International Conference on Computational Sciences and Its Applications ICCSA 2008

Automatically Approximating 3D Points with Co-Axisal Objects

Russell Tempero[*] Sergey Bereg[†] Xiangxu Meng [‡] Changhe Tu[‡] Chenglei Yang[‡]
Binhai Zhu [†]

Abstract

In this paper, we investigate the problem of approximating a set $\mathcal{S}$ of 3D points with co-axisal objects typically from CAD/CAM (namely, cylindrical segments, cones and conical frustums). The objective is to minimize the sum of volumes of these objects (as well as the number of objects used). The general problem when the objects can have arbitrary axes is strongly NP-hard as a cylindrical segment, a cone and a conical frustum can all degenerate into a line segment. We present a general algorithm which combines a neat doubling search method to decompose $\mathcal{S}$ into desired subsets (or components). For each subset S, we present a unified practical approximation algorithm for minimizing the volume of the cone (conical frustum, or cylindrical segment) which encloses points in S. Preliminary empirical results indicate that the algorithm is in fact very accurate.

Keywords: Geometric modeling, Approximation algorithms, Smallest enclosing cone, Smallest enclosing conical frustum, Smallest enclosing cylindrical segment

1 Introduction

Approximating/fitting points with simple objects is a traditional problem in geometric modeling and computational geometry. Typically this is due to that simpler objects can facilitate the late computation process, e.g., the famous bounding-box method [Zh97, ZS99, BS01] and its application in collision detection (e.g., in [ASC+06]). Since a few years ago, several groups of researchers investigated the problem of computing the smallest enclosing cylinder (or cylindrical segment) for a set of points in three dimensions (3D) [SSTY00, Ch02, Be04, LZJO04, Zh04]. The applications of some of these research are from the manipulation of molecular configuration [Be04] and the approximation of a neuron with cylindrical segments in constructing neural maps to study their functionality (so as to further study human/animal behavior) [JT96, JT00, PDJ99, LZJO04].

In many applications related to CAD/CAM and solid modeling, enclosing a set of points with a cylinder or a cylindrical segment does not really solve the problem. (An example is when we model a tree or a neuron with cylindrical segments, the bottom of the trunk in a tree is usually thicker than other parts. A conical frustum is obviously a more natural fit.) So naturally one would think about using a cone or conical frustum, which will be one of the problems studied in this paper. However, in many applications even that is not enough. In Figure 1, we show an example of the Oil Pressure Sending Unit produced by the OMIX company. In this case, if many sample points are obtained from the surface (or interior) of the unit and one is to compute some kind of simple object enclosing these points, then apparently a cylinder or a cylindrical segment (or even a cone or conical frustum) does not make much sense. For this specific problem, we probably need a set of these objects with a common axis. To the best of our knowledge, this problem of fitting points with a set of co-axisal cones, conical frustums and cylindrical segments has never been formally studied before. In fact, even for fitting points with a single cone or conical frustum there has been no formal result.

In this paper, we first study the problem of enclosing a set of 3D points using either a cone or a conical frustum (or a cylindrical segment) of the minimum volume. We present a unified approximation algorithm which can not only solve the above base problems, but can also be generalized to the cases of enclosing a set of 3D points with a pair of cones or conical frustums or at most two cones or conical frustums together with a cylindrical segment (i.e., as long as the approximate objects form a convex set, see Figure 4). We then show how to generalize the algorithms further so that

[*]Department of Computer Science, Montana State University, Bozeman, MT 59717-3880, USA. Email: russellt,bhz@cs.montana.edu.
[†]Department of Computer Science, University of Texas at Dallas, Richardson, TX 75083, USA. Email: besp@utdalllas.edu.
[‡]School of Computer Science and Technology, Shandong University, Jinan, Shandong 250100, China. Email: mxx,chtu,chl_yang@sdu.edu.cn.

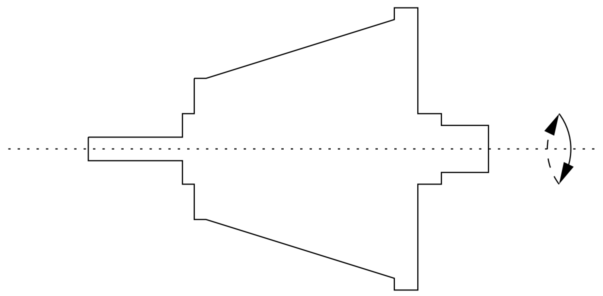

Figure 1. 2D Illustration of the OMIX® Oil Pressure Sending Unit.

we can enclose a set of 3D points with an arbitrary number of cylindrical segments, cones and conical frustums as long as they are all co-axisal and connected (i.e., base by base, see Figure 1). Again, the objective for these generalizations is to minimize the sum of volumes of the objects.

Be reminded that we are in fact dealing with a special case of an NP-complete problem. Theoretically, given n points in 3D if one wants to compute the minimum number of cylindrical segments so as to minimize the sum of radii of these segments (or the sum of volume, or the volume of the union of the segments), then the problem is strongly NP-hard [Zh04]. The proof there uses line segments to represent the degenerated cylindrical segments, so it also works for degenerated cones or conical frustums.

We comment that this research is related to reverse engineering, but in our case we work on the point cloud directly instead of a surface triangulation. Moreover, we present theoretically-sound approximations for our problems. This is different from some alternative solutions in reverse engineering [VMC97, WK05, AFS06].

The paper is organized as follows. In Section 2, we make necessary definitions. In Section 3 we discuss the smallest enclosing cone (conical frustum) and related problem of covering points with enclosing objects whose union is convex. In Section 4, we discuss how to generalize the algorithm, by using a doubling search technique, to the more general problem of covering points with many co-axial cones, conical frustums and cylindrical segments. In Section 5, we conclude the paper.

2 Preliminaries

In this section we make some necessary definitions regarding geometry and approximation algorithms which are related to the problem to be studied. Throughout this paper, the distance metric is Euclidean unless otherwise specified. Most of the definitions and the related materials can be found in textbooks on algorithms and computational geometry, e.g., in [CLRS01, PS85].

An approximation algorithm for a minimization optimization problem Π provides a **performance guarantee** of λ if for every instance I of Π, the solution value returned by the approximation algorithm is at most λ of the optimal value for I. (*Notice that following the above definition, λ is at least 1.*) For the simplicity of description, we simply say that this is a factor λ approximation algorithm for Π. Typical we are only interested in polynomial time approximation algorithms.

Given a set S of n points in 3D, the *diameter* of S, $D(S)$, is the maximum distance $d(p_i, p_j)$, $p_i, p_j \in S$, over all points in S. The *width* of S, $W(S)$, is the minimum distance between two parallel planes which contain all the points in S in between. A cylinder $\mathcal{C}$ is an infinite set of points which have at most a distance R to a given line l in 3D. The line l is called the *center* of $\mathcal{C}$. The section area of $\mathcal{C}$ which is vertical to l corresponds to a disk with radius R.

Given a line segment s_1s_2 in 3D, let the distance from a point q to the line through s_1s_2 be $d(q,r)$. The distance from q to s_1s_2 is $d(q,r)$ if r is on the line segment s_1s_2, otherwise the distance from q to s_1s_2 is infinite. A cylindrical segment G is an infinite set of points which have at most distance R to a given line segment s_1s_2 in 3D. Similarly, the line segment s_1s_2 is called the *center* of G and R is called the *radius* of segment. The two section areas through s_1, s_2 are called the *bases* of G. The length of s_1s_2, $d(s_1,s_2)$, is called the *length* of G and $2R$ is called the *width* of G. We denote them as *length*(G) and *width*(G) respectively.

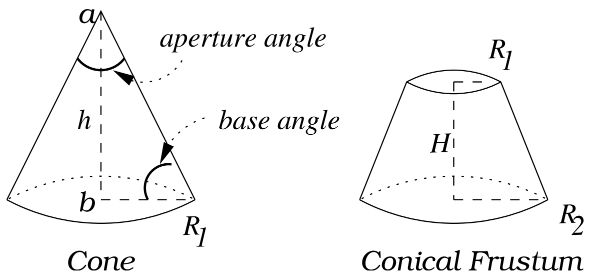

Figure 2. A Cone and a Conical Frustum.

A *cone* C is a pyramid with a circular cross section which is symmetric along the line through its *apex* a and the center of its base, b. The length between a and b, $h = d(a,b)$, is called the height of the cone. If the radius of C's base is R_1 then the volume of C is $V_C = \frac{\pi}{3}hR_1^2$. The intersection of C and a plane through the line segment ab

is a triangle, whose internal angle at the vertex a is called the *aperture angle* and whose internal angle at the vertex b is called the *base angle* of the cone C. A *conical frustum* F (or *truncated cone*) is a frustum obtained by slicing the top of a cone (with a cut parallel to its base). The length between the centers of F's two bases, H, is the *height* of F. If the radii of F's bases are R_1, R_2 respectively, then the volume of F is $V_F = \frac{\pi}{3}H(R_1^2 + R_1R_2 + R_2^2)$. In Figure 2, we show two examples of a cone and a conical frustum respectively. As in theory a conical frustum is a special case of a 3D cone (i.e., by setting exactly one of R_1, R_2 as zero), we add a constraint that R_1 is at least a constant fraction of R_2 (say 1/5), or vice versa. We assume that the conical frustums we discuss henceforth satisfy this assumption.

3 The Smallest Enclosing Cone and the Smallest Enclosing Conical Frustum Problems

In this section, we investigate the problem of computing the smallest enclosing cone (conical frustum) of a set of points in 3D such that the volume of the cone (conical frustum) is minimized. This problem has found applications in CAD and CAM, and to the best of our knowledge, there has never been a formal study for these problems before.

We will first present an $(1 + \varepsilon)$-approximation for the smallest enclosing cone problem. Then we will show how to generalize (modify) it to handle the smallest enclosing conical frustum and the smallest enclosing cylindrical segment problems. We will also sketch some further extension to co-axisal convex objects.

3.1 The smallest enclosing cone problem

In this subsection we present a simple yet efficient approximation algorithm for the smallest enclosing cone problem. Given a set S of n points, let C^* be the smallest enclosing cone of S (i.e., S is completely contained in C^* and the volume of C^* is minimized). It is easy to see that C^* can be determined by 7 parameters: the coordinates of its apex and the center of its base, as well as the radius of its base. Therefore, one can determine C^* using 7 points in S. So, using a brute-force method we can compute C^* in $O(n^8)$ time (the additional n factor comes from checking whether the computed cone contains all the points in S).

In practice, even if we can improve the running time for the smallest enclosing cone problem (say to $O(n^6)$), it will be hardly useful practically as n could be easily greater than 5000. So instead of trying to reduce the running times for obtaining the optimal or approximate solution, we will resort to a practical close approximation for C^*. In this case, by 'practical' we really mean easily implementable yet fast enough for practical datasets. We first have the following simple approximation algorithm:

Algorithm Cone-1(S).

(1) Compute the width W of S. Let $\delta \leq \frac{1}{35} \approx 0.02857$ be a given small constant. Let $\varepsilon = 13\delta \leq \frac{13}{35} \approx 0.371$.

(2) Compute a minimum axis-parallel bounding box $\mathcal{B}$ of S. Decompose the faces of $\mathcal{B}$ into grids with grid edge length being at most $\sqrt{2}\delta W$.

(3) Pick all possible pairs of grid points $p', p"$ such that they are on different faces of $\mathcal{B}$. Perform a coordinate transformation such that the line $p'p"$ is the X-axis, every point in S has a non-negative X-coordinate in the new coordinate system.

(4) For each point $q = (x, y, z)$ of S in the new coordinate system, perform a transformation $q \rightarrow q' = (x, 0, \sqrt{y^2 + z^2})$. Let the set of these transformed points be Q'. Compute the 2D convex hull of Q' and let each hull edge e on $CH(Q')$ define a candidate apex for a cone $C(p'p", e)$ enclosing S (this apex is the intersection of the extension of e with $p'p"$, or the X-axis).

(5) Order the upper hull of $CH(Q')$ from the highest vertex and we have two lists of edges, L_1 and L_2. Consider adjacent edges $e_i = (v_{i-1}, v_i), e_{i+1} = (v_i, v_{i+1})$ in L_1, and the corresponding cones $C(p'p", e_i)$ and $C(p'p", e_{i+1})$. Let $x = x_0$ be the X-coordinate of the leftmost point of $CH(Q')$. Discretize the interval I_i on the X-axis, determined by the two apexes of cones $C(p'p", e_i)$ and $C(p'p", e_{i+1})$ with discrete points A_j's such that $d(A_j, A_{j+1}) \leq \frac{\delta W}{2}$. Discretize the interval J_i on $x = x_0$ in the XZ-plane, determined by the two apexes of cones $C(p'p", e_i)$ and $C(p'p", e_{i+1})$ with discrete points B_j's such that $d(B_j, B_{j+1}) \leq \frac{\delta W}{2}$. For each discrete point A_j on I_i, compute the enclosing cone $C_{i,j}^{1,1}$ with apex A_j and with aperture angle $2\angle v_i A_j o$ and compute the volume of $C_{i,j}^{1,1}$ (Figure 3). For each discrete point B_j in J_i, compute the enclosing cone $C_{i,j}^{1,2}$ with an apex on the X-axis in the XZ-plane and with base angle $\angle v_i B_j X_0$, where $X_0 = (x_0, 0)$ is on the XZ-plane.

(6) Compute symmetrically all the cones relative to L_2, $C_{i,j}^{2,1}, C_{i,j}^{2,2}$.

(7) Among all possible $C_{i,j}^{1,1}, C_{i,j}^{1,2}, C_{i,j}^{2,1}, C_{i,j}^{2,2}$ return the one with the minimum volume.

Regarding Algorithm Cone-1, we have the following theorem.

Theorem 3.1 *Algorithm Cone-1 presents a factor-$(1 + \varepsilon)$ approximation for the smallest enclosing cone problem. The running time of Algorithm Cone-1 is $O(\frac{n \log n}{\varepsilon^4} + \frac{n}{\varepsilon^5})$.*

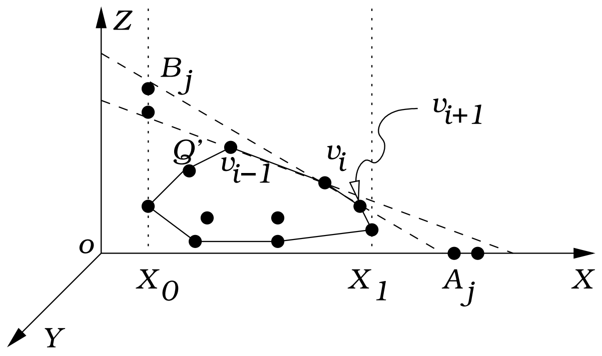

Figure 3. Illustration for Algorithm Cone-1.

Proof. We first show the approximation factor part. Let $V^*(S)$ be the optimal volume of the cone C^* enclosing S. Let $V_1^*(S)$ be the optimal volume of a cone enclosing S whose center goes through two grid points on $\mathcal{B}$. Let $V(S)$ be the optimal volume of a cone enclosing S returned by Algorithm Cone-1(S). We first show that

$$V(S) \leq (1 + 5\delta) \cdot V_1^*(S).$$

Let r and h be the base radius and height of the cone whose volume is $V_1^*(S)$. By construction, $V(S) \leq \frac{\pi}{3}(r + \frac{\delta W}{2})^2(h + \frac{\delta W}{2})$. As we have $W \leq 2r, W \leq h$, we have $V(S) \leq \frac{\pi}{3}(r + r\delta)^2(h + 2h\delta) \leq (1 + 5\delta)\frac{\pi}{3}r^2h = (1 + 5\delta)V_1^*(S)$, when $\delta \leq \frac{\sqrt{33}-5}{2} \approx 0.18614$.

Second we show that

$$V_1^*(S) \leq (1 + 7\delta) \cdot V^*(S).$$

Let c^*, r^* and h^* be the center, the base radius and height of the cone whose volume is $V^*(S)$. The line through c^* must pass through two square grids and by construction the smallest discrete cone has volume $V_1^*(S) \leq \frac{\pi}{3}(r^* + \delta W)^2(h^* + \delta W + \delta W)$. Again, we have $W \leq 2r^*, W \leq h^*$. Therefore, we have $V_1^*(S) \leq (1 + 2\delta)^3 \cdot \frac{\pi}{3}(r^*)^2h^* \leq (1 + 7\delta)V^*(S)$, when $\delta \leq \frac{\sqrt{11}-3}{4} \approx 0.07915$.

Putting these together, we have

$$V(S) \leq (1 + 13\delta) \cdot V^*(S) = (1 + \varepsilon)V^*(S),$$

provided that $\delta \leq \frac{1}{35} \approx 0.02857$ and $\varepsilon = 13\delta \leq 0.371$.

Now we summarize the running time of Algorithm Cone-1(S). Apparently we have $O(\frac{1}{\delta^2} \times \frac{1}{\delta^2}) = O(\frac{1}{\varepsilon^4})$ possible pairs of p', p". After one of them is fixed (we still call them p', p"), the transformation to obtain set Q' takes $O(n)$ time, computing $CH(Q')$ takes $O(n \log n)$ time and there are $O(n)$ possible hull edge e's and each $C(p'p", e)$ can be computed in $O(1)$ time (as the leftmost and rightmost points of Q' can be computed in $O(1)$ time, once $CH(Q')$ is known). Therefore, when p', p" are fixed, at the end of step (4) the overall running time of the algorithm is $O(n \log n)$.

At step (5) and (6), notice that we have $O(n)$ extreme vertices v_i's and for each i we have $O(\frac{1}{\delta})$ discrete points A_j's and B_j's. So when p', p" are fixed, the running time at step (5) and (6) is $O(\frac{n}{\delta}) = O(\frac{n}{\varepsilon})$. Therefore, the running time of Algorithm Cone-1 is $O(\frac{n \log n}{\varepsilon^4} + \frac{n}{\varepsilon^5})$. □

In some practical applications, δ must be very small (say, $\delta \leq 0.001$). Under those situations, the running time of the above approximation algorithm might be too high. However, we notice that after the axis $p'p$" is fixed, Algorithm Cone-1 really takes $O(n \log n + \frac{n}{\delta})$ time. So we can use some of the known heuristic methods to quickly find an approximate optimal axis, e.g., the one in [LZJO04]. This can obviously handle the high running time concern for those applications.

3.2 The smallest enclosing conical frustum problem

For the smallest enclosing conical frustum problem, we only need to make minor modifications to Algorithm Cone-1(S). First, let x_1 be the maximum X-coordinate of a point in Q'. Let $X_1 = (x_1, 0)$ be a point on the XZ-plane (Figure 2). We need to generate discrete points on the lines $x = x_0$ and $x = x_1$ (instead of on $x = x_0$ and the X-axis). At the end of steps (5) and (6) of Algorithm Cone-1(S) we should make some changes such that conical frustums $F_{i,j}^{1,1}$, $F_{i,j}^{1,2}$, $F_{i,j}^{2,1}$, $F_{i,j}^{2,2}$ (instead of the corresponding cones) are computed. The remaining details are similar to those in the proof of Theorem 3.1 hence omitted. We thus have the following corollary.

Corollary 3.1 *Algorithm Cone-1 can be modified to compute a factor-$(1 + \varepsilon)$ approximation for the smallest enclosing conical frustum problem in $O(\frac{n \log n}{\varepsilon^4} + \frac{n}{\varepsilon^5})$ time.*

3.3 Extension to the smallest enclosing cylindrical segment problem

Algorithm Cone-1(S) can be extended to compute a factor-$(1 + \varepsilon)$ approximation for the smallest (volume) enclosing cylindrical segment problem. Most of the research on the smallest enclosing cylinder (cylindrical segment) aims at minimizing the radius, which the exception of [Be04], in which the objective function can be volume as well.

This modification can be done by simply finding the point in Q' with the maximum Y-coordinate at the end of step (4). So the running time would be reduced to $O(\frac{n \log n}{\varepsilon^4})$. Of course, this is not a new result; in fact, it is a bit slower than the running time of the algorithm in [Be04], which takes $O(\frac{n}{\varepsilon^4})$ time to compute a factor-$(1+\varepsilon)$ approximation. But then we have a unified method for approximating S with either an approximate smallest enclosing cone, conical frustum or cylindrical segment. This is important in our implementation as it saves a lot of coding effort. So we have the following corollary.

Corollary 3.2 *When used as a subroutine for the general problem in Section 4, Algorithm Cone-1 can be modified to compute a factor-$(1+\varepsilon)$ approximation for the smallest enclosing cone, conical frustum or cylindrical segment of S.*

3.4 Further extension

Algorithm Cone-1(S) can be further extended to compute a factor-$(1+\varepsilon)$ approximation to the minimum volume convex co-axisal object (i.e., the object is either a pair of cones and/or conical frustums sharing a common base, or at most two cones or conical frustums together with a cylindrical segment in the middle (Figure 4). The idea is to choose a pair of points on the chains L_1, L_2 (which are on $CH(Q')$) and the point on $CH(Q')$ with the highest Y-coordinate, then use these points to determine topologically a convex co-axisal object. Then similar to Algorithm Cone-1, we can discretize the search intervals and compute the one with the smallest volume (among all discrete convex co-axisal objects considered). This optimal discrete solution is a factor-$(1+\varepsilon)$ approximation to the minimum volume convex co-axisal object. As we did not implement this part of extension, we just sketch this idea and skip further technical details.

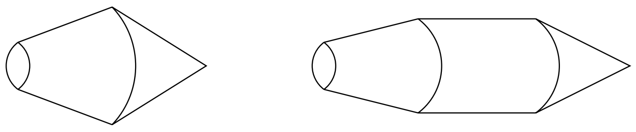

Figure 4. Convex Co-axisal Objects.

4 Enclosing 3D points with Co-Axisal Objects

In this section, we discuss how to enclose a set $\mathcal{S}$ of 3D points with a set of (connected) co-axisal objects (i.e., cylindrical segments, cones or conical frustums). "Connected" means that neighboring objects must share bases. We show that the solutions we have obtained in the previous section can be used as efficient subroutines for solving the problem. Without loss of generality, we assume that the points in $\mathcal{S}$ are sorted along their X-coordinates (i.e, along $p'p$" in Algorithm Cone-1) and are stored in an array $A[1..n]$.

Let $O^*[1..n]$ be the optimal solution; i.e., there are n^* connected co-axisal objects in $O^*[1..n]$ such that they enclose $A[1..n]$ and their sum of volume is minimized. When the context is clear, we also use $O^*[1..n]$ to denote the corresponding minimum volume. We first prove the following lemma.

Lemma 4.1 *If $O^*[1..n]$ is the optimal solution, then there is no partition of $A[1..n]$ into $A[1..n_p']$ and $A[n_p'+1..n]$ such that the resulting solution $O^*[1..n_p'] + O^*[n_p'+1..n] < O^*[1..n]$; moreover, if $O[1..n]$ is not the optimal solution, then there exists a partition which will result in a solution better than $O[1..n]$.*

Proof. The first part of the lemma is easy to prove: If there is such a partition then we can obtain a solution better than $O^*[1..n]$, which contradicts the optimality assumption of $O^*[1..n]$.

The second part of the lemma can be proven as follows. If there exists no partition which will induce a solution better than $O[1..n]$, then by definition $O[1..n]$ is the optimal solution. Again, this contradicts with the non-optimality assumption of $O[1..n]$. □

The above lemma does not immediately imply an algorithm. However, we can use it on finding the first optimal object in $O^*[1..n]$ in a greedy fashion. We have the following local lemma.

Lemma 4.2 *(***Local Lemma***) If k_1^* is the first partition in the optimal solution $O^*[1..n]$, then there is no partition of $A[1..k_1^*]$ into $A[1..n_1']$ and $A[n_1'+1..k_1^*]$ such that the resulting solution $O^*[1..n_1'] + O^*[n_1'+1..k_1^*] < O^*[1..k_1^*]$; moreover, if $O[1..l_1]$ is not in the optimal solution, where l_1 is the corresponding first partition, then there exists a partition $l_1' < l_1$ which induces a better solution $O[1..l_1'] + O[l_1'+1..l_1] < O[1..l_1]$.*

The proof of this local lemma is similar to that for Lemma 4.1 hence omitted. Based on this local lemma, we can obtain a decision procedure Algorithm $Local(A[i..j],\sigma)$. Let $Cone1(A[i..j])$ be the approximate minimum volume of either a cone, conical frustum or a cylindrical segment returned by Algorithm Cone-1 (or one of its generalizations). Algorithm $Local(A[i..j],\sigma)$ decides whether there is a local partition on $A[i..j]$ whose resulting solution has a sum of volume which differs from $Cone-1(A[i..j])$ by an amount of σ, where σ is some given constant.

Algorithm $Local(A[i..j],\sigma)$

(1) Search with $p = j-1, j-2, j-3, ..., i+1$ for the first p such that $|(O[i..p-1]+O[p,j]) - Cone1(A[i,j])| >$

$\sigma \cdot Cone1(A[i,j]))$. If p is found then return false; otherwise, return true.

Let $j - i = m$. The running time of Algorithm $Local(A[i..j], \sigma)$ is $f(m) = O(mT(m))$, where $T(m)$ is the running time for Algorithm Cone-1 on an input of m points. So $f(m) = O(\frac{m^2 \log m}{\varepsilon^4} + \frac{m^2}{\varepsilon^5})$.

We can use binary search (using the Local(-,-) algorithm as a decision procedure) to compute the first partition $A[n_1]$ (hence the optimal co-axisal object enclosing $A[1..n_1]$). This would take $O(f(n) \log n)$ time. So the whole time complexity for computing $O^*[1..n]$ would be $O(n^* f(n) \log n)$.

However, we show below that with doubling search [KK89, Ch96] this running time can be improved greatly, especially when n^* is large. We present the following Algorithm $MinVol(A[1..n], \sigma)$ which solves the problem in $O(f(n) \log \frac{n}{n^*})$ time. We use a greedy method to find the first *breakpoint* (the largest index j) such that $A[1..j]$ can be covered by one optimal object (within error σ), but $A[1..j+1]$ cannot be covered by any one object with the minimum volume (again within error σ).

Algorithm $MinVol(A[1..n], \sigma)$

(1) Search with $t = 1, 2, 3, ...$ the first t such that $Local(A[1..2^{t-1}], \sigma)$ returns true while $Local(A[1..2^t], \sigma)$ returns false. Then find the first breakpoint k_1 in $A[2^{t-1}..2^t]$ using binary search.

(2) Repeat the above process on $A[k_1 + 1..n]$ to compute all of the $n^* - 1$ breakpoints.

Clearly k_1 can be found in $O(f(k_1) \log k_1)$ time. This is due to that when k_1 is in $A[2^{t-1}..2^t]$, then 2^t is at most $2k_1$. Let $n_1, n_2, ..., n_{n^*}$ be the sizes of the subarrays determined by the $n^* - 1$ breakpoints (note that $n_1 = k_1$). The overall running time of MinVol is

$$\sum_i f(n_i) \log n_i,$$

which is $O(f(n) \log \frac{n}{n^*})$, where $\sum_{1 \le i \le n^*} n_i = n$. As Algorithm Cone-1 is used as a subroutine, the eventual approximation factor is increased by at most σ. By setting $\alpha = \varepsilon + \sigma$ with $\varepsilon = \Theta(\sigma)$, we then have the following theorem.

Theorem 4.1 *There is a factor-$(1 + \alpha)$ approximation for the problem of enclosing a set of n 3D points with n^* connected co-axisal objects such that their sum of volume is minimized. The running time of this algorithm is $O((\frac{n^2 \log n}{\alpha^4} + \frac{n^2}{\alpha^5}) \log \frac{n}{n^*})$, where n^* is the number of objects in the optimal solution.*

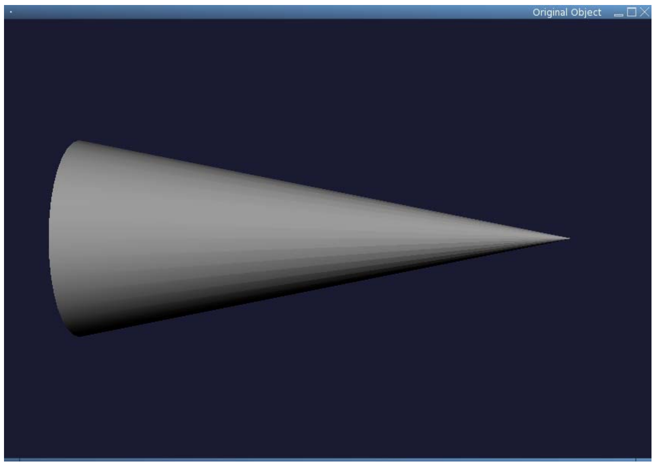

Figure 5. An Input Cone.

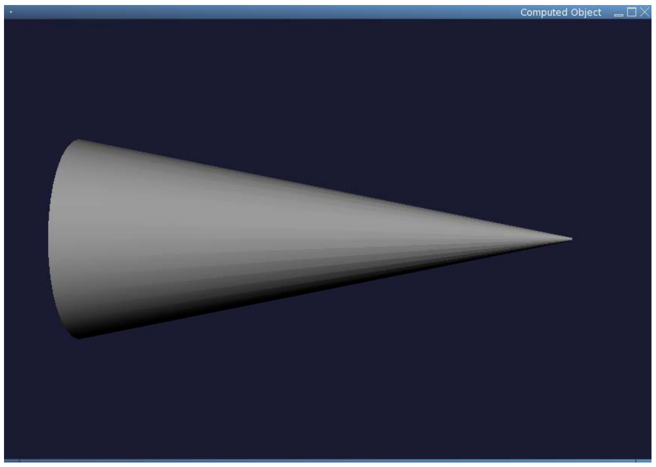

Figure 6. The Reconstructed Cone from Algorithm Cone-1.

5 Empirical Results

We show some empirical results in this section. Our preliminary implementation is in Java3D API. In Figure 5, we show an input cone with volume 41.8879. We generate 10,000 points on the surface of it and feed these points to our program (which is based on Algorithm Cone-1). We set $\delta = 0.0285$. The output cone is shown in Figure 6 with a volume of 43.8552. The difference of volume is only about 4.7%, which is significantly better than the theoretical upper bound of $13 \times \delta = 37.05\%$.

In the implementation of Algorithm $Local(A[i,j], \sigma)$ for Algorithm MinVol, we try to practically speed up the search for p by making simultaneously a number of κ partitions according to the density of the points generated on the sur-

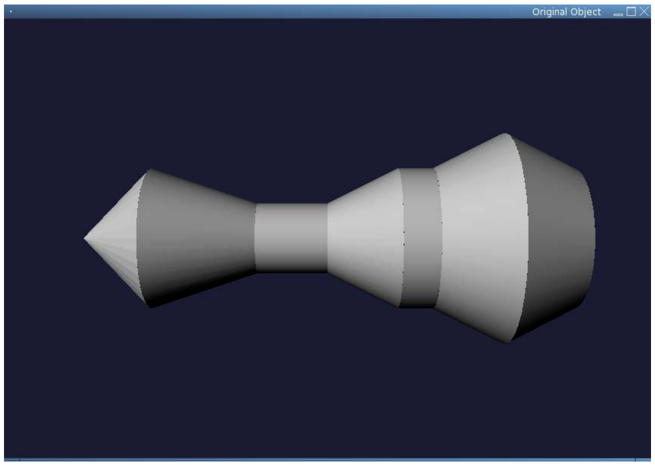

Figure 7. A Complex Input Co-axisal Object.

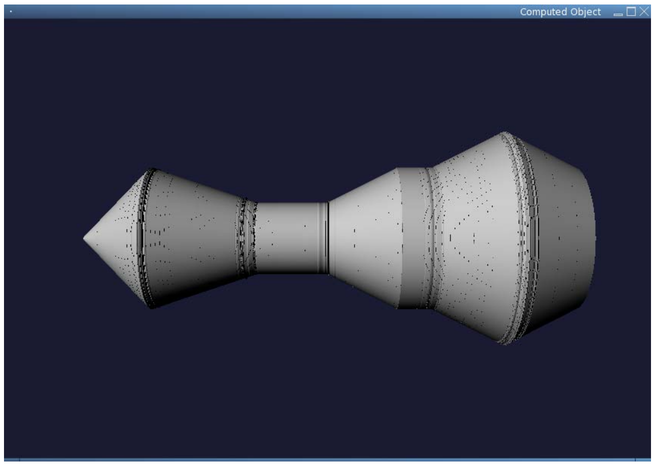

Figure 8. The Reconstructed Object from Algorithm MinVol.

face of the object. Then we use the summation of the $\kappa + 1$ co-axisal objects' volume to compare with that of $Cone1(A[i,j])$. Intuitively, this is faster than searching for p sequentially (i.e., trying all possible cuts) and is supported by some of our empirical results. (In the worst case this will not be able to reduce the running time of the algorithm, as one can easily construct a worst case scenario. See also the results in Table 1 and Table 2 below.)

In Figure 7, we show an input co-axisal object with volume 143.4660 and with a total of seven connected pieces. We generate 14,000 points on the surface of it and feed these points to our program (which is based on Algorithm MinVol). We set $\delta = 0.0285, \sigma = 0.005$ and $\kappa = 5$. The reconstructed object cone is shown in Figure 8 with a volume of 147.6515. The difference of volume is only 2.92%. Readers might notice the 'points' on the surface of the reconstructed object, which we do not erase on purpose. These 'points' are shadows of the cuts we used in the doubling search steps.

In summary, our empirical results give small errors and conform with the theoretical results.

In Figure 9, we show another co-axisal object composed of nine connected pieces (including a long skinny one). We test on this object subject to different κ, σ and the density of sample points (i.e., number of sample points used per unit surface area). The objective is to explore the relationship of these parameters with the percentage of the approximation error on volume, i.e., α. Throughtout this test, we set $\delta = 0.0285$ (with no intention of using a smaller δ to minimize/reduce the overall approximation error, which is already supported by the theoretical results). Some testing results are shown in Table 1 and Table 2. The conclusions from the two tables are that the final approximation error has almost nothing to do with the density, at least when enough sample points are generated on the surface. Small σ, on the other hand, contributes smaller errors. κ seems to be the most difficult parameter to control, and there does not seem to be a correlation between α and κ. So with some application in which a very small error bound is desirable it might be a better idea to implement the original version of Algorithm $Local(A[i,j], \sigma)$, besides selecting a small α.

	$\kappa = 3$	$\kappa = 6$	$\kappa = 9$	$\kappa = 12$
$\sigma = 0.008$	13.77	12.94	13.06	13.25
$\sigma = 0.006$	13.53	12.81	13.03	13.25
$\sigma = 0.004$	13.46	12.58	13.00	13.25
$\sigma = 0.002$	13.20	12.46	12.62	13.24

Table 1. The percentage of approximation error α, with $\delta = 0.0285$ and density = 100.

	$\kappa = 3$	$\kappa = 6$	$\kappa = 9$	$\kappa = 12$
$\sigma = 0.008$	14.07	13.52	13.66	12.74
$\sigma = 0.006$	14.00	13.50	13.65	12.74
$\sigma = 0.004$	14.00	13.49	13.64	12.73
$\sigma = 0.002$	13.74	13.47	13.60	12.67

Table 2. The percentage of approximation error α, with $\delta = 0.0285$ and density = 500.

6 Concluding Remarks

In this paper, motivated by applications in CAD/CAM, we describe a general method to approximate/fit a set of 3D points with a set of co-axisal objects (cones, conical frustums or cylindrical segments). Besides theoretical proofs,

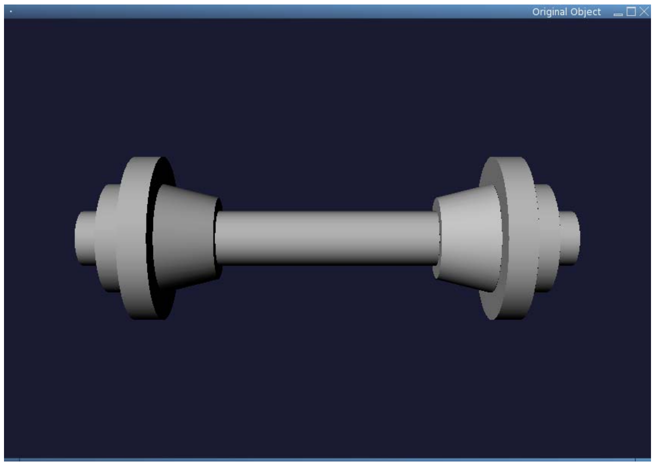

Figure 9. The Second Complex Co-axisal Object Used for Testing.

our preliminary empirical results are very promising. We comment that while approximating 3D points with arbitrary branches of co-axisal objects is strongly NP-hard [Zh04], our method might be extended to situations where the centers of the enclosing cylindrical segments, cones and conical frustums form a monotone chain in 3D. This (more general) problem has other applications, e.g., in fitting a branch of a tree or a neuron. In any case, our algorithms in this paper can be used as efficient subroutines for this general problem.

Acknowledgments

We gratefully acknowledge the support from the NSF of China (No. 60703028, No. 60473103, No. 60473127), the National Basic Research Program of China (No. 2006CB303102) and K.C. Wong Education Foundation, Hong Kong. Special thanks to anonymous referees for several useful comments.

References

[AAS97] P. Agarwal, B. Aronov and M. Sharir. Line transversals of balls and smallest enclosing cylinders in three dimensions, In *Proc. 8th ACM-SIAM Symp on Discrete Algorithms (SODA'97)*, New Orleans, LA, pages 483-492, Jan, 1997.

[AFS06] M. Attene, B. Falcidieno and M. Spagnuolo. Hierarchical mesh segmentation based on fitting primitives, *The Visual Computer*, 22(3):181-193, 2006.

[ASC+06] D. Albocher, U. Sarel, Y-K Choi, G. Elber and W. Wang. Efficient continuous collision detection for bounding boxes under rational motion, In *Proc. 2006 IEEE Intl. Conf. on Robotics and Automation (ICRA'06)*, pages 3017-3022, 2006.

[Be04] S. Bereg. Cylindrical hierarchy for deforming necklaces, *Intl. J. of Computational Geometry and Applications*, 14(1-2):3-17, 2004.

[BS01] G. Barequet and S. Har-Peled. Efficiently approximating the minimum-volume bounding box of a point set in three dimensions, *J. Algorithms*, 38:91-109, 2001.

[Ch96] T. Chan. Optimal output-sensitive convex hull algorithms in two and three dimensions, *Discrete and Computational Geometry*, **16**:361-368, 1996.

[Ch02] T. Chan. Approximating the diameter, width, smallest enclosing cylinder, and minimum-width annulus, *Intl. J. of Computational Geometry and Applications*, 12(1-2):67-85, 2002.

[CLRS01] T. Cormen, C. Leiserson, R. Rivest, C. Stein. *Introduction to Algorithms*, second edition, MIT Press, 2001.

[JT96] G. Jacobs and F. Theunissen. Functional organization of a neural map in the cricket cercal sensory system, *J. of Neuroscience*, 16(2):769-784, 1996.

[JT00] G. Jacobs and F. Theunissen. Extraction of sensory parameters from a neural map by primary sensory interneurons, *J. of Neuroscience*, 20(8):2934-2943, 2000.

[KK89] C. Kenyon-Mathieu and V. King. Verifying Partial Orders, In *Proceedings of the 21st Annual Symposium on Theory of Computing (STOC'89)*, pages 367–374, 1989.

[WK05] J. Wu and J. Kobbelt, Structure recovery via hybrid variational surface approximation, *Comput. Graph. Forum*, 24(3):277-284, 2005.

[LZJO04] W. Lin, B. Zhu, G. Jacobs and G. Orser. Cylindrical approximation of a neuron from reconstructed polyhedron, In *Proc. Intl. Conf. Computational Science and Applications*, LNCS 3045, pp. 257-266, 2004.

[PDJ99] S. Paydar, C. Doan and G. Jacobs. Neural mapping of direction and frequency in the cricket cercal sensory system, *J. of Neuroscience*, 19(5):1771-1781, 1999.

[PS85] F.P. Preparata and M.I. Shamos. *Computational Geometry: An Introduction*. Springer-Verlag, 1985.

[SSTY00] E. Schömer, J. Sellen, M. Teichmann and C.K. Yap. Smallest enclosing cylinders, *Algorithmica*, 27:170-186, 2000.

[VMC97] T. Varady, R. Martin and J. Cox. Reverse engineering of geometric models — an introduction. *CAD*, 29(4):255-268, 1997.

[ZS99] Y. Zhou and S. Suri. Analysis of a bounding heuristic for object intersection, *J. ACM*, 46(6):833-857, 1999.

[Zh97] B. Zhu. Approximating convex polyhedra with axis-parallel boxes. *Intl. J. of Computational Geometry and Applications*, 7(3):253-267, 1997.

[Zh04] B. Zhu. Approximating 3D points with cylindrical segments, *Intl. J. of Computational Geometry and Applications*, 14(3):189-201, 2004.