Given a finite set of points in $\mathbb{R}^3$, polyhedronization deals with constructing a simple polyhedron such that the vertices of the polyhedron are precisely the given points. In this paper, we present randomized approximation algorithms for minimal volume polyhedronization (MINVP) and maximal volume polyhedronization (MAXVP) of three dimensional point sets in general position. Both, MINVP and MAXVP problems have been shown to be NP-hard and to the best of our knowledge, no practical algorithms exist to solve these problems. It has been shown that for any point set $S$ in $\mathbb{R}^3$, there always exists a tetrahedralizable polyhedronization of $S$. We exploit this fact to develop a greedy heuristic for MINVP and MAXVP constructions. Further, we present an empirical analysis on the quality of the approximation results of some well defined point sets. The algorithms have been validated by comparing the results with the optimal results generated by an exhaustive searching (brute force) method for MINVP and MAXVP for some well chosen point sets of smaller sizes. Finally, potential applications of minimum and maximum volume polyhedra in 4D printing and surface lofting respectively have been discussed.

1 Introduction

Given a finite set of $n$ points, $S \in \mathbb{R}^3$, polyhedronization of $S$ is a way of connecting $n$ points in $S$ to form a simple polyhedron that is topologically homeomorphic to a sphere. In 1994, Grunbaum [1] proved the existence of a polyhedronization for every 3-dimensional point set. However, Grunbaum’s method may yield Schonhardt polyhedra which can not be tetrahedralized [2, 3]. In an extension of this work, Agarwal et. al [4] showed that, for any point set $S \in \mathbb{R}^3$, there always exists a polyhedronization which can be decomposed into a union of tetrahedra. Further, Edelsbrunner et.al [5] showed that every convex polyhedron can be decomposed into $O(n^2)$ tetrahedra. Our work is mainly motivated by the existence of a tetrahedralizable polyhedra for 3-dimensional point sets. In this work, we address the problem of polyhedronization with an additional volume constraint (minimum volume and maximum volume).

1.1 Problem Definition

S.P Fekete et.al [6] put forward a volume constrained polyhedron construction problem in higher dimensions. The problem is referred to as FACE which is formally stated as the following:

Definition 1. FACE: Let $d \geq 2$ and $1 \leq k \leq d$. Let a finite set $S$ of points in $d$-dimensional Euclidean space is given. Among all simple polyhedra that are feasible for vertex set $S$, find one with the smallest volume of its $k$-dimensional faces.

In our work, we consider one of the three dimensional version of FACE. We focus on the problem FACE with parameters $d=3$ and $k=2$, i.e. the problem of computing smallest volume polyhedron (MINVP) of any three dimensional point set. MINVP Problem, in its general form is the following:

Definition 2. MINVP Problem: Given a finite set $S$ of $n$ points in $\mathbb{R}^3$, find the simple polyhedron with the smallest volume from all feasible simple polyhedra for $S$.

A closely related maximization problem is MAXVP Problem, which can be stated as follows:

Definition 3. MAXVP Problem: Given a finite set $S$ of $n$ points in $\mathbb{R}^3$, find the simple polyhedron with the largest volume from all feasible simple polyhedra for $S$.

1.2 Theoretical Challenges

Though there exist quite a few algorithms for constructing polyhedra from three dimensional point sets, most of them are theoretical in nature. We couldn’t find any implemented results in some of these work [4] [3]. To the best of our knowledge, no prior work exist on volume constrained polyhedronization and hence the problem represents an intriguing challenge.

In [6], FACE is shown to be NP-hard for any choice of $d$ and $k$ and hence MINVP and MAXVP problems are also NP-hard. For any given point set, there exist polyhedronizations [1], but finding how many of them exist is a computationally hard problem and hence a deterministic algorithm does not exist for the same. As a consequence, minimal volume polyhedronization (correspondingly maximum volume
polyhedrization) of any point set of appreciable size becomes an extremely hard one. The two-dimensional version of the problem (minimum area and maximum area polygons) is already shown to be NP-complete [6] and an infeasibility of a constant factor approximation algorithm is mentioned in [7]. Being a direct extended version of minimal area (maximal area) polygonization, it is most likely that minimum volume (maximum volume) polyhedrization could also end up in a similar fate.

An exhaustive searching method for MINVP/MAXVP construction appears to be impractical. It considers all possible combinations of $2n - 4$ triangular faces of the point set $S$. For each combination, it checks whether a simple closed polyhedron is feasible or not. If a simple closed polyhedron can be formed, volume information will be updated. Finally, the polyhedron with smallest/largest volume will be returned. So overall, this method takes a time complexity of $O\left(\binom{n}{2} \binom{2n-4}{4}\right)$ [8]. So a brute force method for MINVP/MAXVP construction seem to be clearly infeasible for point sets of larger sizes.

### 1.3 Possible Applications

Apart from the theoretical interests (please refer Section 1.2), both MINVP and MAXVP have potential to be employed in many futuristic applications. For instance, in 4D printing, minimum volume representations of 3D faces made of smart materials that can expand or contract themselves, saves huge amount of space and shipping costs. In space exploration, such a volume reduction could help space agencies to reduce overall costs for sending large printed parts to outer space using the limited space available in the rockets or other space carriers and let the parts assemble themselves into desired object and shape (refer to 4D Printing in Section 7). Another potential application for minimum volume representations arises in shortest path computation on a 4D polytope. In [9], polyhedral unfolding has been used in computing the shortest path between two points along the polyhedral surface. Similarly, an appropriate usage of MINVP of 3D facets of the unfolded 4D polytope can considerably reduce the search space of the shortest path algorithm and consequently improve the computational complexity by a large extent. MAXVP can be applied in lofting of surfaces from a series of planar and convex contours which in turn find applications in medical imaging, digitization of objects and geographical information systems [10]. We have conceptualized a technique for surface lofting with the help of maximum volume polyhedrization in Section 7.

### 1.4 Our Contribution

We explore randomized methods for MINVP and MAXVP constructions. Some interesting observations on the properties of MINVPs and MAXVPs have been made and analyzed. Our major contributions are as follows:

1. We present randomized techniques for optimal volume tetrahedralizable polyhedrizations of point sets in $\mathbb{R}^3$. The running time of our algorithm is significantly better than the exhaustive search algorithm for the optimal volume polyhedrization.
2. We define concepts such as trap regions and local rearrangements in a polyhedron and exploit these ideas to show that the presented algorithms are capable of generating polyhedrizations of any point set.
3. We take some well defined convex point sets and analyze the approximation quality of the results generated by both MINVP and MAXVP algorithms. Further, empirical results have been shown to demonstrate the practical potential of our algorithms.
4. We discuss a potential application of MINVP in 4D printing, which is considered as the future of manufacturing. Further, we conceptualize an idea surface lofting from closed contours using MAXVP.

Rest of this paper is organized as follows: Related work has been mentioned in Section 2. Some ground work required for the description of the randomized algorithm is given in Section 3. Section 4 presents the incremental construction of MINVP and MAXVP along with the complexity analysis. Section 5 illustrates some of the experimental results. Correctness of the algorithms is established in Section 6. Applications of MINVP and MAXVP are elaborated in Section 7. In Section 8, we reflect on various properties of MINVP and/or MAXVP and conclude the paper.
of a given set of 3-dimensional points. An approximation algorithm is presented for computing the polyhedron with the minimal surface area. Basically, it starts from the convex hull of the given point set and then shrinks towards the minimal surface area polyhedron by employing a simple local transformation called as flip transformation. However, a fixed approximation factor is not guaranteed by this algorithm which is mentioned as the main drawback of this method [11].

Another work in this direction has been done by Veltkamp [8]. He presents a method for constructing polygonal or polyhedral boundary passing through all the points in the given set based on a parameterized geometric graph called as γ-neighborhood graph [8]. However, to the best of our knowledge, apart from a preliminary work on volume constrained polyhedronization by the authors [12], there does not exist any work which specifically addresses optimal volume polyhedronizations.

3 Notations and Definitions
Let \( S = \{p_0, p_1, p_2, ..., p_{n-1}\} \) denotes a set of \( n \) points and each point \( p_i \) is represented by its \( x \), \( y \) and \( z \) coordinates. We assume that duplicate points are not present in the input set \( S \). Two sets are updated and maintained till the termination of the algorithm. A Point set \( P_{i-1} \) consists of all the random points selected till the \( i - 1^{th} \) iteration and a Face set \( F_{i-1} \) consists of all the faces that the current polyhedron has in an arbitrary order. Each element in \( F_{i-1} \) is a triangle, \( \triangle \{p_i, p_m, p_n\} \) where \( 0 \leq i \leq n-1 \), \( 0 \leq m \leq i-1 \), \( 0 \leq n \leq i-1 \) and \( i \neq m \neq n \). In the \( i^{th} \) iteration, \( \text{polyhedron size} \) is defined as the number of points on the current polyhedron \( (P_{i-1}) \). Recall that MINVP is always composed of triangular faces. So if \( \text{polyhedron size} = n \) then the size of face set is \( 2n - 4 \) [13].

We define different types of polyhedral faces for the convenience of presenting the algorithm. Let \( P_{i-1} \) be a polyhedron and \( p \) be a point to be inserted to \( P_{i-1} \). Further consider that \( T \) be the tetrahedron that \( p \) forms with one of the face of \( P_{i-1} \).

Definition 4. An anchoring face joins the newly formed tetrahedron, \( T \) with the polyhedron \( P_{i-1} \). In Figure 1, the triangle with red edges constitutes the anchoring face.

![Fig. 1. A polyhedron which illustrates different types of faces](image)

Definition 5. A reference face is any face of the tetrahedron \( T \) except the anchoring face. For eg. triangles with blue edges represent reference faces in Figure 1.

Definition 6. A \( p \)-face is any face of the polyhedron \( P_{i-1} \) except the anchoring face. For eg. \( p \)-faces are shown as black edged triangles in Figure 1.

Definition 7. If a reference face \( f \) of \( T \) intersects with any of the \( p \)-faces (except when the \( f \) shares a common edge with the neighboring \( p \)-face of the anchoring face) then \( f \) is labelled as intersecting and subsequently \( T \) is called as an intersecting tetrahedron.

If all the reference faces of a tetrahedron \( T \), are non-intersecting, then \( T \) is referred to as a non-intersecting tetrahedron. Intersections between triangular faces are determined using volume predicates [13]. Ray crossing method [13] is used to check whether the selected random point \( p \) is in the interior/exterior or on the boundary of the current polyhedron \( P_{i-1} \).

We denote the randomized approximation algorithm for MINVP construction as RAA_MINVP and the polyhedron returned by RAA_MINVP as RAND_MINVP. Similarly, RAA_MAXVP is used as an abbreviation for randomized approximation algorithm for MAXVP construction and the corresponding polyhedron as RAND_MAXVP in the rest of the paper.

4 The Algorithm
In this section, we describe the algorithms for minimal and maximal volume polyhedronizations. MINVP and MAXVP construction algorithms build the polyhedra by either attaching or detaching tetrahedra in a greedy fashion while keeping the required volume constraint as invariant. However, starting point of both the methods differ. While in MINVP construction, it is a random tetrahedron, MAXVP construction uses the convex hull of the point set. The proposed algorithms are fairly simple for implementation and maintenance. An analysis of the worst case time complexity of RAA_MINVP algorithm is also presented in Lemma 4.1.

4.1 Randomized Minimal Volume Polyhedronizations (RAA_MINVP)
Initialization: The algorithm starts by constructing an initial tetrahedron. The tetrahedron is formed by selecting four points uniformly at random from the point set \( S \). Tetrahedron is denoted by \( P_3=\{p_0, p_1, p_2, p_3\} \) (please note that the subscripts are used to denote the randomly selected points from the set \( S \). The first randomly selected point is denoted by \( p_0 \) and so on). \( P_3 \) is used as the initial polyhedron, with the face set \( F_3=\{f_0, f_1, f_2, f_3\} \).

Iteration: Once the initial step of tetrahedron formation is completed, the algorithm runs for \( n-4 \) iterations. In each of these iteration \( i \), a point \( p_i \) is selected uniformly at random from the set \( S \setminus P_{i-1} \) and determines whether it lies in the interior or in the exterior or on the boundary of the previous polyhedron \( P_{i-1} \). After determining relative position of \( p_i \) with the previous polyhedron \( P_{i-1} \), algorithm proceeds as follows.
1. If the point \( p_i \) lies in the interior \( P_{i-1} \), then search for the largest volume non-intersecting tetrahedron that the point \( p_i \) makes with the faces of \( P_{i-1} \). Let us denote the tetrahedron as \( T_i = (p_p, p_i, p_q, p_r) \) where \( p_p, p_q, p_r \in P_{i-1} \). Remove the volumetric space bounded by \( T_i \) from the polyhedron \( P_{i-1} \). Update point set \( P_{i-1} \) by adding the point \( p_i \) to form the new polyhedron \( P_i \). Update the face set \( F_{i-1} \) by adding three new reference faces, \( \triangle(p_p, p_q, p_i) \) and \( \triangle(p_r, p_p, p_i) \) of the tetrahedron and deleting the old anchoring face, \( \triangle(p_p, p_q, p_r) \) of the polyhedron to get the new face set \( F_i \).

2. If the point \( p_i \) lies outside the \( P_{i-1} \), then search for the smallest volume non-intersecting tetrahedron that the point \( p_i \) makes with the faces of \( P_{i-1} \). Let us denote the tetrahedron as \( T_i = (p_p, p_i, p_q, p_r) \) where \( p_p, p_q, p_r \in P_{i-1} \). Add the volumetric space bounded by \( T_i \) to the polyhedron \( P_{i-1} \). Update point set \( P_{i-1} \) by adding the point \( p_i \) to form the new polyhedron \( P_i \). Update the face set \( F_{i-1} \) by adding three new reference faces, \( \triangle(p_p, p_q, p_i) \) and \( \triangle(p_r, p_p, p_i) \) of the tetrahedron and deleting the old anchoring face, \( \triangle(p_p, p_q, p_r) \) of the polyhedron to get the new face set \( F_i \).

3. If the point \( p_i \) lies on any face of previous polyhedron \( P_{i-1} \), then divide the face into three new faces which will be added to the new polyhedron. Let the point \( p_i \) lies on the face, \( \triangle(p_p, p_q, p_r) \). Algorithm updates the point set as \( P_i = P_{i-1} \cup \{ p_i \} \). The face set will be updated as \( F_i = F_{i-1} \cup \{ (p_p, p_q, p_i), (p_p, p_r, p_i), (p_q, p_r, p_i), (p_p, p_q, p_r) \} \).

4. If the point \( p_i \) lies on any edge of previous polyhedron \( P_{i-1} \), then divide the adjacent faces of the edge into two new faces resulting in the creation of four new faces which will be added to the new polyhedron. Let the adjacent faces of the edge \( p_i \) on which the point lies are \( \triangle_1(p_p, p_q, p_r) \) and \( \triangle_2(p_p, p_q, p_r) \). Algorithm updates the point set as \( P_i = P_{i-1} \cup \{ p_i \} \). The face set will be updated as \( F_i = F_{i-1} \cup \{ (p_p, p_q, p_i), (p_p, p_r, p_i), (p_q, p_r, p_i), (p_p, p_q, p_r) \} \).

Finalization: Once the iterations are completed, the algorithm returns the final point set \( P_{n-1} \) and face set \( F_{n-1} \). Sizes of \( P_{n-1} \) and \( F_{n-1} \) will be \( n \) and \( 2n - 4 \) respectively. The boundary of the final \( \text{RAND-MINVP} \) is composed of triangular faces contained in \( P_{n-1} \). Various stages in \( \text{RAND-MINVP} \) algorithm are illustrated in Figure 2.

Lemma 4.1: \( \text{RAND-MINVP} \) of a set \( S \) of \( n \) points in \( \mathbb{R}^3 \) can be constructed in \( O(n^3) \) time using linear space.

Proof: There exists a loop for randomly picking the points \( p_i \) from the point set \( S \setminus P_{i-1} \) which runs for \( n - 4 \) times. In each iteration of this loop, another loop for finding the smallest volume tetrahedron runs for 0 through \( 2i - 4 \). \( \text{RAND-MINVP} \) is always composed of triangular faces which implies that Euler’s formula can be applied to \( \text{RAND-MINVP} \). For a \( n \) vertex \( \text{RAND-MINVP} \), there exists 2n-4 faces [13]. In the second loop, one more loop runs for \( 2i - 5 \) times for checking for intersections of tetrahedron with the current \( p \) faces except the anchoring face. So the total time complexity is \( \sum_{k=0}^{n-1}(2i-4) \cdot (2i-5) = \sum_{k=0}^{n-1}(4i^2 - 18i + 20) \) which is equal to \( O(n^3) + O(n^2) + O(n) = O(n^3) \). So the overall time complexity is \( O(n^3) \). \( \text{RAND-MINVP} \) utilizes only two sets (Point set and Face set) of \( \Theta(n) \) and \( \Theta(6n - 12) \) for its operation. Hence it takes only linear space for the construction of \( \text{RAND-MINVP} \).

Lemma 4.2: \( \text{RAND-MINVP} \) of a set \( S \) of \( n \) points in \( \mathbb{R}^3 \) is always tetrahedralizable.

Proof: This property follows immediately from the construction of \( \text{RAND-MINVP} \). \( \text{RAND-MINVP} \) starts with an initial tetrahedron which is tetrahedralizable. Assume that we have incrementally constructed a polyhedron \( P \) which is tetrahedralizable till \( i^{th} \) iteration. In \( i + 1^{th} \) iteration, if the selected point is outside the current polyhedron, it adds a tetrahedron to \( P \) which preserves the tetrahedralizable property of the polyhedron. If the selected point lies on an edge, two tetrahedra whose faces share the edge are further divided into two each, giving rise to four new tetrahedra (effectively two, as four new tetrahedra replace the two old tetrahedra). Similarly, if the selected point lies on a face, three new faces induce three new tetrahedra from the old tetrahedra. If the selected random point is inside the current polyhedron, \( \text{RAND-MINVP} \) deletes a tetrahedron from the inner volumetric space of the current polyhedron \( P \). So the tetrahedralization structure of the \( P \) may be redefined and this leads to the formation of a new polyhedron which is tetrahedralizable.

Intersection checking employed in cases 1 and 2 of the iteration of \( \text{RAND-MINVP} \) algorithm ensures that the resultant polyhedron is tetrahedralizable.
polyhedron $P_i$ in each iteration is simple and closed. Further, it is trivial to observe that the cases 3 & 4 do not violate the topology of $P_i$ and hence RAND_MINVP of any set of points is always homeomorphic to a sphere.

4.2 Randomized Maximal Volume Polyhedronizations (RAND_MAXVP)

The method is similar to the RAND_MINVP algorithm. Algorithm starts with the convex hull of the point set. Convex hull of the point set is constructed using the $O(n^2)$ incremental algorithm presented in [13]. Once the convex hull of the given point set is constructed, then in each iteration, it selects one of the interior points (remaining points lying inside the convex hull) uniformly at random. The selected point is checked for the spatial containment in the previous polyhedron (initially, it is the convex hull). If the point lies in the interior of the previous polyhedron, then the smallest volume non-intersecting tetrahedron that the selected point forms with the faces of the previous polyhedron is removed to generate the current polyhedron. Otherwise, i.e. if the selected point lies external to the previous polyhedron, then the largest volume non-intersecting tetrahedron that the selected point forms with the faces of the previous polyhedron is added to it to generate the current polyhedron. The steps 3 and 4 of the RAND_MINVP are repeated, if the point lies on any face or edge of the previous polyhedron.

Lemma 4.3. For any finite set of convex points, $S \subseteq \mathbb{R}^3$ and $|S| = n$, RAND_MAXVP always constructs an optimal maximum volume polyhedronization of $S$ in $O(n^3)$ time.

Proof. This is obvious as RAND_MAXVP algorithm starts by computing convex hull of the given point set. For convex point sets, the maximum volume polyhedronization is the convex hull itself and this is computed using incremental algorithm that runs in $O(n^2)$ [13].

It can be observed that, if all but one points are convex in a point set $S$, then RAND_MAXVP generates an optimal volume polyhedronization. In such cases, RAND_MAXVP constructs convex hull and then interior point is attached to the face with which it forms the smallest volume tetrahedron. The running time of RAND_MAXVP algorithm is $O(n^3)$. The analysis is similar to the running time analysis of RAND_MINVP algorithm including an extra cost of $O(n^2)$ incurred from the construction of convex hull.

5 Experimental Results

Both the algorithms were implemented in C++ on MS Visual Studio 2008. The results are generated as stl files and viewed using SolidView 2011 3D modeling software. Several experiments were conducted on different point sets to evaluate the quality of RAND_MINVP and RAND_MAXVP algorithms. We experimented on well defined point sets such as platonic, prismatic, pyramid point sets and few non-convex point sets from Princeton shape benchmark (PSB). Both RAND_MINVP and RAND_MAXVP were generated for each of them and the generated results were analyzed for the optimality.

A machine having Intel core i5 processor with 2.40 GHz and 2GB RAM was used for performing computations. 20 trials were used to generate RAND_MINVP (RAND_MAXVP) of each of the point sets. One trial consists of 100 executions which implies that, for a point set, the algorithm was run for 2000 times before making a conclusion on the upper bound (or lower bound) on the optimal volume. It should be noted that some of the resulting polyhedron might not look like a closed one, which is basically due to lack of good viewing direction for them. However, we have verified that the resulting one is a polyhedron in the real sense of it. We categorize point sets into convex and non-convex point sets. The characterization is based on the convex hull of corresponding point set, which is defined as follows:

1. Convex point set: A point set $S \subseteq \mathbb{R}^3$ is said to be convex if all the points in $S$ are co-located on the convex hull of $S$.
2. Non-convex point set: In a non-convex point set $S$, some of the points lie inside the convex hull of $S$.

5.1 Standard Convex Point Sets

We conducted a study on the behavior of RAND_MINVP on platonic solid point sets which include tetrahedron(4 points), octahedron(6 points), cube(8 points), icosahedron(12 points) and Dodecahedron(20 points). Since every convex polytope can be tetrahedralized [14], RAND_MINVP is always generated for convex point sets. From the experi-

![Fig. 3. RAND_MINVP generated for convex point sets. (a) Tetrahedral (b) Octahedral (c) Cube (d) Icosahedral and (e) Dodecahedral platonic point sets. (f) Triangular (g) Square and (h) Pentagonal prismatic point sets. (i) Square (j) Hexagonal and (k) Octagonal pyramid point sets.](image-url)
Table 1. Minimum volume table for various point sets. The edge lengths of platonic solids used in our experiment are as follows: Cube:8, Octahedron:√2, Icosahedron:2, Dodecahedron:√5−1.

All values given in the Table are in cubic units.

<table>
<thead>
<tr>
<th>Trial No.</th>
<th>Octahedron</th>
<th>Cube</th>
<th>Icosahedron</th>
<th>Dodecahedron</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>Volume</td>
<td>Convex hull Volume</td>
<td>Volume</td>
<td>Convex hull Volume</td>
</tr>
<tr>
<td>1</td>
<td>1.000000</td>
<td>426.6667</td>
<td>5.64793</td>
<td>5.333193</td>
</tr>
<tr>
<td>2</td>
<td>1.000000</td>
<td>426.6667</td>
<td>3.370583</td>
<td>3.750793</td>
</tr>
<tr>
<td>3</td>
<td>1.000000</td>
<td>512</td>
<td>6.569188</td>
<td>6.647893</td>
</tr>
<tr>
<td>4</td>
<td>1.000000</td>
<td>426.6667</td>
<td>3.23599</td>
<td>3.370583</td>
</tr>
<tr>
<td>5</td>
<td>1.000000</td>
<td>426.6667</td>
<td>9.804977</td>
<td>7.270291</td>
</tr>
<tr>
<td>6</td>
<td>1.000000</td>
<td>512</td>
<td>8.059731</td>
<td>4.921115</td>
</tr>
<tr>
<td>7</td>
<td>1.000000</td>
<td>426.6667</td>
<td>3.23599</td>
<td>7.270291</td>
</tr>
<tr>
<td>8</td>
<td>1.000000</td>
<td>512</td>
<td>1.745466</td>
<td>3.939772</td>
</tr>
<tr>
<td>9</td>
<td>1.000000</td>
<td>426.6667</td>
<td>9.550531</td>
<td>5.273342</td>
</tr>
<tr>
<td>10</td>
<td>1.000000</td>
<td>426.6667</td>
<td>8.471874</td>
<td>5.333197</td>
</tr>
<tr>
<td>11</td>
<td>1.000000</td>
<td>512</td>
<td>2.824082</td>
<td>5.018406</td>
</tr>
<tr>
<td>12</td>
<td>1.000000</td>
<td>512</td>
<td>1.745466</td>
<td>3.23599</td>
</tr>
<tr>
<td>13</td>
<td>1.000000</td>
<td>426.6667</td>
<td>2.824082</td>
<td>5.018406</td>
</tr>
<tr>
<td>14</td>
<td>1.000000</td>
<td>512</td>
<td>6.314587</td>
<td>2.88401</td>
</tr>
<tr>
<td>15</td>
<td>1.000000</td>
<td>426.6667</td>
<td>7.138478</td>
<td>4.606546</td>
</tr>
<tr>
<td>16</td>
<td>1.000000</td>
<td>512</td>
<td>3.902739</td>
<td>2.88401</td>
</tr>
<tr>
<td>17</td>
<td>1.000000</td>
<td>426.6667</td>
<td>7.902322</td>
<td>4.823846</td>
</tr>
<tr>
<td>18</td>
<td>1.000000</td>
<td>512</td>
<td>6.314587</td>
<td>2.88401</td>
</tr>
<tr>
<td>19</td>
<td>1.000000</td>
<td>426.6667</td>
<td>7.138478</td>
<td>4.606546</td>
</tr>
<tr>
<td>20</td>
<td>1.000000</td>
<td>512</td>
<td>3.902739</td>
<td>2.88401</td>
</tr>
</tbody>
</table>

5.2 Non-convex Point Sets

In 1992, Ruppert et.al. [15] showed that it is NP-complete to decide whether a polyhedron can be tetrahedralized or not. Further, it is not so straightforward to tetrahedralize non-convex polyhedra without adding Steiner points and hence MINVP and MAXVP construction for non-convex point sets is extremely harder. However, we experimented with non-convex points of models taken from PSB. Figure 4 illustrates the RAND_MINVPs and RAND_MAXVPs for non-convex point sets taken from Princeton shape benchmark. Figures 4(a), 4(d), 4(g) & 4(j) show the models, Figures 4(b), 4(e), 4(h) & 4(k) show the RAND_MINVPs and Figures 4(c), 4(f), 4(i) & 4(l) show corresponding RAND_MAXVPs.

5.3 Validation of the Algorithms

In order to validate the results generated by RAND_MINVP algorithm, we implemented the brute force algorithm (BFA) for minimum volume polyhedronization of point sets. The brute force method uses an exhaustive searching for the set of valid polyhedra and picks up the polyhedron with the least volume as explained in Section 1.1. The inherent limitation of this exhaustive searching method is the exponential time it incurs (Θ(2^n) [8]). For instance, while the point set of size 5 has 210 combinations
of triangular faces to be generated, point set of size 6 generates 125970 combinations of triangular faces thereby scaling the running time by a factor of 600. Hence we limit our validation experiment to point sets of smaller sizes (In fact we consider various point sets of size up to 7).

Table 2. Running times of RAA_MINVP and brute force algorithm for various point sets. Times are reported in seconds (s), minutes(m), hours(h) and days(d).

<table>
<thead>
<tr>
<th>Serial No.</th>
<th>Point set Size</th>
<th>RAA_MINVP</th>
<th>BFA</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>5</td>
<td>2.007 s</td>
<td>2.010 s</td>
</tr>
<tr>
<td>2</td>
<td>6</td>
<td>2.048 s</td>
<td>5.167 s</td>
</tr>
<tr>
<td>3</td>
<td>7</td>
<td>2.148 s</td>
<td>47 m 55.76 s</td>
</tr>
<tr>
<td>4</td>
<td>8</td>
<td>3.425 s</td>
<td>265 d 2 h 6 m 40.45 s</td>
</tr>
</tbody>
</table>

For all the point sets that we considered, RAA_MINVP generates an optimal minimal volume polyhedron. This is verified by the results generated by the brute force algorithm. However, RAA_MINVP may take several executions to generate the optimal results. In our experiment, each trial consists of hundred executions of RAA_MINVP and on an average, we obtained the optimal result in 2-3 trials. Table 2 reports the running times of BFA and RAA_MINVP for point sets of sizes 5 to 8. Time taken for the total number of trials is reported in the case of RAA_MINVP in Table 2. It is to be noted that, the running time of BFA for point set of size 8 is approximate and computed by multiplying the time taken for generating one combination in the previous cases (point sets of size 5, 6 & 7) by the number of possible combinations for point sets of size 8. From Table 2, one can observe the huge difference between the times taken by BFA (∼265 days) and RAA_MINVP (3.425 seconds) for point set of size 8.

Figure 6 shows the optimal minimal volume polyhedronizations of some point sets of sizes 5 to 7. In each of these cases (Figures 6(a)-(e)), the polyhedra generated by both RAA_MINVP and brute force method were identical in terms of shape and volume.

6 Correctness of Algorithms

In RAND_MINVP and RAND_MAXVP algorithms, if none of the faces is completely visible from the selected point, an attempt to add or remove a tetrahedron to current polyhedron will result in a non-simple polyhedron. In such situations, the algorithms have to be restarted in order to obtain a new polyhedronization. Thus it becomes crucial to address the following questions in order to establish the validity of the given approaches.

1. For any point set $S \subseteq \mathbb{R}^3$, does there exist an ordering of points in $S$, which when subjected to the rules of the algorithms generates a polyhedronization?
2. Does there exist an ordering of points upon which RAA_MINVP (RAA_MAXVP) generates optimal (minimum/maximum) volume polyhedronization?

In this section, we address these questions in detail. For convenience, we use the term valid sequence ($\pi_v$) to refer to the sequence which generates a polyhedron and optimal sequence ($\pi_o$) to refer to the sequence which generates the optimal polyhedron. For the ease of presenting the idea, hereafter we refer to RAA_MINVP algorithm though most of the statements are equally applicable to RAA_MAXVP.

6.1 Existence of $\pi_v$

We start by illustrating a failure case of RAA_MINVP algorithm which use the concept of trap regions in polyhe-
A trap region in a polyhedron is a region in $\mathbb{R}^3$ from where no polyhedral faces are completely visible. Formally, one can define the trap region of a polyhedron as the following:

**Definition 8.** Trap region $(T(P))$ of a polyhedron, $P$.

$$T(P) = \{ x \in \mathbb{R}^3 \mid x \in \mathbb{R}^3 \text{ and no faces of } P \text{ are completely visible from } x \}.$$ 

Trap regions exist for certain polygons and polyhedra. For example, a trap region in 2D is illustrated in Figure 7(a). Figure 7(b) illustrates a configuration of planes which forms a trap region in the interior of the arrangement. Front plane (P4), back plane (P5) and side plane (P3) are perpendicular to xy-plane and top and bottom planes are parallel to the xy-plane. Even though, planes do not intersect each other, adjacent (in a logical sense) planes are very close to each other. It should be noted that albeit three planes with a suitable alignment with xy-plane is sufficient to form a trap region, for simplicity and ease of visualization we use the configuration with 5 planes to illustrate the idea. The trap region inside such a configuration assumes an approximate shape of triangular prism (Figure 7(c)) which is quite evident from the arrangement in Figure 7(b). We assume that all five planes are part of five polyhedral faces.

In the context of RAA_MINVP algorithm, any point lying inside a trap region will not be able to form non-intersecting tetrahedra with any of the polyhedral faces and hence the algorithm will get stuck. The points, edges and faces from the polyhedron $P$, which block the visibility of a point lying inside the trap region, with the polyhedral faces of $P$ are referred to as **blocking points**, **blocking edges** and **blocking faces** respectively. A minimal set of **blocking points** is the set of of minimum number of points, upon removing which, at least one face of $P_{n-1}$ is completely visible from $q$.

**Observation 6.1.** For any finite set of points, $S \subseteq \mathbb{R}^3$ and $|S| = n$, the only hindrance for polyhedronization of $S$ using RAA_MINVP is the presence of trap regions containing points from $S \setminus P_{i-1}$ at some iteration $i$ of RAA_MINVP.

**Local rearrangement** A local rearrangement is a way of resolving trap regions containing input points. The process, first detaches the minimal set of blocking points from the polyhedron and then attaches the trapped point to the resultant polyhedron, followed by attaching each point of the minimal set of blocking points. An illustration of local rearrangement for a point $q$ lying in the trap region of a polyhedron is shown in Figure 8. Minimal blocking set, $B = \{b, c\}$ is also shown in Figure 8. To effectively convey the idea and to avoid any clutter in the image, we have used different views of the polyhedron (Figure 8(i)-(iii)). Figures 8(iv)-(vi) illustrate the sequence depicting the final polyhedronization of $S$. Figure 8(iv) represents the polyhedronization of the remaining points, $P_i$ (excluding $B$ and $q$). Then $q$ is attached to $P_i$ followed by $b$ and $c$, always keeping the volume constraint as invariant, to obtain the final polyhedronization as illustrated in Figure 8(v) and Figure 8(vi) respectively.

Since the primary goal of this work is polyhedronization of point sets, it is extremely important to establish that RAA_MINVP is capable of generating a polyhedronization for any three dimensional point set $S$. Lemma 6.2 establish the claim through a delicate construction of final polyhedron through local rearrangements when a point gets trapped in a trap region. Observation 6.1 indicates that the removal of trap regions containing points from $S \setminus P$ is inevitable in polyhedronizing a point set. We try to show that there exists at least one sequence which is free of trap regions in Lemma 6.2.

![Fig. 8. Illustration of local rearrangement in 3D with minimal set of blocking points, $B = \{c, d\}$ and the trapped point $q$ (i). Bird's eye view of the polyhedron with few faces from the top and bottom removed. (ii). Top view (iii). Bottom view. Figures 8(iv)-(vi) illustrate the rearrangement from the top view with the removed/added tetrahedra zoomed in.](image-url)
Lemma 6.2. For any finite set of points, $S \subseteq \mathbb{R}^3$ and $|S| = n$, there exists an ordering of points, $\pi_v$ which when used by RAA\_MINVP algorithm generates a polyhedronization of $S$.

Proof. The proof is by induction on $|S| = n$. We try to show at least one valid sequence, $\pi_v$ for each of the cases. For $n=4$, it is trivial to find that $\pi_v$ is any permutation of points in $S$ and for $n=5$, $\pi_v$ consists of points on the convex hull of $S$ followed by the interior point.

We hypothesise that claim is true for all sets of size $< n$. Consider the case where a point $q \in S$ lies in $T(P_{n-1})$ where $P_{n-1}$ is a polyhedron of $n-1$ points of $S$ and $|S| = n$. For instance, $q$ lies somewhere in the triangular prismatic trap region of Figure 7(b). Let $P_{n-1}$ be decomposed into 2 sub sets of points, $\{S_r, B\}$ where $B = \{b_1, b_2, \ldots b_m\}$, is the minimal set of blocking points and $S_r = P_{n-1} \setminus B$. Using local rearrangements, the construction proceeds by first polyhedronizing $S_r$ to get $P_r$. Hypothesize ensures that points in $S_r$ where $|S_r| < n$, has a valid ordering which leads to polyhedronization. Then, $q$ is attached to $P_r$ followed by attaching points in $B$ keeping the volume constraint as invariant. Hence, for such a configuration, $\pi_v$ consists of ordering which leads to the polyhedronization of $S_r$, followed by $q$, followed by $b_1, b_2, \ldots b_m$. During the construction if trap regions arise again due to any point from $B$, a similar local rearrangement can be done. Since, the local rearrangements are possible as illustrated in Figure 8, we conclude that there exists at least one valid sequence ($\pi_v$) for any point set.

Moreover, polyhedronizations generated for larger point sets (refer Figure 5) reassures Lemma 6.2.

Corollary 6.3. For any finite set of points, $S \subseteq \mathbb{R}^3$ and $|S| = n$, there exists an ordering of points, $\pi_v$ which when used by RAA\_MAXVP algorithm generates a polyhedronization of $S$.

A similar line of arguments used in Lemma 6.2 hold good for corollary 6.3.

6.2 Analysis on Optimal Solutions

Both RAA\_MINVP and RAA\_MAXVP generate polyhedra which are topologically equivalent to a sphere, otherwise it can not enclose a volume (This property is being referred to as homeomorphic property). The tetrahedralization of octahedral point sets results in four tetrahedra (Refer Figure 9(a)). It should be observed that the optimal solution in the case of octahedral point sets contains three tetrahedra (devoid of any one tetrahedra. Let it be ABFD). Removal of a second tetrahedron from the resultant tetrahedralization will lead to either disconnected vertex (in the case of ABDE or BDCF) or violates the homeomorphic property (BCDE).

Figure 9(b) shows the tetrahedralization of cube point sets with one tetrahedron removed (ABDH). Further removal of any of the remaining five tetrahedra will lead to either disconnected vertices or violation of homeomorphic property. For example, removing tetrahedra BDCH and BAHE will disconnect the vertices D and A respectively. Removal of any of the remaining three tetrahedra (HCGB, HGBF or FBEH) will disturb the polyhedron property. So the optimal minimal volume polyhedron of cube point set consist of five tetrahedra (we follow the tetrahedralization of convex polyhedra). A similar argument holds good for the optimal solution of the square prismatic point set as well.

Tetrahedralization of triangular prism contains three tetrahedra (please refer Figure 9(d)). Removal of any of the tetrahedra will result in either disconnected vertices or violation of homeomorphic property. Here, removal of ACDF or ABCE will lead to the disconnected vertices D and B respectively. The removal of tetrahedron ACFE will violate the homeomorphic property. So the minimal volume polyhedronization of triangular prismatic point set is the triangular prism.

Tetrahedralization of square pyramid point set consist of two tetrahedra. Removal of any of these tetrahedra will make some points disconnected. In Figure 9(c), removal of tetrahedron ABCE will make vertex B disconnected, whereas removal of ACDE will make D disconnected. So the optimal minimal volume polyhedronization of square pyramid point set is the square pyramid itself.

This argument is applicable to any pyramid point set. The number of tetrahedra in an $n$-pyramid point set tetrahedralization is $\Theta(n - 2)$. A result that can be directly extended from the number of triangulations of a convex polygon (base polygon of the pyramid is always convex). Each triangle in the triangulation of the base polygon ($n$-gon) gives rise to a tetrahedron in the tetrahedralization of $n$-pyramid. Removal of any of these tetrahedra will either disconnect a vertex or violate the homeomorphic (to sphere) property of the resultant polyhedron.

A case by case analysis on the optimal solutions of large sized and/or non-convex point sets is not feasible. Given the randomness in the construction and the complex nature of the problem, it’s extremely difficult to theoretically establish
that there exists a $\pi_o$ for any given point set. So we conducted experiments on well defined smaller point sets and analyzed the optimal solutions of some well-defined point sets (Figure 9). From the results obtained so far, we can conjecture that $\text{RAA}_{\text{MINVP}}/\text{RAA}_{\text{MAXVP}}$ has the potential to generate optimal solution for a general point set (Conjecture 6.4). However, confirming this statement would require comparing this with brute force algorithm for those sets, yet another infeasible task, as brute force approach takes longer time to run even for five points.

**Conjecture 6.4.** For any finite set of points, $S \subseteq \mathbb{R}^3$ and $|S|=n$, there exists an optimal ordering of points ($\pi_o$), upon which $\text{RAA}_{\text{MAXVP}}$ (correspondingly $\text{RAA}_{\text{MINVP}}$) generates an optimal maximum (correspondingly minimum) volume polyhedronization.

### 7 Potential Applications

In this section, we discuss potential applications of MINVP in 4D printing and MAXVP in surface lofting.

**4D Printing:** In 4D printing, 3D objects are printed using a combination of smart materials and standard printing material and then, the printed material is self-assembled into a pre-programmed object having different shape or size. The whole concept is possible with the advent of composite material that can use shape memory to transform an item printed in 3D by allowing it to then change into a new a different shape or size. 4D printing has potentially huge benefits, from airplane wings that change form in flight to furniture shape or size. The whole concept is possible with the advent of composite material that can use shape memory to transform an item printed in 3D by allowing it to then change into a new a different shape or size. 4D printing has potentially huge benefits, from airplane wings that change form in flight to furniture shape or size. The whole concept is possible with the advent of composite material that can use shape memory to transform an item printed in 3D by allowing it to then change into a new a different shape or size. 4D printing has potentially huge benefits, from airplane wings that change form in flight to furniture shape or size.

Apart from manufacturing 3D unfolded object in a relatively small space, keeping minimum volumes for each 3D sub part will further reduce the space by a large amount. For instance, consider a tesseract that consists of 8 cubes shown in Figure 10(a). Let the volume of each cube be $V$. The unfolded tesseract in Figure 10(b) occupies a volume of $8V$. On the contrary, if the minimum volume polyhedron of cube point set has been used, the unfolded tesseract (Figure 10(c)) occupies only a physical space of approximately $6.6V$ (A cube can be decomposed into 5 tetrahedra as shown in Section 6.2 and therefore, each decomposed tetrahedron occupies approximately $\frac{1}{5}V$, leading to an approximate volume of $\frac{4V}{5}$ for a MINVP). This is a significant space reduction while manufacturing complex and large objects having huge volumes. Further, such a volume reduction could help space agencies to reduce overall costs for sending large printed parts to outer space and let the parts assemble themselves into desired object and shape. It is to be noted that the fourth dimension in 4D printing is the ability of the 3D printed object to change. However, we purposefully considered a 4D object (tesseract) to point out the enormous potentials of minimum volume polyhedronization in 4D printing technology.

**Surface Lofting:** Surface lofting from a series of planar contours arises in many fields such as medical imaging, digitization of objects and geographical information systems. Data obtained by medical imaging equipments, range sensors or elevation contours are interpolated to visualize and reconstruct human organs, CAD models and terrain surfaces [10]. We conceptualize a simple surface lofting technique using maximum volume polyhedronization of points sampled from adjacent contours. The technique consists of two stages: a user-assisted pre-processing and surface lofting. In the pre-processing step, the user can suitably group contours belonging to different segments of the 3D model such that two attached segments have one or more common contours (Figure 11(b)). Each segment, $R_i$ is then provided with a direction to process the member contours. Using a random sampling algorithm, each contour is converted to the corresponding point set, denoted by $C_i$.

During the surface lofting step, each pair of adjacent point sets, $C_i$ and $C_{i+1}$ is interpolated first through maximum volume polyhedronization, $M_i$. Then, the union of all $M_i$ will be taken to get the required surface. The pseudo-code of the surface lofting stage is presented in Algorithm 1. Figure 11 shows that surface lofting using MAXVP has the ability to reconstruct shapes with genus such as teapot. If the input consists of only convex contours, the proposed technique guarantees a piece-wise linear surface reconstruction. This guarantee stems from the fact that adjacent convex contours generates only convex point sets and $\text{RAA}_{\text{MAXVP}}$ assures to generate optimal MAXVP for convex point sets (due to Lemma 4.3).

---

Fig. 10. Illustration of how minvp can reduce the physical space in 4D printing. Assuming a volume of $V$ to each cube, unfolded tesseract in Figure 10(b) occupies a volume of $8V$ whereas the one in Figure 10(c) occupies a volume of $6.6V$. 

(a) Tesseract  
(b) Unfolded1  
(c) Unfolded2
Algorithm 1: Surface Loft(I)

Input: A set I of n convex, planar closed contours grouped into m segments, each with processing direction
Output: Surface SURF lofted from I.
1 Randomly sample each contour to get the associated point set, denoted by C_i;
2 for each segment R_i do
3 for each adjacent pair (C_i,C_{i+1}) taken in the processing direction do
4 $M_i = RAA\_MAXVP(C_i \cup C_{i+1})$;
5 end
6 end
7 $SURF = \bigcup_{i=1}^{m-1} M_i$;
8 Return SURF;

8 Concluding Remarks
We have presented randomized approximation algorithms for constructing tetrahedralizable minimal (correspondingly maximal) volume polyhedron enclosing a given point set in 3D. The problem of MINVP and MAXVP constructions are NP-hard problems and hence no exact algorithms of sub-exponential running times are known for these problems. To the best of our knowledge, this is the first practically implementable algorithm for MINVP and MAXVP of a given set of points in 3-space. Our experimental results show that both RAA.MINVP and RAA.MAXVP can be used for polyhedronization of three dimensional point sets of appreciable sizes. We list three basic properties of RAND.MINVPs (and RAND.MAXVPs) without formal proof here.

1. **Non-uniqueness of RAND.MINVP**- For a given point set $S$ in $\mathbb{R}^3$ the RAND.MINVP may not be necessarily unique. This observation is made on the basis of our experiments on cube (and octahedron) point sets. However, generated RAND.MINVPs of cube point set were found to be automorphic to each other.

2. **Boundary is composed of triangular Faces**- Conceptually, RAA.MAXVP proceeds by adding or deleting tetrahedra. At programming level, this is essentially implemented by adding three new triangular faces at the expense of an existing triangular face of the current polyhedron and hence the final polyhedron generated consists of only triangular faces.

3. **Maximum vertex degree**- Given a set $S$ of $n$ points, $n - 1$ of them are coplanar and a point $p$ in space. MINVP can be constructed as follows. Let $Q$ be the triangulated minimal area polygon of $n - 1$ points. Connect vertex of $Q$ to the point $p$ in space which makes the vertex degree of $p$ to be $n - 1$.

It is perhaps worth noting that, for a related problem on computing the exact arbitrarily-oriented minimum-volume bounding box of a set of $n$ points in $\mathbb{R}^3$, initially O’Rourke presented an algorithm [17] that runs in $O(n^3)$ time (subsequently, improvement in complexity has been attained [18]). Also, for general point sets, as constant factor approximation has proven to be difficult even for a 2D version (minimal area polygon) [7], it is likely that the problems MINVP/MAXVP could end up in similar fate. Two potential applications (4D printing and surface lofting) of volume constrained polyhedrizations have been discussed. It remains to be proved or disproved the conjecture on optimal sequences. Further work is being carried out on improving the time complexity of the algorithms.

References


