in $T_2$. Thus, $|T_2| = O(n) + O(|T_1|)$. Again, the edges of $Z_g$ along with their intersections with the facets of $S$ give rise to the edges in $T_3$. Thus, $|T_3| = O(r^2) + O(|T_1|)$ since $Z_g$ has $O(r^2)$ edges.

To count the number of edges in $T_1$, consider a facet $f_i$ of $S$ superimposed on $Z_g$. Let $L_i$ denote the line $f_i \cap P_g$. The edges formed by the intersection of $f_i$ and the facets of $Z_g$ subdivide the region inside $f_i$. See Figure 4(a). Among these edges, the ones which are adjacent to $L_i$ in this subdivision belong to $T_1$. The segments in $T_1$ charge each edge of a facet of $S$ at most twice since three or more such segments intersecting the same edge of $S$ cannot be adjacent to $L_i$ simultaneously.

Figure 4: Intersections of facets of $S$ and $Z_g$.

4(a). The intersection of a facet $f_i$ of $Z_g$ with $f_i$ consists of several segments, in general. Partition the set $T_1$ into three disjoint sets, namely, $T_{11}, T_{12}, T_{13}$ as follows. The set $T_{11}$ consists of segments which do not intersect $L_i$ but intersect an edge of $S$. The set $T_{12}$ consists of segments which intersect $L_i$. The set $T_{13}$ consists of segments which intersect neither $L_i$ nor any edge of $S$. For each segment in $T_{11}$, charge a cost of 1 to the edge of $S$ it intersects. For each segment in $T_{12}$ and $T_{13}$, charge a cost of 1 to the facet of $Z_g$ which produces that segment. We claim that, with this charging scheme, each facet of $Z_g$ is charged at most $O(r)$ times and each edge of a facet of $S$ is charged at most twice.

The segments in $T_{11}$ charge each edge of a facet of $S$ at most twice since three or more such segments intersecting the same edge of $S$ cannot be adjacent to $L_i$ as illustrated in Figure 4(b). The intersection point of $L_i$ and a segment in $T_{12}$ lie on the line of intersection of $P_g$ and $P_h$ where $P_h$ is the notch plane corresponding to the facet in $Z_g$ which contributes that segment. By Lemma 2.3, any line can intersect a simple polyhedron $S$ with $r$ notches at most $O(r)$ times. This implies a facet of $Z_g$ is charged at most $O(r)$ times by the segments in $T_{12}$. A facet of $Z_g$ can intersect the facets of $S$ without intersecting any of its edges at most $r$ times since for every four consecutive such intersections, there must exist a notch of $S$. See Figure 4(c). This implies a facet of $Z_g$ is charged only $O(r)$ times by the segments in $T_{13}$.

The total cost charged to the edges of $S$ is at most $4n$ since each edge of $S$ can appear at most on two facets of $S$. The total cost charged to the facets of $Z_g$ is at most $O(r^2)$ since there are $O(r^2)$ facets of $Z_g$ which are charged at most $O(r)$ times. Hence, the number of edges in $T_1$ is $O(n + r^2)$. This gives, $t = |T_1| = |T_{11}| + |T_{12}| + |T_{13}| = |T_{11}| + O(n + |T_1|) + O(r^2 + |T_1|) = O(n + r^2)$. Further, by Lemma 3.3, $t = O(nr)$ which implies $t = O(n + sr)$ where $s = \min(n, r^2)$. 


As discussed in [4], one can always produce a worst case optimal number \(O(r^2)\) convex polyhedra by carefully choosing the notch planes.

**Lemma 3.5:** A simple polyhedron \(S\) with \(r\) notches, can be decomposed into \(\frac{r^2}{2} + \frac{r}{2} + 1\) convex pieces if all subnotches of a notch are eliminated by a single notch plane. Further, this convex decomposition is worst-case optimal since there exists a class of polyhedra which cannot be decomposed into fewer than \(O(r^2)\) convex pieces.

**Proof:** See [4].

**Theorem 3.1:** A simple polyhedron \(S\), having \(r\) notches and \(n\) edges can be decomposed into \(O(r^2)\) convex polyhedra in \(O(nr + sr^2)\) time and \(O(nr)\) space where \(s = \min(n, r^2)\).

**Proof:** Decomposition of a polyhedron consists of a sequence of cuts through the notches of \(S\) as illustrated in the Algorithm ConvDecomp. Step 1 which assigns a notch plane for each notch in \(S\) takes \(O(r)\) preprocessing time. According to the Lemma 3.5, ConvDecomp produces worst case optimal \(O(r^2)\) convex pieces at the end since all subnotches of a notch are removed by a single notch plane.

At a generic instance of the algorithm, let \(S_1, S_2, ..., S_k\) be the \(k\) distinct (non-convex) polyhedra in the current decomposition, where each \(S_i\) contains the sub notch \(g_i\) of a notch \(g\) which is going to be removed. Let \(S_i\) have \(p_i\) edges and \(p = \sum_i p_i\). Let \(t_i\) be the number of edges present in the facets intersected by \(P_g\) in \(S_i\) and \(t = \sum_i t_i\).

Applying Lemma 3.1, each sub notch \(g_i\) in \(S_i\) can be removed in \(O(t_i)\) time and in \(O(p_i)\) space. Thus, removal of a notch \(g\) can be carried out in \(O(\sum_i t_i)\) time and in \(O(\sum_i p_i)\) space. By Lemma 3.4, \(O(\sum_i t_i) = O(n + sr)\). Hence, a notch \(g\) can be removed in \(O(n + sr)\) space. Thus, elimination of \(r\) notches takes \(O(nr + sr^2)\) time. By Lemma 3.3, \(O(\sum_i p_i) = O(nr)\) which gives the space complexity of ConvDecomp.

**4 Conclusion**

We believe that the upper bound for the quantity \(t\) in Lemma 3.4 is actually \(O(n)\) instead of \(O(n + r^2)\). This immediately improves the time complexity of the algorithm to \(O(nr)\) which is the best bound one can achieve for this algorithm. In non-simple polyhedra with holes, \(Q_g\) can have inner and outer boundaries as described in [3]. It is not known currently how to find an initial point on the outer boundary of \(Q_g\) when the boundary containing \(g\) is an inner boundary of \(Q_g\). Of course, as described in [3], we can visit all the edges of the relevant polyhedral pieces, but this increases the complexity of the algorithm. Currently, research is going on these two topics.

**References**


