Inserting Line Segments into Triangulations and Tetrahedralizations

Javier Bernal
National Institue of Standards and Technology, Gaithersburg, MD 20899, U. S. A.

Abstract

An algorithm by Bernal, De Floriani, and Puppo, for inserting a line segment into a Constrained Delaunay triangulation is further developed. The new version of the algorithm inserts the line segment in exactly the same manner in which the old one does but has the additional capability that it does not delete the triangles intersected by the line segment but transforms them through edge-swapping. Since the concept of edge-swapping generalizes to 3−dimensional space, a version of the algorithm without the optimization steps for the Delaunay property is also proposed for attempting to insert a line segment into a tetrahedralization. A result is then presented that shows that for certain cases the failure of this algorithm to insert a line segment is an indication that it can not be done. Finally, 3−dimensional problems that can be approached as 2−dimensional problems are identified.

1. Introduction

A *triangulation* for a finite set of points S in the plane is a finite collection of triangles in the plane having pair-wise disjoint interiors, each of which intersects S exactly at its vertices, and the union of which is the convex hull of S. Given a triangulation T for S, we say that T is *Delaunay* if for each triangle in T there does not exist a point of S inside the circumcircle of the triangle [11]. A *(Delaunay) tetrahedralization* is similarly defined with tetrahedra and spheres taking the place of triangles and circles.

A more general triangulation can be defined. Let S be as above, and let E be a finite collection, possibly empty, of line segments with endpoints in S that intersect only at points in S. We say that a triangulation T for S is *constrained by* E if each line segment in E is the union of edges in T. Given T, a triangulation for S constrained by E, we say that T is *Delaunay constrained by* E if for each t in T there does not exist a point P of S inside the circumcircle of t such that no line segment in E intersects the interior of the convex hull of $t \cup \{P\}$.

Let E be as above. Given T, a triangulation constrained by E, we say that T *satisfies the empty circle criterion on a local basis* if given any two triangles t, t' in T that share a common edge not contained in any line segment in E, then the vertex in $t' \setminus t$ is not inside the circumcircle of t. That triangulations of this type and constrained Delaunay triangulations are equivalent has been proven in [5], [9].

Algorithms for the computation of a Delaunay triangulation for the vertices of a polygon constrained by the boundary of the polygon have been presented in [5], [9], [10]. As for the general problem of computing a Delaunay triangulation for a set of n points constrained by a set of line segments, an $O(n^2)$ algorithm has been presented in [9], $O(n \log n)$ divide-and-conquer algorithms have been presented in [4], [13], and an $O(n \log n)$ plane-sweep algorithm has been presented in [12]. Each one of these algorithms has the disadvantage that the set of line segments must be known before the execution of the algorithm.

In [6] a method has been presented for the incremental computation of a constrained Delaunay triangulation by stepwise insertion of points and line segments. Accordingly, algorithms are presented in [6] for point insertion and line segment insertion into a constrained Delaunay triangulation. Independently, the algorithm for line segment insertion was also presented in [1]. In the following section, we describe a new version of this algorithm that works in the same manner in which the old one does, but that has the additional capability of not deleting the triangles intersected by the line segment, transforming them instead through edge-swapping (Lawson's transformation [8]). In Section 3, we take advantage of the fact that edge-swapping generalizes to 3−dimensional space and propose what would be considered the generalization to 3−dimensional space of the algorithm without the optimization steps for the Delaunay property. A result is then presented that shows that for certain cases the failure of this algorithm to insert a line segment into a tetrahedralization is an indication that it cannot be done. Finally, in the same section, 3−dimensional problems are identified that can be approached algorithmically as if they are 2−dimensional.

2. Segment insertion by edge-swapping

Let T be a triangulation in the plane, not necessarily Delaunay, let P_1, P_2, $P_1 \neq P_2$, be vertices in T, and let T^* be the triangles in T whose interiors are intersected by $\overline{P_1P_2}$, i. e. the line segment with endpoints P_1, P_2. We say that $\overline{P_1P_2}$ *has been inserted into* T *producing* $\hat{T}$ if $\hat{T}$ is a triangulation for the vertices of T such that $\overline{P_1P_2}$ is the union of edges in $\hat{T}$ and each triangle in $T \setminus T^*$ is also in $\hat{T}$. In what follows, and assuming that T is constrained Delaunay, we present procedure INSERT_SEGMENT which inserts $\overline{P_1P_2}$ into the triangulation T by edge-swapping, producing a constrained Delaunay triangulation with $\overline{P_1P_2}$ as an additional constraint. Without any loss of generality, we assume that $\overline{P_1P_2}$ is not an edge in T and that its relative interior does not contain any vertices in T.

In [1] and [6] this algorithm was presented but without edge-swapping. This older version consists essentially of two steps. In the first step, the triangles whose interiors are intersected by the line segment are detected and deleted so that a non-triangulated region inside the convex hull of the original triangulation results. In the second step, this region is divided into two polygons separated by the line segment, and a Delaunay triangulation is then computed for each polygon. Each polygon satisfies the property that each point in the polygon is visible through the polygon from the line segment. Because of this property, each polygon can be easily triangulated in a linear fashion, and then optimized for the Delaunay property with procedures based on the empty circle criterion. Outlines of this older version, justifications, optimization procedures, and related results can be found in [1], [2], [6], [7].

The new version of the algorithm presented here works essentially in the same manner in which the old one does, thus producing exactly the same triangles, but has the capability through edge-swapping of maintaining at all times a complete triangulation. Let T, P_1, P_2, T^* be as above. Essentially, ignoring the optimization steps, the new version of the algorithm works as follows. For some integer n, let t_i, $i = 1, \ldots, n$, be the triangles in T^* in the order in which they are intersected by $\overline{P_1P_2}$ from P_1 to P_2. Inductively, assume that for an integer i, $1 \leq i \leq n-1$, the triangles t_j, $j = 1, \ldots, i$, have been processed in that order by the algorithm and that T^* and T have been transformed accordingly. Triangle t_{i+1}, which still belongs to T^*, is then processed as follows. Triangle t' is initialized to t_{i+1}. Triangle t'' is set equal to the triangle in T^*, not necessarily t_i, that currently shares a facet with t' intersected by $\overline{P_1P_2}$ and which is closer to P_1 than t' in the direction of $\overline{P_1P_2}$. If $t' \cup t''$ is not a strictly convex quadrilateral then the algorithm is done processing t_{i+1}. Otherwise t', t'', and therefore T^* and T, are transformed through the replacement of the common edge by the alternative diagonal of the quadrilateral. If only one of the two new triangles is intersected by $\overline{P_1P_2}$ then t' is redefined as the one that is intersected. Otherwise it is redefined as the one of the two triangles that is closer to P_1 in the direction of $\overline{P_1P_2}$. The process above is repeated for the new t', i. e. t'' is redefined, etc. until t'' does not exist as desired or $t' \cup t''$ is not a strictly convex quadrilateral. The insertion of $\overline{P_1P_2}$ into T is accomplished as soon as t_n is processed by the algorithm.

Let T, P_1, P_2, T^* be as above. In the following, we list and describe, in the order of their first appearance, procedures used in INSERT_SEGMENT as primitives.

INTERSECTED_TRIANGLES$(T, T^*, P_1, P_2, Q, t_F)$: This procedure identifies T^*. It also locates t_F in T^* with P_1 as one of its vertices and a vertex Q of t_F different from P_1.

NEXT_TRIANGLE(T, P_1, P_2, t_P, t_C): Assuming that $\overline{P_1P_2}$ intersects the interior of t_P, t_P in T, $P_2 \notin t_P$, this procedure locates t_C in T which shares a facet with t_P intersected by $\overline{P_1P_2}$, and which is closer to P_2 than t_P in the direction of $\overline{P_1P_2}$.

NEXT_VERTEX(t_P, t_C, P): Assuming that triangles t_P and t_C share a facet, this procedure locates vertex P of t_C not in t_P.

PREVIOUS_VERTEX(t_C, P_1, P_2, P, Q): Assuming that P is a vertex of triangle t_C and that $\overline{P_1P_2}$ intersects exactly one of the edges of t_C with P as an endpoint, this procedure locates the vertex Q of t_C for which $\overline{P_1P_2}$ does not intersect $\overline{PQ}$.

STRICT_CONVEXITY$(t_C, t_P, flag2)$: Assuming that triangles t_C and t_P share a facet, this procedure sets $flag2$ to zero whenever $t_C \cup t_P$ is not a strictly convex quadrilateral.

EDGE_SWAP(t_C, t_P, Q, T, T^*): Assuming that $t_C \cup t_P$ is a strictly convex quadrilateral, t_C, t_P in T^*, and that Q is a vertex in $t_C \cap t_P$, this procedure transforms t_C, t_P, and therefore T^* and T, through the replacement of the common edge by the alternative diagonal of the quadrilateral in such a way that Q is the vertex of the transformed t_P not in the transformed t_C.

OPTIMIZE(T, T^*, t_P, P, Q, R): Assuming that P, Q, R are the vertices of t_P, t_P in T^*, starting with t_P this procedure transforms T^*, and therefore T, through applications of the empty circle criterion and edge-swapping in the direction of $\overline{QR}$.

PREVIOUS_TRIANGLE(T, P_1, P_2, t_C, t_P): Assuming that $\overline{P_1P_2}$ intersects the interior of t_C, t_C in T, $P_1 \notin t_C$, this procedure locates t_P in T which shares a facet with t_C intersected by $\overline{P_1P_2}$, and which is closer to P_1 than t_C in the direction of $\overline{P_1P_2}$.

THIRD_VERTEX(t_C, R, P, Q): Assuming that R, P are distinct vertices of triangle t_C, this procedure identifies Q, the vertex of t_C different from R and P.

The outline of INSERT_SEGMENT follows. We notice that without the optimization steps (steps 20 and 32) the procedure simply becomes one for inserting a line segment into a triangulation.

```
    procedure INSERT_SEGMENT(T, P1, P2)
        begin
1.      INTERSECTED_TRIANGLES(T, T*, P1, P2, Q, tF);
2.      F(1, tF) := P1; F(2, tF) := Q; flag1 := 1;
3.      while (flag1 = 1) do
            begin
4.          tP := tF;
5.          NEXT_TRIANGLE(T, P1, P2, tP, tC);
6.          NEXT_VERTEX(tP, tC, P);
7.          if (P ≠ P2) then
                begin
8.              PREVIOUS_VERTEX(tC, P1, P2, P, Q);
9.              tF := tC
                end
            else
                begin
10.             Q := F(2, tP); flag1 := 0
                end
11.         if (F(1, tP) = P1) then F(2, tP) := Q;
12.         F(1, tC) := Q; F(2, tC) := P; flag2 := 1;
13.         while (flag2 = 1) do
                begin
14.             STRICT_CONVEXITY(tC, tP, flag2);
15.             if (flag2 = 1) then
                    begin
16.                 R := F(1, tP); tL := tC;
17.                 EDGE_SWAP(tC, tP, Q, T, T*);
18.                 if (tF = tL) then tF := tC;
19.                 if (F(1, tC) = F(2, tP)) then
                        begin
20.                     OPTIMIZE(T, T*, tP, P, Q, R);
21.                     F(1, tC) := R; Q := R
                        end
                    else
                        begin
22.                     F(1, tC) := R; F(2, tC) := F(2, tP);
23.                     F(1, tP) := Q; F(2, tP) := P;tC := tP
                        end
24.                 if (R ≠ P1) then
                        begin
25.                     PREVIOUS_TRIANGLE(T, P1, P2, tC, tP);
26.                     if (F(1, tP) = P1) then F(2, tP) := Q;
27.                     if (P = P2) then
                            begin
28.                         Q := F(2, tP); F(1, tC) := Q
                            end
                        end
                    else
                        begin
29.                     flag2 := 0;
30.                     if (P = P2) then
                            begin
31.                         THIRD_VERTEX(tC, R, P, Q);
32.                         OPTIMIZE(T, T*, tC, P, Q, R)
                            end
```

```
                        else
                            begin
33.                         NEXT_TRIANGLE(T, P1, P2, tC, tN);
34.                         F(2, tC) := F(1, tN)
                            end
                        end
                    end
                end
            end
        end
```

Justifications of this procedure appear in [1], [3], [7].

3. The 3−dimensional version of the algorithm

Let T be a tetrahedralization, not necessarily Delaunay, let P_1, P_2, $P_1 \neq P_2$, be vertices in T, and let T^* be the tetrahedra in T each of which is intersected by $\overline{P_1P_2}$ at either its interior or the relative interior of one of its facets. We say that $\overline{P_1P_2}$ *can be inserted into* T if a tetrahedralization $\hat{T}$ for the vertices of T exists such that $\overline{P_1P_2}$ is the union of edges in $\hat{T}$ and each tetrahedron in $T \setminus T^*$ is also in $\hat{T}$. In what follows, we present procedure 3D_INSERT_ATTEMPT which attempts to insert $\overline{P_1P_2}$ into T, and which can be considered as the generalization to 3−dimensional space of INSERT_SEGMENT without the optimization steps. We notice that only the case for which the relative interior of $\overline{P_1P_2}$ does not intersect any edges in T is addressed in what follows.

Let T, P_1, P_2 be as above. In the following, we list and describe, in the order of their first appearance, procedures used in 3D_INSERT_ATTEMPT as primitives. Procedures with obvious 2−dimensional counterparts in INSERT_SEGMENT are neither listed nor described here.

FIRST_TETRAHEDRON(T, P_1, P_2, Q, t_F): This procedure locates t_F in T with P_1 as one of its vertices and interior intersected by $\overline{P_1P_2}$, and locates a vertex Q of t_F, $Q \neq P_1$.

COMMON_VERTEX(t_C, t_P, Q, S, U): Assuming that tetrahedra t_C and t_P share a facet, and that Q and S are vertices, not necessarily distinct, of the facet, this procedure locates U, a vertex of the facet different from Q and S.

TWO_THREE$(T, t_C, t_P, P, R, Q, U)$: Assuming that $t_C \cup t_P$ is a strictly convex hexahedron, t_C, t_P in T, that P is the vertex in $t_C \setminus t_P$, that R is the vertex in $t_P \setminus t_C$, and that Q, U, $Q \neq U$, are vertices in $t_C \cap t_P$, this procedure transforms T by transforming t_C and t_P into the three tetrahedra that have $\overline{PR}$ in common and whose union is the hexahedron, in such a way that t_C becomes the one of the three tetrahedra that does not have Q as a vertex, and t_P the one that has Q and U as vertices.

FACET_INTERSECT$(P, R, U, P_1, P_2, flag2)$: Assuming that P, R, U are the vertices of a facet of a tetrahedron, this procedure sets $flag2$ to zero whenever $\overline{P_1P_2}$ does not intersect the relative interior of the facet.

The outline of 3D_INSERT_ATTEMPT follows. Here a variable $flag$ is defined which at the end of the execution of the procedure equals 1 if $\overline{P_1P_2}$ has been inserted, zero otherwise.

```
    procedure 3D_INSERT_ATTEMPT(T, P1, P2, flag)
        begin
1.      flag := 0;
2.      FIRST_TETRAHEDRON(T, P1, P2, Q, tF);
3.      F(1, tF) := P1; F(2, tF) := Q; flag1 := 1;
4.      while (flag1 = 1) do
            begin
5.          tP := tF;
6.          NEXT_TETRAHEDRON(T, P1, P2, tP, tC);
7.          NEXT_VERTEX(tP, tC, P);
8.          if (P ≠ P2) then
                begin
9.              PREVIOUS_VERTEX(tC, P1, P2, P, Q);
10.             tF := tC
                end
            else
                begin
11.             Q := F(2, tP); flag1 := 0
```

```
            end
        if (F(1,t_P) = P_1) then F(2,t_P) := Q;
        F(1,t_C) := Q; F(2,t_C) := P; flag2 := 1;
        while (flag2 = 1) do
            begin
            STRICT_CONVEXITY(t_C, t_P, flag2);
            if (flag2 = 1) then
                begin
                R := F(1,t_P); S := F(2,t_P);
                COMMON_VERTEX(t_C, t_P, Q, S, U);
                if (F(1,t_C) = F(2,t_P)) then
                    begin
                    t_L := t_C;
                    TWO_THREE(T, t_C, t_P, P, R, Q, U);
                    if (t_F = t_L) then t_F := t_C;
                    F(1,t_C) := R; Q := R
                    end
                else
                    begin
                    FACET_INTERSECT(P, R, U, P_1, P_2, flag2);
                    if (flag2 = 1) then
                        begin
                        t_L := t_C;
                        TWO_THREE(T, t_C, t_P, P, R, Q, U);
                        if (t_F = t_L) then t_F := t_C;
                        F(1,t_C) := R; F(2,t_C) := F(2,t_P);
                        F(1,t_P) := Q; F(2,t_P) := P;t_C := t_P
                        end
                    end
                if (flag2 = 1) then
                    begin
                    if (R ≠ P_1) then
                        begin
                        PREVIOUS_TETRAHEDRON(T, P_1, P_2, t_C, t_P);
                        if (F(1,t_P) = P_1) then F(2,t_P) := Q;
                        if (P = P_2) then
                            begin
                            Q := F(2,t_P); F(1,t_C) := Q
                            end
                        end
                    else
                        begin
                        flag2 := 0;
                        if (P = P_2) then flag := 1
                        else
                            begin
                            NEXT_TETRAHEDRON(T, P_1, P_2, t_C, t_N);
                            F(2,t_C) := F(1,t_N)
                            end
                        end
                    end
                end
            end
        end
    end
```

Experiments show that 3D_INSERT_ATTEMPT seldom succeeds in inserting a line segment. However, this may just be an indication that it is seldom possible to insert a line segment into a tetrahedralization. Let T, P_1, P_2, T^* be as above. The following proposition shows that for a certain kind of T^* the failure of the procedure simply signifies that $\overline{P_1P_2}$ can not be inserted into T.

Proposition 1. If points Q_1, Q_2 exist, $Q_1 \neq Q_2$, that are vertices of every tetrahedron in T^*, then at the end of the execution of 3D_INSERT_ATTEMPT, variable $flag$ equals 1 if and only if $\overline{P_1P_2}$ can be inserted into T.

Proof. That $flag$ equal to 1 implies that the line segment can be inserted into T follows trivially. Thus, it remains to be shown that if $flag$ equals zero then the line segment can not be inserted into T.
For some positive integer n, let t_i, $i = 1, \ldots, n$, be the tetrahedra in T^* in the order in which they are intersected by the line segment from P_1 to P_2.
At the end of the execution of the procedure let T^{**} be the collection of tetrahedra in T that are intersected by the relative interior of the line segment, and for some positive integer m, let t'_i, $i = 1, \ldots, m$, be the tetrahedra in T^{**} in the order in which they are intersected by the line segment from P_1 to P_2.
Clearly, $n \geq m$, and since $flag$ equals zero it follows that $m \geq 3$.
Let R_0 equal P_1, and, inductively, for each i, $i = 1, \ldots, n$, let R_i be the vertex of t_i different from R_{i-1}, Q_1, and Q_2. Similarly, points R'_i, $i = 0, \ldots, m$ are defined with respect to t'_i, $i = 1, \ldots, m$.
We define a function f from $\{0, \ldots, m\}$ into $\{0, \ldots, n\}$ in such a way that for each i, $i = 0, \ldots, m$, R'_i equals $R_{f(i)}$. Based on this definition, for each i, $i = 1, \ldots, m$, we then define sets $W_i \subset \{R_0, \ldots, R_n\}$, by

$$W_i \equiv \{R_{f(i-1)} = R'_{i-1}, R_{f(i-1)+1}, \ldots, R_{f(i)} = R'_i\}.$$

From the definition of T^{**} it follows that given i, $2 \leq i \leq m$, the union of t'_{i-1} and t'_i is not a strictly convex hexahedron (step 15 of procedure). Thus, it is not possible to insert the line segment and at the same time to have a new tetrahedron in T with vertices Q_1, R'_{i-2}, R'_{i-1}, R'_i. The same is true for a tetrahedron with vertices Q_2, R'_{i-2}, R'_{i-1}, R'_i. From this and the fact that it is always true that $F(1, t_C)$ equals $F(2, t_P)$ in step 19 of the procedure, it follows that for each i, $i = 2, \ldots, m$, it is not possible to insert the line segment and at the same time to have a new tetrahedron with one vertex equal to either Q_1 or Q_2, two vertices in W_{i-1}, and one vertex in $W_i \setminus \{R'_{i-1}\}$.
In what follows, we assume that the line segment can be inserted into T. Thus, we must assume that T^* has been transformed in such a way that the line segment is one of its edges. Clearly, in the transformed T^*, which we denote by $\hat{T}^*$, only one tetrahedron can have both Q_1 and Q_2 as vertices, namely the tetrahedron with vertices Q_1, Q_2, P_1, and P_2. All other tetrahedra with either Q_1 or Q_2 as a vertex have in addition three vertices of the form R_j, R_k, R_l, $0 \leq j < k < l \leq n$.
For some integer n', $1 \leq n' < n$, we define integers h_i, l_i, $i = 0, \ldots, n'$, as follows. We let h_0 and l_0 equal 0 and n, respectively. Inductively, given i, $i > 0$, we assume integers h_{i-1}, l_{i-1}, $0 \leq h_{i-1} < l_{i-1} \leq n$, have been defined such that for integers j, k, $1 \leq j < k \leq m$, $R_{h_{i-1}} \in W_j$, $R_{l_{i-1}} \in W_k$, $R_{h_{i-1}} \neq R'_j$, $R_{l_{i-1}} \neq R'_{k-1}$, and the triangle with vertices Q_1, $R_{h_{i-1}}$, $R_{l_{i-1}}$ is a facet of a tetrahedron in $\hat{T}^*$. Then from the geometry of T^* and the last fact about the triangle with vertices Q_1, $R_{h_{i-1}}$, $R_{l_{i-1}}$, it follows that integers h_i, l_i exist, $h_{i-1} < h_i < l_i \leq l_{i-1}$, for which $R_{h_i} \in W_j$, $R_{l_i} \notin W_j$, and the tetrahedron with vertices Q_1, $R_{h_{i-1}}$, R_{h_i}, R_{l_i} belongs to $\hat{T}^*$. If R_{l_i} belongs to W_{j+1} then we let n' equal i. That for some i, $1 \leq i < n$, and some j, $1 \leq j < m$, R_{l_i} belongs to W_{j+1}, while $R_{h_{i-1}}$, R_{h_i} belong to W_j, follows from the fact that $\{h_i\}$ is an increasing sequence of integers bounded above by $\{l_i\}$ which is itself a non-increasing sequence of integers. Thus, n' is well defined. However, this is a contradiction, for it implies for some j, $1 \leq j < m$, the existence of a tetrahedron in $\hat{T}^*$ with one vertex equal to Q_1, two vertices in W_j, namely $R_{h_{n'-1}}$ and $R_{h_{n'}}$, and one vertex in $W_{j+1} \setminus \{R'_j\}$, namely $R_{l_{n'}}$. This completes the proof of the proposition.

Finally, we shed more light on the fundamental differences between planar and 3−dimensional line insertion problems by identifying those 3−dimensional problems that can be approached algorithmically as 2−dimensional problems. In order to do this we first develop some notation. For a positive integer n, let P_i, $i = 1, \ldots, n$, be distinct points in the $x - y$ plane of 3−dimensional space, and for each i, $i = 1, \ldots, n$, let x_i, y_i be the $x-$ and $y-$coordinates, respectively, of P_i. Given a triangulation T for the set of points P_i, $i = 1, \ldots, n$, and numbers z_i, $i = 1, \ldots, n$, we let T' be the collection of distinct 2−dimensional triangles in 3−dimensional space whose perpendicular projection onto the $x - y$ plane is T, and whose set of vertices equals the set of points P'_i, $i = 1, \ldots, n$, defined by setting P'_i equal to (x_i, y_i, z_i) for each i, $i = 1, \ldots, n$.

Let P_i, P'_i, x_i, y_i, z_i, $i = 1, \ldots, n$, T, T' be as above. Assume that $\overline{P_1P_2}$ is not an edge in T and that its relative interior does not contain any vertices in T. Let T^* be the collection of triangles in T that are intersected by the relative interior of $\overline{P_1P_2}$, and let $\bar{T}$ be the collection of triangles in T' whose perpendicular projection onto the $x - y$ plane is T^*. For arbitrarily large positive z we let $\hat{Q}$ represent the point $(0, 0, z)$, and $\hat{T}$ the collection of tetrahedra obtained by computing the convex hulls of $\hat{Q}$ together with each of the triangles in $\bar{T}$. In what follows, we say that $\overline{P'_1P'_2}$ *can be inserted into* $\hat{T}$ if a collection of tetrahedra $\tilde{T}$ exists such that the tetrahedra in $\tilde{T}$ have pair-wise disjoint interiors, the relative interior of $\overline{P_1'P_2'}$ is contained in the interior of the union of the tetrahedra in $\tilde{T}$, $\overline{P_1'P_2'}$ is an edge in $\tilde{T}$, $\tilde{T}$ and $\hat{T}$ have the same set of vertices, and the union of the tetrahedra in $\tilde{T}$ equals the union of the tetrahedra in $\hat{T}$. Based on these definitions, we notice that if $\overline{P_1'P_2'}$ *satisfies the prerequisite for insertion into* $\hat{T}$, i. e. its relative interior lies entirely in $\hat{T}$ and does not intersect any edges of tetrahedra in $\hat{T}$, then an attempt can be made to insert it into $\hat{T}$ with 3D_INSERT_ATTEMPT even though $\hat{T}$ is not necessarily a complete tetrahedralization for its vertices.

We assume that $\overline{P_1'P_2'}$ satisfies the prerequisite for insertion into $\hat{T}$, that INSERT_SEGMENT (without the optimization steps) has been executed for inserting $\overline{P_1P_2}$ into T, and that procedure EDGE_SWAP (step 17 of INSERT_SEGMENT) has been executed m times during the insertion. Similarly, we assume that 3D_INSERT_ATTEMPT has been executed for attempting to insert $\overline{P_1'P_2'}$ into $\hat{T}$ and that procedure TWO_THREE (steps 21 and 27 of 3D_INSERT_ATTEMPT) has been executed m' times during the attempt.

We define functions a, e from $\{1,\ldots,m\}$ into $\{(i,j) : 1 \le i < j \le n\}$ as follows: Given l, $1 \le l \le m$, we set $a(l)$ and $e(l)$ equal to (h,k) and (q,r), respectively, where h, k, q, r are integers, $1 \le h < k \le n$, $1 \le q < r \le n$, for which after the l^{th} execution of EDGE_SWAP in INSERT_SEGMENT, $\overline{P_hP_k}$ is the new edge in the triangulation and $\overline{P_qP_r}$ is the edge that has been eliminated. Correspondingly, assuming $m' > 0$, we also define functions a', e' from $\{1,\ldots,m'\}$ into $\{(i,j) : 1 \le i < j \le n\}$ as follows: Given l, $1 \le l \le m'$, we set $a'(l)$ and $e'(l)$ equal to (h,k) and (q,r), respectively, where h, k, q, r are integers, $1 \le h < k \le n$, $1 \le q < r \le n$, for which after the l^{th} execution of TWO_THREE in 3D_INSERT_ATTEMPT, $\overline{P_h'P_k'}$ is the edge that the three new tetrahedra have in common and $\overline{P_q'P_r'}$ is the edge that the two eliminated tetrahedra had in common and that does not have $\hat{Q}$ as an endpoint. Clearly, $a(m)$ equals $(1,2)$, and if 3D_INSERT_ATTEMPT is successful then $m' > 0$ and $a'(m')$ also equals $(1,2)$.

Finally, in what follows, given integers h, k, q, r, $1 \le h < k \le n$, $1 \le q < r \le n$, we say that (h,k) *crosses* (q,r) if the relative interiors of $\overline{P_hP_k}$ and $\overline{P_qP_r}$ have one and only one point in common. Assuming (h,k) crosses (q,r), we say then that (h,k) *is below* (q,r) if at the point at which $\overline{P_hP_k}$ intersects $\overline{P_qP_r}$, $\overline{P_h'P_k'}$ is lower than $\overline{P_q'P_r'}$ with respect to the $z-$axis.

We are now ready to present a proposition that identifies the conditions that the tetrahedra in $\hat{T}$ must satisfy so that $\overline{P_1'P_2'}$ satisfies the prerequisite for insertion into $\hat{T}$, and can then be inserted into $\hat{T}$ with 3D_INSERT_ATTEMPT in a manner that mimics exactly what INSERT_SEGMENT does when inserting $\overline{P_1P_2}$ into T.

Proposition 2. $\overline{P_1'P_2'}$ satisfies the prerequisite for insertion into $\hat{T}$, m equals m', and for each integer l, $l = 1,\ldots,m$, $a(l)$ equals $a'(l)$, and $e(l)$ equals $e'(l)$ so that $\overline{P_1'P_2'}$ can be inserted into $\hat{T}$ if and only if for each integer l, $l = 1,\ldots,m$, $e(l)$ is below $a(l)$.

Proof. The 'only if' part follows easily. In order to prove the 'if' part it suffices to prove that for each integer l, $l = 1,\ldots,m$, $e(l)$, which obviously crosses $(1,2)$, is below $(1,2)$. This will imply that the line segment satisfies the prerequisite for insertion in $\hat{T}$, and that $flag2$ always equals 1 in step 25 of 3D_INSERT_ATTEMPT (after the execution of procedure FACET_INTERSECT in step 24).
Let T^* be as defined above, and let T_0^* equal T^*. Inductively, for each l, $l = 1,\ldots,m$, let T_l^* be the collection of triangles in the $x-y$ plane of $3-$dimensional space which is the transformation of T_{l-1}^* after the l^{th} edge swap.
Let $\bar{T}$ be as defined above. For each l, $l = 0,\ldots,m$, let $\bar{T}_l$ be the collection of distinct $2-$dimensional triangles in $3-$dimensional space whose perpendicular projection onto the $x-y$ plane equals T_l^*, and whose set of vertices equals that of $\bar{T}$.
For each l, $l = 0,\ldots,m$, we define a real-valued function f_l with domain the union of the triangles in T^* as follows. Given a point P in a triangle in T^* we let $\hat{x}$, $\hat{y}$ be the $x-$ and $y-$coordinates, respectively, of P, and let $f_l(P)$ be the unique number for which the point defined by $(\hat{x},\hat{y},f_l(P))$ belongs to a triangle in $\bar{T}_l$.
Given an integer l, $1 \le l \le m$, let h, k, q, r, $1 \le h < k \le n$, $1 \le q < r \le n$, be those integers for which $a(l)$ equals (h,k) and $e(l)$ equals (q,r). By definition T_l^* is the transformation of T_{l-1}^* obtained by replacing the edge with endpoints P_q, P_r by the edge with endpoints P_h, P_k. Clearly, the replaced edge is shared by two triangles in T_{l-1}^* whose union is a strictly convex quadrilateral and the new edge is the alternative diagonal of this quadrilateral. These observations and the fact that $e(l)$ is below $a(l)$ imply that f_{l-1} equals f_l everywhere except in the relative interior of the aforementioned quadrilateral in which f_{l-1} is strictly less than f_l. In particular, given a point P in the relative interior of the replaced edge, it then follows that $f_{l-1}(P) < f_l(P)$. Thus, since the edge with endpoints P_1, P_2 belongs to T_m^*, given an integer l, $1 \le l \le m$, and a point P which is the intersection of the edge with endpoints P_1, P_2 and the edge replaced in T_{l-1}^* during the l^{th} edge swap, it must follow that $f_{l-1}(P) < f_l(P) \le f_m(P)$. Hence, $e(l)$ is below $(1,2)$ and the proof of the proposition is complete.

References

[1] J. Bernal, On constructing Delaunay triangulations for sets constrained by line segments, National Institute of Standards and Technology Technical Note 1252 (1988).

[2] J. Bernal, Computing Delaunay triangulations for comet-shaped polygons, National Institute of Standards and Technology Internal Report 4716 (1991).

[3] J. Bernal, Inserting line segments into triangulations and tetrahedralizations, National Institute of Standards and Technology Internal Report 5596 (1995).

[4] L. P. Chew, Constrained Delaunay triangulations, Algorihtmica 4 (1989), 97-108.

[5] L. De Floriani, B. Falcidieno, and C. Pienovi, Delaunay-based representation of surfaces defined over arbitrarily shaped domains, Computer Vision, Graphics, and Image Processing 32 (1985), 127-140.

[6] L. De Floriani and E. Puppo, Constrained Delaunay triangulation for multiresolution surface description, Proc. 9^{th} International Conference on Pattern Recognition (1988), 566-569.

[7] L. De Floriani and E. Puppo, A dynamic incremental algorithm for constrained Delaunay triangulation, Istituto per la Matematica Applicata Tech. Rep. (1988).

[8] C. L. Lawson, Transforming triangulations, Discrete Math. 3 (1972), 365-372.

[9] D. T. Lee and A. K. Lin, Generalized Delaunay triangulation for planar graphs, Discrete Comput. Geom. 1 (1986), 201-217.

[10] B. A. Lewis and J. S. Robinson, Triangulation of planar regions with applications, The Comput. J. 21 (1978), 324-332.

[11] F. P. Preparata, M. I. Shamos, Computational Geometry - An Introduction, Springer-Verlag, New York (1985).

[12] R. Seidel, Constrained Delaunay triangulation and Voronoi diagrams with obstacles, Rep. 260, IIG-TU Graz, Austria (1988), 178-191.

[13] C. A. Wang and L. Schubert, An optimal algorithm for constructing the Delaunay triangulation of a set of line segments, Proc. 3^{rd} Ann. ACM Symp. on Computational Geometry (1987), 223-232.