ELSEVIER

Computer-Aided Design 33 (2001) 873–878

COMPUTER-AIDED DESIGN

www.elsevier.com/locate/cad

A linear-time, constant-space algorithm for computing the spanning line segments in three dimensions

K.-L. Chung[a,1,*], S.-M. Chang[a], T.C. Woo[b]

[a]*Department of Information Management, Institute of Computer Science and Information Engineering, National Taiwan University of Science and Technology, No. 43, Section 4, Keelung Road, Taipei 10672, Taiwan, ROC*

[b]*Department of Industrial Engineering, University of Washington, Seattle, WA 98195-2650, USA*

Received 5 October 1999; revised 28 July 2000; accepted 7 August 2000

Abstract

Given a polyhedron, the set of spanning line segments (SLS) satisfies two properties: (1) completeness property — the set SLS must cover all the extreme vertices on the convex hull of the polyhedron; (2) inseparability property — there cannot be a plane that separates the set SLS into two nonempty subsets without intersecting one of them. Owing to the above two properties, the set SLS is proposed as a representation for testing the intersection between a plane and a three-dimensional (3D) polyhedron. Given a 3D polyhedron with N vertices, this paper presents an incremental $O(N)$-time algorithm for constructing the set SLS. The proposed algorithm has the same time complexity as the previous best result [Wang ME, Woo TC, Chen LL, Chou SY. Computing spanning line segments in three dimensions, The Visual Computer 12 (1996) 173–180], but it reduces the working memory required in the previous work from $O(N)$ to $O(1)$.

Keywords: Algorithms; Spanning line segments; 3D convex polyhedron

1. Introduction

Computing plane intersections with geometric objects is a key capability of CAD/CAM systems and many other geometric-modeling systems. We can obtain most conventional cross-sections of parts by computing the appropriate plane intersections [1]. In this research, we consider to test whether there is a possible intersection between a plane (from the bounding box or the convex hull) and another three-dimensional (3D) polyhedron. Intuitively, we first find the minima and the maxima in the x, y, and z directions of the vertices of the polyhedron, then create a box to bound the polyhedron. If there is no intersection between the bounding box and the plane, there is no intersection. The converse, however, can produce "false alarms". Fig. 1 illustrates the situation: when there is an intersection between the bounding box and the plane, it needs a further intersection test to check whether the polyhedron intersects the plane or not.

The idea of the spanning line segments (SLS) refines the intersection test [3].

Definition 1. [3] Given a polyhedron, the set SLS satisfies two properties: (1) completeness property — the set SLS must cover all the extreme vertices on the convex hull of the polyhedron; (2) inseparability property — there cannot be a plane that separates the set SLS into two nonempty subsets without intersecting one of them.

From Definition 1, an intersection between the given plane and the set SLS reports unambiguously.

The set SLS can be found from the line segments connecting the N extreme vertices which lie on the convex hull of the given polyhedron. In Ref. [4], the set SLS is determined in $O(N)$ time, but the size of the set SLS may not be minimal. It turns out that the size of the set SLS ranges from $\lceil N/2 \rceil$ to $N(N-1)/2$ and the *minimal* size of the set SLS is $\lfloor N/2 \rfloor + 1$. Wang et al. [5] present an $O(N)$-time algorithm for finding such a minimal set SLS while using $O(N)$ working memory. This paper presents an incremental linear-time algorithm while reducing the working memory required in Ref. [5] from $O(N)$ to

* Corresponding author. Tel.: +886-2-27376771; fax: +886-2-27376771.

E-mail address: klchung@cs.ntust.edu.tw (K.-L. Chung).

[1] The work is supported by NSC89-2213-E011-061. Partial work on this paper was done when the first author was at Washington University, USA.

0010-4485/01/$ - see front matter
PII: S0010-4485(00)00111-1

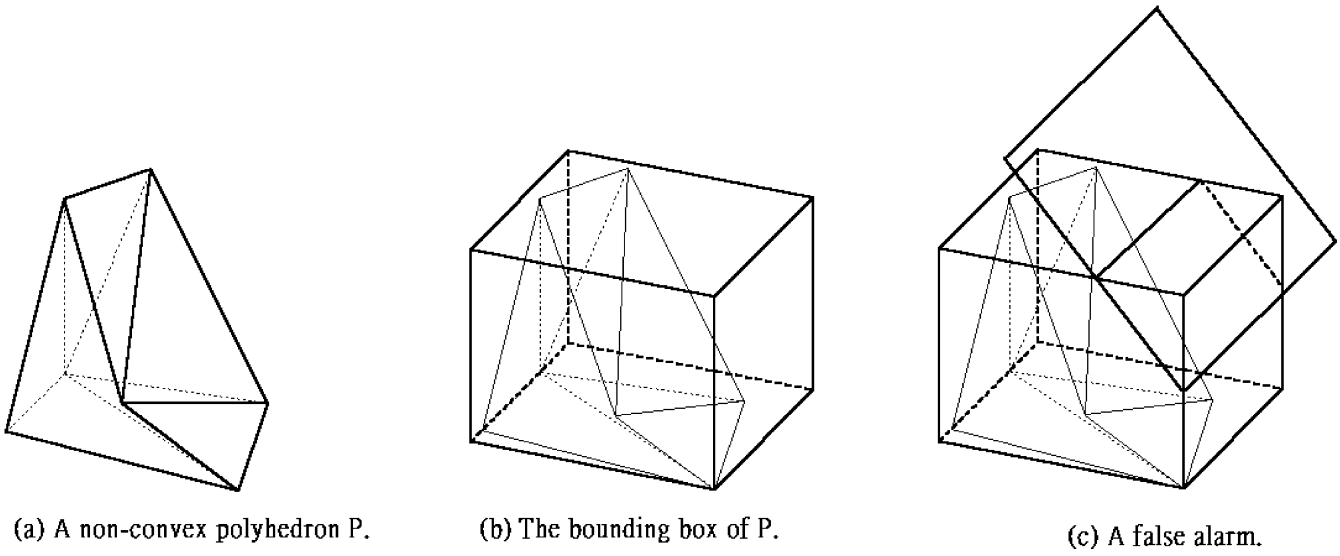

Fig. 1. An example of a polyhedron and its bounding box. (a) A non-convex polyhedron *P*. (b) The bounding box of *P*. (c) A false alarm.

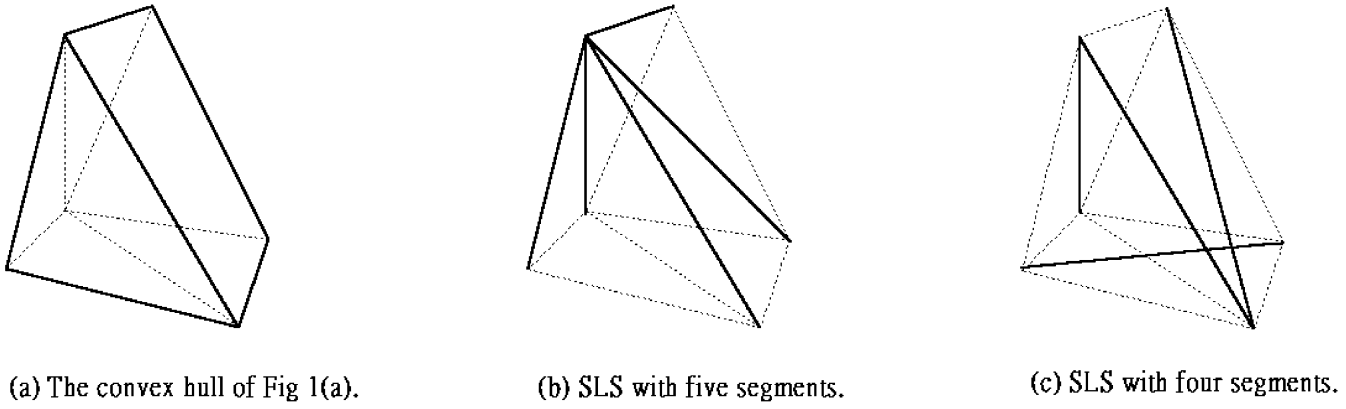

Fig. 2. The set SLS for Fig. 1(a). (a) The convex hull of Fig. 1(a). (b) SLS with five segments. (c) SLS with four segments.

$O(1)$. Continuing with the same polyhedron as shown in Fig. 1(a), we show two SLS sets for it in Fig. 2, one with five SLS in bold and the other with four.

The rest of this paper is organized as follows. Some of the preliminaries are reviewed in Section 2. The proposed algorithm is presented in Section 3, and we prove the correctness in Section 4.

2. Previous work

The algorithm in Ref. [5] is an iterative procedure consisting of: constructing the convex hull, finding a pair of nonadjacent vertices, removing the pair, and then adding the spanning line segment connecting the pair to the set SLS until the number of the vertices in the given polyhedron is less than five. Upon the termination condition, it obtains the last three SLS by connecting the remaining vertices in an arbitrary order. The size of the set SLS thus obtained, $\lfloor N/2 \rfloor + 1$, is minimal.

For clarity, an example to demonstrate the algorithm in Ref. [5] is useful. Given a polyhedron, its convex hull is first found and shown in Fig. 3(b). Initially, the set SLS is $\{\phi\}$. Later, we traverse the representation of the convex polyhedron and put the related information into a linked-list. Then a pair of nonadjacent vertices is found (see Fig. 3(c)), and the set SLS is now $\{L_1\}$. After removing the vertices of L_1 from the convex hull, the remaining convex hull is updated and is shown by bold lines in Fig. 3(c). We get the second SLS, L_2, from the convex polyhedron in Fig. 3(c) and the set SLS is now $\{L_1,L_2\}$, as shown in Fig. 3(d). After removing L_2, the remaining convex hull denoted by solid lines is shown in Fig. 3(d). As illustrated in Fig. 3(e), we get the third SLS, L_3, by connecting two nonadjacent vertices of the convex hull in Fig. 3(d) and now the number of remaining vertices is four. Finally, we select a chain of three line segments L_4, L_5 and L_6 which join the four remaining vertices. The resulting set SLS for this example is shown in Fig. 3(f) and it is represented by $\{L_1, L_2, L_3, L_4, L_5, L_6\}$.

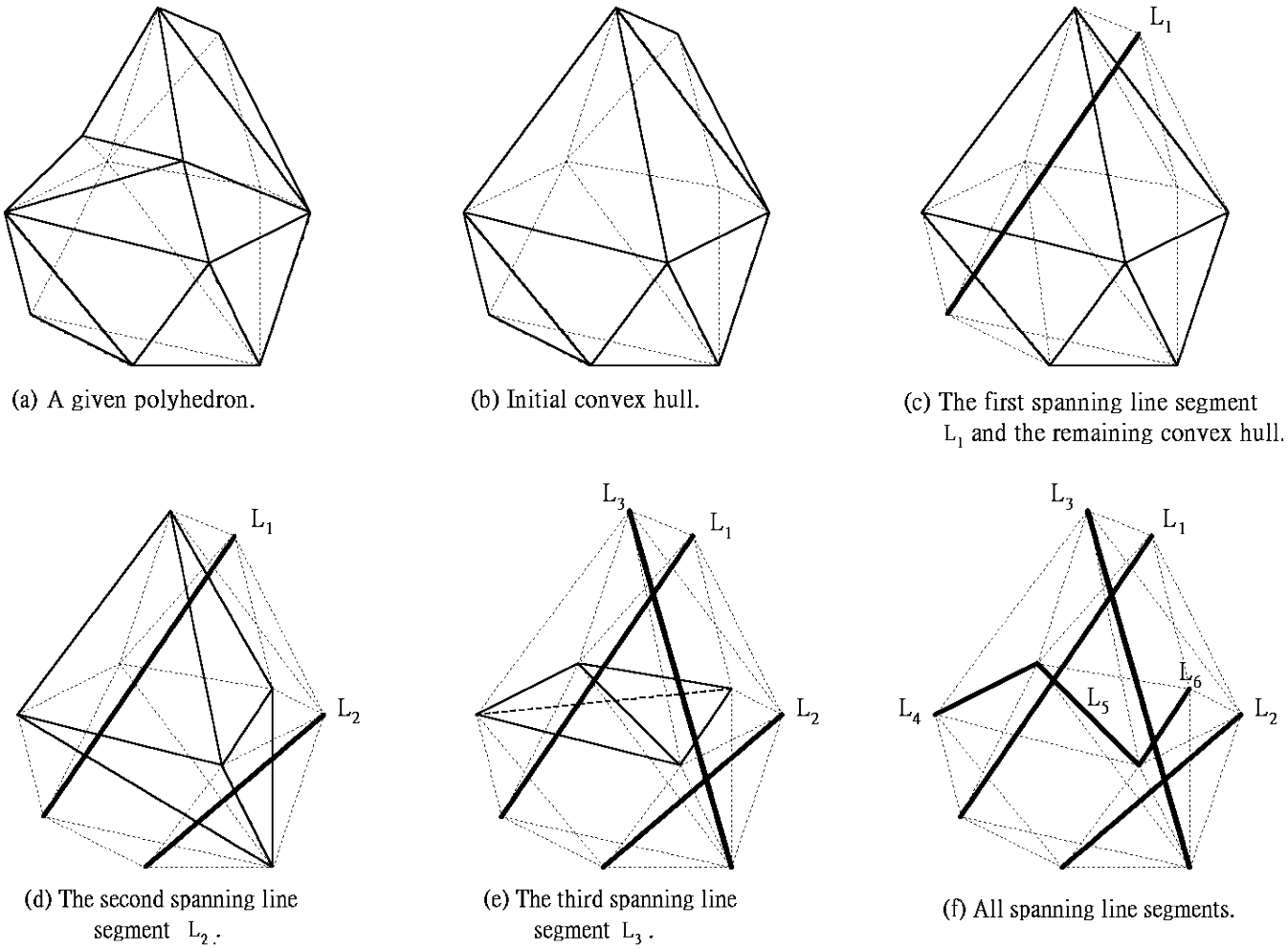

Fig. 3. A simulation of Wang et al.'s algorithm. (a) A given polyhedron. (b) Initial convex hull. (c) The first spanning line segment $\mathscr{L}_1$ and the remaining convex hull. (d) The second spanning line segment $\mathscr{L}_2$. (e) The third spanning line segment $\mathscr{L}_3$. (f) All spanning line segments.

In Ref. [5], the construction of the convex hull for the given polyhedron takes $O(N)$ time and this construction is incurred only once; the subsequent convex hulls are updated, incurring constant time. The total time is $O(N)$. Since, in each iteration, the convex hull of the remaining vertices is kept in memory, it takes $O(N)$ space.

Finally, we cite the following lemma from Ref. [5]. Here, we simplify their proof.

Lemma 1. *In a convex polyhedron P with vertices set V, there always exists two non-adjacent vertices p and q in V. Let S be the set of spanning line segments for the convex hull of the set V-{p,q}. Then adding the line segment (p,q) to S spans P.*

Proof. First, the line segment (p,q) is obtained by connecting p and q. It is obvious that the union $S \cup \{(p,q)\}$ satisfies the completeness property. Since p and q are non-adjacent, the line segment (p,q) intersects polyhedron with vertices set $V - \{p,q\}$. Because of the inseparable property in S, the union $S \cup \{(p,q)\}$ satisfies the inseparability property. This completes the proof. □

3. A constant space algorithm

The main concept of the proposed incremental algorithm is that, instead of maintaining all the remaining vertices, we keep a small convex polyhedron of a constant vertex size, say 5. We call it the "kernel'' Q. By Lemma 1 and the concept of the kernel, we have the following algorithm.

We first take five vertices from the given polyhedron P and construct the kernel Q using $O(1)$ time. By Lemma 1, we can find a pair of nonadjacent vertices in Q using $O(1)$ time. Connect this pair by a line segment, say $\mathscr{L}_1$. We output the line segment L_1 and remove the two corresponding vertices from the five vertices. The set SLS is now $\{\mathscr{L}_1\}$. We then add two arbitrary vertices, p and q, from the set of vertices in P to the three vertices in Q. We construct a new kernel from the set of five vertices. Again, by Lemma 1, a pair of nonadjacent vertices in the new kernel can be found in $O(1)$ time and another line segment

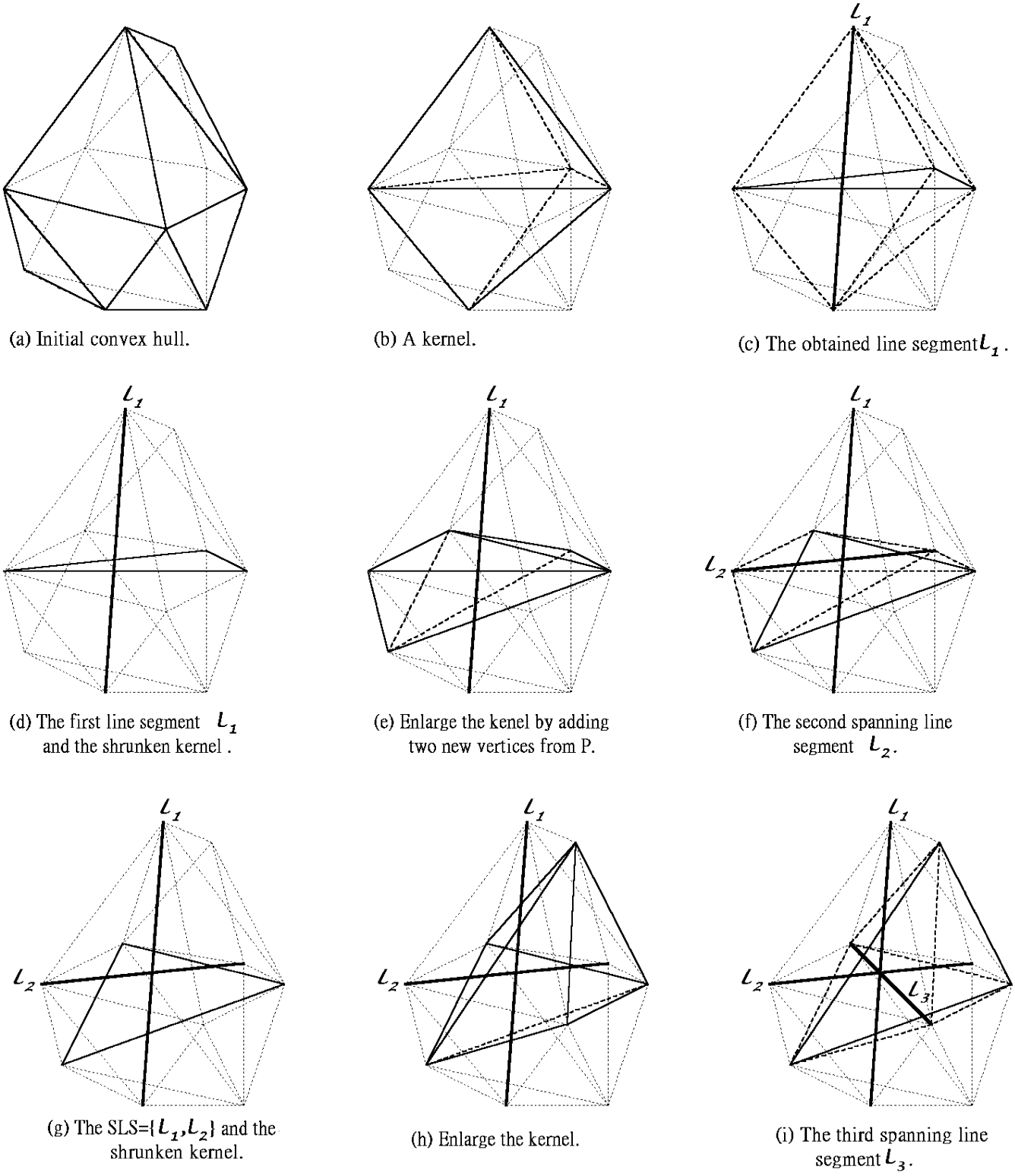

Fig. 4. A simulation of the proposed algorithm. (a) Initial convex hull. (b) A kernel. (c) The obtained line segment $\mathscr{L}_1$. (d) The first line segment $\mathscr{L}_1$ and the shrunken kernel. (e) Enlarge the kernel by adding two new vertices from P. (f) The second spanning line segment $\mathscr{L}_2$. (g) The SLS = $\{\mathscr{L}_1,\mathscr{L}_2\}$ and the shrunken kernel. (h) Enlarge the kernel. (i) The third spanning line segment $\mathscr{L}_3$..

$\mathscr{L}_2$ is obtained. The set SLS now becomes $\{\mathscr{L}_1,\mathscr{L}_2\}$. We then remove the two vertices p and q from Q.

Repeating the above process, we add two new vertices from the given polyhedron to the kernel. Then, we find a pair of nonadjacent vertices from the newly constructed kernel and output the corresponding line segment to the set SLS. Finally, we remove the pair of vertices from the kernel. These steps take constant time and are continued until the number of the vertices of the given polyhedron is less than five. Since a kernel of no more than five vertices is kept in memory while processing, the algorithm takes $O(1)$ space. The procedure is illustrated in Fig. 4, in which the final kernel is of size less than five, and where we obtain three line segments.

A formal algorithm is given below:

Algorithm MSLS: Minimal Spanning Line Segments
Input: a convex polyhedron P of N vertices.
Output: the set SLS of size $\lfloor N/2 \rfloor + 1$.
Begin
Step 0 Initialize SLS $= \{\phi\}$.
Step 1 Select five vertices from the given vertex set of P randomly; delete them from P.
Step 2 Repeat until the number of vertices is less than five:
2.1 Construct a kernel Q from the five vertices and identify two nonadjacent vertices from it.
2.2 Output the line segment $\mathscr{L}$ joining two nonadjacent vertices to the set SLS, i.e. SLS $=$ SLS $\cup \{L\}$. Delete the two nonadjacent vertices from Q.
2.3 Choose any two vertices from P and add them to Q.
Step 3 Compute the spanning line segments for the remaining four (or three) vertices and add them to the set SLS.
End

4. Correctness and performance of the algorithm

Recall the two properties of the set SLS: *completeness* (of covering all extreme vertices of the given polyhedron P) and *inseparability* (by a plane which intersects P but not the set SLS). To show the correctness of the algorithm *MSLS*, the following new lemma will be needed for the correctness proof and performance discussion.

Lemma 2. *Given a polyhedron with five or more vertices, there is always a pair of nonadjacent vertices.*

Proof. We prove by contradiction. Let V, E, and F denote the number of vertices, edges and faces of the polyhedron, respectively. From the well-known Euler formula [2], there are two relations: (1) $V - E + F = 2$ and (2) $3F \leq 2E$. It is easy to obtain the relation $E \leq 3V - 6$. Assume a polyhedron with $V \geq 5$ vertices being adjacent to each other, i.e. we have $E = V(V-1)/2$. From $E \leq 3V - 6$, we have $V(V-1)/2 \leq 3V - 6$. Using some algebraic manipulations, we have $3 \leq V \leq 4$, which contradicts the assertion. □

We are now ready to show that the set SLS obtained by the algorithm *MSLS* satisfies both completeness and inseparability.

Theorem 1. *Given a convex polyhedron with N vertices, the algorithm MSLS obtains the set SLS correctly.*

Proof. The property of completeness is satisfied since, in each iteration, two nonadjacent vertices are connected by a SLS, then these two vertices are removed from the polyhedron. Furthermore, the last kernel with either three or four vertices is partitioned into either two or three line segments.

We now prove that the property of inseparability is also satisfied by the set SLS. By Lemma 2, it is shown that in each iteration of Step 2, we always can find a pair of nonadjacent vertices in the kernel. We thus get the first line segment in the first iteration of Step 2, say $\mathscr{L}_1$, from the kernel Q constructed from five vertices. From Lemma 1, it can be shown that $\mathscr{L}_1$ spans Q and this implies that a plane intersects the kernel Q if and only if it intersects either the line segment $\mathscr{L}_1$ or the kernel constructed from the remaining three vertices. This confirms that the obtained line segment $\mathscr{L}_1$ satisfies the inseparability property for Q.

By the same argument, any new line segment connecting two nonadjacent vertices always intersects the "shrunken'' kernel (with three vertices). Before obtaining the last two or three line segments from it, we have a set of line segments, $\{\mathscr{L}_1, \mathscr{L}_2, \mathscr{L}_3, \ldots, \mathscr{L}_{\lceil (N-4)/2 \rceil}\}$,which spans Q. Either three vertices or four vertices remain in Q, depending on whether N is even or odd. For the three (four) vertices case, an arbitrary chain of two (three) line segments are found and put into the set SLS. These line segments consequently satisfy the inseparability property. □

We next examine the performance of the algorithm. If N is odd, $(N-3)/2$ iterations are needed in Step 2 and $(N-3)/2$ line segments are obtained before the final Step 3 which gives an additional two line segments. Therefore, the total size of the set SLS is equal to $(N-3)/2 + 2 = \lfloor N/2 \rfloor + 1$ when N is odd. If N is even, $(N-4)/2$ iterations are needed in Step 2 and $(N-4)/2$ line segments are obtained. And then a chain of three line segments is obtained in Step 3 so that the size of the set SLS is $(N-4)/2 + 3 = \lfloor N/2 \rfloor + 1$ when N is even. Since we consider only five vertices in each iteration and $\lceil (N-4)/2 \rceil$ iterations are executed in Step 2, the working memory and time complexity is $O(1)$ for Q and $O(N)$ for P. Consequently, we have the main result.

Theorem 2. *Given a convex polyhedron P with N vertices, the algorithm MSLS obtains the set SLS of size* $\lfloor N/2 \rfloor + 1$ *for P in O(N) time using O(1) space.*

We use the convex polyhedron consisting of two tetrahedrons, say A and B, one facet of A touching one facet of B side by side, to compare the execution time performance of the proposed algorithm and the previous algorithm [5]. The experiment is performed on the IBM compatible personal computer Pentium III microprocessor with 500 MHz, where the RAM is of size 128 MB. The language used is Visual C++ and the environment is Window 98. For the same input, the proposed algorithm needs 10 ms while the previous algorithm [5] needs 34 ms. The experimental results reveals that although the time complexity in terms of big-O notation is the same for both algorithms, the proposed algorithm is faster than the previous algorithm in practice. The main reason is that the concerning manipulations on the constant-space working memory used in the proposed algorithm are much simpler than those on the $O(N)$-space working memory in the previous algorithm.

Acknowledgements

The authors are indebted to the anonymous reviewers and Editor-in-Chief Prof. Les A. Piegl for making some valuable suggestions and corrections that led to the improved version of the paper. We also appreciate Wan-Yue Chen for her programming help.

References

[1] Mortenson ME. Geometric modeling. New York: Wiley, 1985 (chap. 7, Intersections).
[2] Harary F. Graph theory. Reading, MA: Addison-Wesley, 1972.
[3] Wang JY, Yuen LD, Woo TC. A method for testing intersection between plane and polyhedron, Technical Report, GINTIC Institute of Manufacturing Technology, Nanyang Technological University, Singapore, 1993.
[4] Wang JY, Yuen LD, Woo TC, Chen XM. The spanning line segments of a polyhedron. ASME Trans. J. Mech. Design 1996;118:40–44.
[5] Wang ME, Woo TC, Chen LL, Chou SY. Computing spanning line segments in three dimensions. The Visual Computer 1996;12:173–80.

Kuo-Liang Chung received the BS, MS, and PhD degrees in Department of Computer Science and Information Engineering from National Taiwan University, ROC. He has been a professor in the Department of Information Management and Institute of Computer Science and Information Engineering of the National Taiwan University of Science and Technology since 1995. His current research interests include image/video processing, compression, computer graphics, and theoretical computer science. He is a member of IEEE.

Shyh-Ming Chang received the BS degree in Computer Science and Information Engineering in Computer Engineering and Science from Yuan-Ze University and the MS degree in Institute of Computer Science and Information Engineering in 1999 from the National Taiwan University of Science and Technology. His current research interests include image processing and computer graphics.

Tony Woo is professor and director of Industrial Engineering as well as Fluke Chair of Manufacturing Engineering at the University of Washington. His research interest includes computer graphics and computational geometry.