Representing stereo data with the Delaunay triangulation

Jean-Daniel Boissonnat, Olivier Faugeras, E. Le Bras-Mehlman

INRIA

UNITÉ DE RECHERCHE
INRIA-ROCQUENCOURT

Institut National
de Recherche
en Informatique
et en Automatique

Domaine de Voluceau
Rocquencourt
B.P.105
78153 Le Chesnay Cedex
France
Tél. (1) 39 63 55 11

Rapports de Recherche

N° 788

REPRESENTING STEREO DATA WITH THE DELAUNAY TRIANGULATION

J.D. BOISSONNAT
O.D. FAUGERAS
E. LE BRAS-MEHLMAN

FEVRIER 1988

Representing Stereo Data with the Delaunay Triangulation*

J.D. Boissonnat, O.D. Faugeras and E. Le Bras-Mehlman
INRIA
Domaine de Voluceau
Rocquencourt
BP 105
78153 Le Chesnay

Abstract

This article proposes a coherent way of interpolating 3D data obtained by stereo, for example, with a simplicial polyhedral surface. The proposed method is based on the use of the Constrained Delaunay Triangulation ; the polyhedral surface is obtained by using a simple visibility property to mark tetrahedra likely to be empty. The method is intrinsically discontinuity preserving, works for multiple viewpoints and yields both a surface representation of objects and a volume representation of free space which may be useful in Robotics. Algorithms to implement the method are described and their complexity analyzed in the worst case and average case situations where tools of probabilistic geometry are used.

Representation de données Stéréo à l'aide de la Triangulation de Delaunay

Résumé

Cet article propose une méthode d'interpolation de données tridimensionnelles obtenues par stéréo, par exemple, à l'aide d'une surface polyhédrale. La méthode proposée est basée sur l'utilisation de la Triangulation de Delaunay Contrainte; la surface polyhédrale est obtenue en utilisant un critère simple de visibilité pour marquer les tétraèdres vides. La méthode préserve intrinsèquement les discontinuités, permet de combiner simplement plusieurs vues et fournit à la fois une représentation de la surface des objets et une représentation volumique de l'espace libre, ce qui peut être utile en Robotique. Les algorithmes implémentés sont décrits et analysés en complexité dans le cas le pire et dans le cas moyen. Des outils de géométrie probabilistique sont utilisés pour ce faire.

Work partially done under Esprit Project P940

1 Introduction

In this paper we are concerned with the representation of 3D space from stereo data. We assume that this data has been obtained from several viewpoints such as in the case of a moving vehicle. The intention is to use this representation for *determining free space* and *identifying objects*. Since multiple views are available, the representation must have good properties with respect to translation and rotation and therefore be object-centered (unlike the well-known octrees [JT80] [Sam84] [CA86]). Since we want to deal both with free-space and objects, it better be both volume and surface oriented. The representation should also be usable when only sparse data is available and should converge toward the true volumes and surfaces when the sampling rate increases. In terms of its computational properties, we would like to be able to compute it efficiently and to easily merge representations obtained from different viewpoints.

To summarize all this, the properties of the representation that we believe to be important are the following :

1. Object centered so that the effect of rigid displacements can be readily assessed
2. Volume and Surface oriented
3. Work for both sparse and dense data
4. Easy to update when new data becomes available
5. Efficient to compute

We propose to use a representation based mainly on the Delaunay triangulation of the three-dimensional data points and then to mark those tetrahedra which are empty. The union of these tetrahedra represents free space and its boundary is a polyhedral representation of the surface of the obstacles. We believe that such a representation possesses all the above listed properties. The stereo data we have been using, come from an edge based algorithm ([AF87c] [AL87a] [AL87b]) that yields a wire frame description of the scene i.e a set of line segments in 3D space. Stereo matches are obtained by matching 2D polygonal approximations of edges between images in the stereo pair by a technique of Hypothesis Prediction and Testing.

In Section 2, we review some relevant properties of the Delaunay triangulation and sketch an algorithm for computing it. In Section 3, we deal with the constrained Delaunay triangulation problem in which we want the stereo segments to be Delaunay edges. In Section 4 we describe how empty tetrahedra can be marked, that is how the representation can be built. In Section 5, we describe the problem of integrating several viewpoints. In Section 6, we discuss the problem of eliminating singularities. In Section 7, we show results on a number of real and synthetic scenes, and conclude in Section 8.

2 Delaunay Triangulation

The data provided by the stereo matcher consist of line segments lying on the boundary of the objects and of triangles that link these line segments to the cameras. These triangles represent the optical rays and, thus cannot pass through any obstacle. Given such line segments and such triangles, there exist several polyhedra with the line segments as edges and not intersecting with the triangles. This situation is quite different from the analogous 2-D case : given a set of points and a set of rays issued from these points, there only exists one polygon with the points as vertices and not intersecting with the rays [ABY87]. In order to prevent a combinatorial explosion, we will a priori restrict our solution to be contained in a particular geometric structure, the Delaunay Triangulation of the end-points of the line segments. This choice is motivated by the numerous interesting properties of the Delaunay Triangulation. In particular, as shown by [Boi85b], the Delaunay triangulation of a set of points on a surface is a geometrical structure that defines neighborhoods of the points which are isotropic, and is also closely related to the metric of the surface.

2.1 Definition and properties

We present here only the definitions and properties of the Delaunay triangulation which are necessary to understand the sequel of the article, the interested reader can find more details in [Rog64], [Kle80], [PS86], and [Sib78].

Given a set $\mathcal{P}$ of n points M_i of an Euclidean space $\mathcal{E}$, the corresponding Voronoi diagram is a set $\mathcal{V}$ of n convex polyhedra V_i where V_i is the set of points which are closer to M_i in $\mathcal{E}$ than to any other point in $\mathcal{P}$:

$$V_i = \{P, P \in \mathcal{E}, \forall j, 1 \leq j \leq n, d(P, M_i) \leq d(P, M_j)\}$$

where d denotes the euclidian distance.

The straight-line dual of the Voronoi diagram, obtained by linking with line segments the points M_i whose Voronoi polyhedra are adjacent, is called the *Delaunay triangulation* (Figure 1) and has some interesting properties.

The Delaunay Triangulation is a collection of simplexes tessellating the convex hull of $\mathcal{P}$. These simplexes are either triangles if $\mathcal{E}$ is the plane or tetrahedra if $\mathcal{E}$ is the three-dimensional space. Their circumscribed sphere contain no other point of $\mathcal{P}$. Moreover, if D is the dimension of $\mathcal{E}$, the Delaunay Triangulation associates to each point M of $\mathcal{E}$ a set of at least $D+1$ neighbors which are, roughly, the nearest neighbors of M in all directions.

2.1.1 Delaunay Triangulation and polyhedral approximations

Since we are interested in surface approximations of objects, let us recall here the general definition of a triangulated polyhedron $\mathcal{T}$ in 3D space :

- Two triangles are either disjoint, have one vertex in common or have two vertices and consequently the entire edge joining them in common.

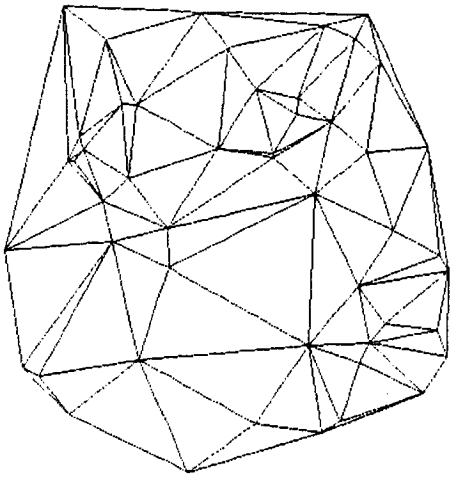

Figure 1: The Delaunay Triangulation of a set of points

- $\mathcal{T}$ is connected.
- For each vertex v of a triangle in $\mathcal{T}$, the edges opposite to v in all triangles of $\mathcal{T}$ having v as a vertex form a simple polygon.

The Delaunay triangulation satisfies the first two conditions above and is therefore a priori an interesting structure to work with.

Returning to the problem of approximating the surface of objects with a polyhedron, it would be desirable if the Delaunay triangulation contained a polyhedron respecting the relative locations of the points on the surface. More precisely, this polyhedron must be diffeomorphic to a curved polyhedron tightly stretched on the surface and passing through the points of the set $\mathcal{P}$. According to the previous definition and properties, this condition is satisfied as soon as there exist spheres passing through the vertices of each of these curved triangles and not containing any other point in their interior.

This condition does not appear to be very restrictive if the number of points on the surface is large enough. In particular, very thin parts with respect to the discretization can be allowed provided that there is enough free space around (see Figure 2, the edge AB belongs to the Delaunay triangulation of the boundary of the object). Therefore, under fairly weak assumptions, the Delaunay triangulation contains a polyhedron which suitably approximates the surface of the object.

There remains the problem of extracting the polyhedron from the triangulation. This has been covered by one of the authors [Boi84] in the general case where only the coordinates of the points in $\mathcal{P}$ are known. The basic idea for non-convex objects is to eliminate tetrahedra until all the points of $\mathcal{P}$ are on the boundary of the polyhedral shape so obtained. The elimination is done sequentially according to an ordering criterion that

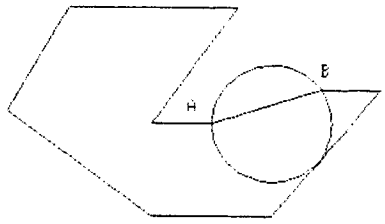

Figure 2: The edge AB belongs to the Delaunay triangulation of the boundary of the object.

depends on the application, and by following a rule that guarantees that at each step the previous boundary satisfies the definition of a polyhedron. This combinatorial rule, which is of interest in its own right, will be used in this paper (see Section 6) to eliminate possible singularities.

2.1.2 Delaunay Triangulation and Curvature

There are some nice properties of the Delaunay Triangulation which are related to the curvature of the object surface and its skeleton.

If we assume that the density of points increases indefinitely, the Delaunay circles corresponding to the Delaunay triangles inside the object become tangent to its surface and are the maximal circles covering its interior. Their centers form the interior **skeleton** ([DH73] [Boi85a] [BN78] [Nac82]); the same is true for the exterior skeleton if we remove the triangles corresponding to the inside of the object. An example is given in Figure 3.

If the number of points is not infinite, an approximation of the 2D (3D) skeleton is obtained by linking the centers of the Delaunay circles (spheres).

The skeleton tells us something about **curvature** ([BN78,Nac82]). Indeed, to each end-point of the skeleton corresponds a local maximum of the curvature. Unfortunately, the correspondence is not always one to one. Some local maxima of curvature do not yield an end-point in the skeleton. However, the following property is always true at a convex point x_i:

Proposition 1 *The radius of each Delaunay circle corresponding to a triangle inside the object whose vertex is x_i is less than or equal to the radius of curvature at this point.*

We assume that the circle S_2 tangent to the boundary of the object is not a Delaunay circle. S_2 must contain at least one other point. The proof of the proposition is equivalent to the proof of the following lemma :

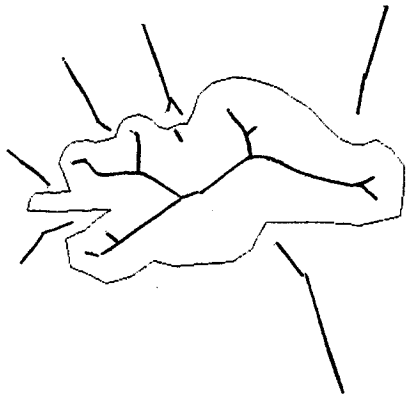

Figure 3: Exterior and interior skeletons

lemma 1 *Let S_2 be a circle, whose center is on the y-axis, and passing through the origin A. Let B and C be two points such that B belongs to the region of the plane $IR_- \times IR_+ \setminus S_2$ and $C \in IR_+ \times IR_+ \setminus S_2$. The radius of the circle circumscribed about the triangle ABC is greater or equal to the radius of S_2 (see Figure 4).*

The proof of this lemma is left to the reader.

The proof of the proposition then follows :

Let us assume that the triangle formed by x_{i-1}, x_i, and x_{i+1} is such that its radius is a good approximation of the radius of curvature at point x_i. If it is a Delaunay triangle then proposition 1 is certainly true. If it is not, then let P and Q be two points on the object such that the triangle x_iPQ is Delaunay. The circle circumscribed about this triangle does not contain any other point on the object, and in particular does not contain x_{i-1} and x_{i+1}. Applying lemma 1 shows that the radius of the Delaunay circle is less than or equal to the radius of curvature.

If the point x_i is not convex, the center of curvature is on the other side of the tangent to the object at x_i and we must compare the radius of curvature with the radii of circles corresponding to Delaunay triangles outside the object. But otherwise, Proposition 1 remains unchanged.

In the three-dimensional case, the curvature is described by the two principal curvatures λ_1 and λ_2 ([DC76]). The two radii of curvature are $\frac{1}{|\lambda_1|}$ and $\frac{1}{|\lambda_2|}$. Three cases can occur :

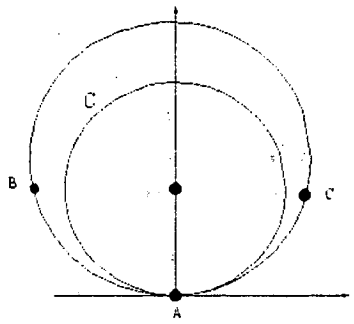

Figure 4:

1. λ_1 and λ_2 are positive (convex point).
2. λ_1 and λ_2 are negative (concave point)
3. λ_1 and λ_2 have opposite signs (saddle point).

Proposition 1 can be extended to case 1 for tetrahedra inside the object, to case 2 for tetrahedra outside the object, and to case 3 if we consider both internal and external tetrahedra. We have therefore the following properties:

Proposition 2 *At a convex (concave) point, the radius of each Delaunay sphere corresponding to an internal (external) tetrahedron whose vertex is x_i is less than or equal to the largest radius of curvature at this point.*

Proposition 3 *At a saddle point, the radius of each Delaunay sphere corresponding to a tetrahedron whose vertex is x_i is less than or equal to the largest radius of curvature at this point.*

These properties related to curvature make the Delaunay triangulation an even more appealing tool for representing shapes. In the case of stereo, we are not considering only the positions of isolated points but segments and we have some extra information : the position of the sensors. This knowledge can be used to decide whether a tetrahedron can be eliminated as free space or not, by taking into account a very simple *visibility property* to be described more fully in the Section 4.

2.2 Computation of the Delaunay Triangulation

Another advantage of the Delaunay triangulation is that it can be computed efficiently. [Kle80] placed a lower bound on the worst case complexity of any algorithm computing the Delaunay triangulation by proving that the number of tetrahedra in the Delaunay

triangulation of n points is at most $O(n^2)$ in the three dimensional case, and $2n-4$ triangles in the two dimensional case. Edelsbrunner and Seidel have shown that the Voronoi diagram and thus the Delaunay Triangulation of n points can be computed in that time boundary [ES85]. This is the worst-case optimal. However, in many typical cases, the number of tetrahedra is (fortunately) linear. It has been proved in [BT86] that in case of homogeneous distributions of points on a surface, the number of tetrahedra is no more than $8n$.

In practice, it is quite efficient to use an *incremental algorithm* whose complexity is $O(n^3)$ in the worst case but appears to work in linear time for points situated on surfaces [Bow81], [Her82], [Wat81]. The algorithm works by inserting *sequentially* the n points, while updating the current triangulation at each insertion.

2.2.1 The Algorithm

We assume that all points fall within a given cube (it is always possible to make this assumption). The vertices of the cube are added to the set of points, and the original triangulation consists of the triangulation of the cube vertices. The algorithm itself is quite simple :

Algorithm 1 :

Construct the frame and triangulate
For each point P
  Create a list of the current tetrahedra of the triangulation whose circumsphere contains P
  Erase these tetrahedra from the triangulation
  Create a list of all the faces of these tetrahedra
  If a face appears twice in the list, erase it from the list.
  For each remaining face, compute the circumsphere of the tetrahedron obtained by adding the new point
  Add the corresponding tetrahedron to the triangulation.

2.2.2 Analysis of this algorithm.

Since after inserting k points, the new point may potentially fall into all of the circumspheres and that they may be $O(k^2)$ of them, the total computing time is $\sum_{k=1}^{n} k^2 = O(n^3)$. Practically, the number of spheres in which a point falls is fairly constant, so that the algorithm runs in almost linear time.

Figure 5 represents the number of tetrahedra nt as a function of the number of points np inserted where the initial set of points corresponds to a real scene. nt is a linear function of np and for this specific example :

$$nt < 7\,np$$

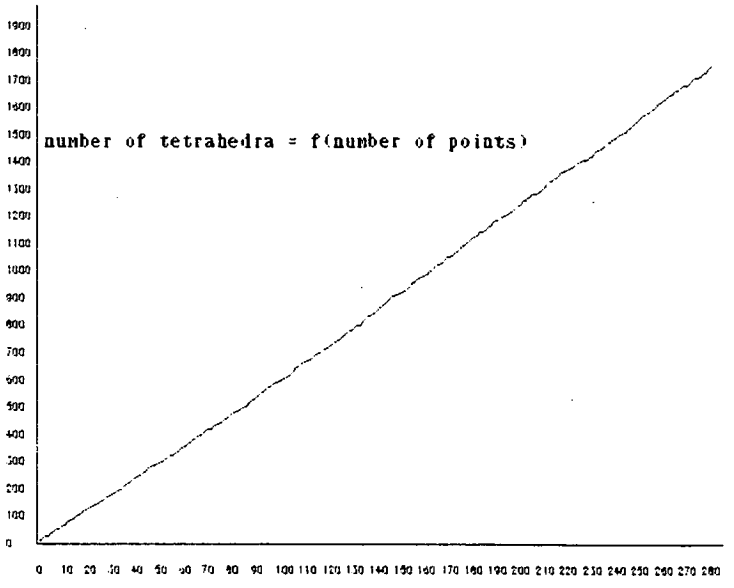

Figure 5: The number of tetrahedra is a linear function of the number of points inserted (the points correspond to a real scene)

The Delaunay triangulation therefore appears as a promising tool for achieving the objectives that we set to ourselves at the beginning of this paper.

3 Constrained Delaunay Triangulation

One problem with the Delaunay triangulation is that it is not immediatly obvious how to modify it so that it is possible to include line segments, triangles or even tetrahedra while making sure that these primitives will be part of the final triangulation. This is a problem known in Computational Geometry as the Constrained Triangulation Problem. A recent paper by Lee and Lin ([LL86]), defines a constrained Delaunay Triangulation for any planar straight line graph which can be computed in $O(n^2 \log n)$ time.

We present here an algorithm which runs in $O(n^2)$ in the worst possible two-dimensional case, and which can be generalized to any dimension ; it is very efficient in practice as shown by the complexity analysis presented later.

The basic idea is to modify the input data by adding more points, in such a way that the resulting Delaunay triangulation is guaranteed to contain, say a set of initial edges. Let S be a set of segments and s an element of S. s will be a Delaunay edge of the Delaunay Triangulation of the endpoints of the elements of S if the sphere whose diameter is s does not intersect any other segment of S.

Let S_3 be such a sphere attached to the segment s, and $s_1, s_2, \ldots, s_n$ be the segments

intersecting S_3. It is possible to split s into a finite number of subsegments such that none of the spheres attached to those segments intersects any of the s_i's. Before going further, let us recall a definition and property :

Definition 1 *The distance between two sets A and B is defined by :*

$$d(A,B) = Inf\{d(a,b), a \in A, b \in B\}.$$

Property 1 *If A and B are compact and $d(A,B) > 0$, there exist two points, $\alpha \in A$ and $\beta \in B$ such that $d(A,B) = d(\alpha,\beta)$.*

Let d be the smallest distance of the s_i's to segment s, $d = Inf\{d(s_i,s), 1 \leq i \leq n\}$.

3.1 Isolated segments

If there is no segment connected to s, the distance d is strictly positive. Let D be the length of s. We may divide s into $\lceil D/2d \rceil$ intervals of length less than or equal to $2d$. This corresponds to adding $\lceil D/2d \rceil - 1$ points, the endpoints of these segments to segment s. Therefore, as shown in Figure 6, the discs having those segments as diameter do not contain any points on other segments. In this figure, points A_1, A_2 and A_3 have to be added to the original set of points.

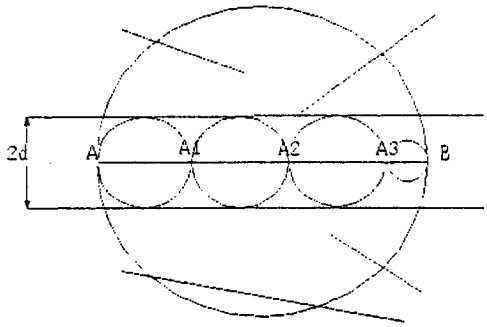

Figure 6: The segment AB is not connected to any other segment and is cut in four subsegments AA_1, A_1A_2, A_2A_3 and A_3B

3.2 Connected segments

In the case where AB is connected to one segment BC, we consider the plane perpendicular to AB in B. If this plane separates AB from BC, then the previous technique for splitting AB can be used (see Figure 7).

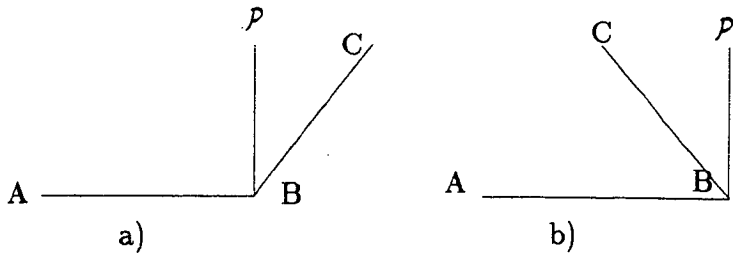

Figure 7: The plane P perpendicular to AB in B a) separates AB and CD b) does not separates AB from BC

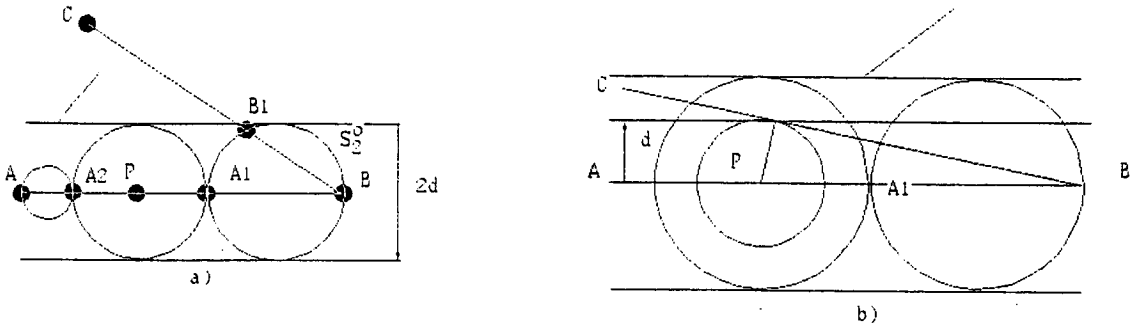

Figure 8: a) $d = \delta_1$ b) $d = \delta_2$

When they are on the same side, if there are some segments intersecting the circle S_2 whose diameter is AB, we look for the smallest distance δ_1 of the s_i's to segment s ; δ_1 is strictly positive.

There must be only one circle which intersects BC. Let P be the point on segment AB with $\|PB\| = 3\delta_1$, δ_2 the distance of P to BC and $d = \min(\delta_1, \delta_2)$. If D is the length of AB, we may divide AB into $\lceil D/2d \rceil$ intervals of length less than or equal to $2d$. The discs having those segments as diameter do not intersect any other segment except one S_2°. Its diameter is A_1B ($\|A_1B\| = 2d$). By adding to BC the intersection B_1 of S_2° and BC, we guarantee A_1B and BB_1 to be Delaunay edges (see Figure 3.2).

After this preprocessing, the resulting augmented set of line segments satisfies the condition that they are Delaunay edges of the Delaunay triangulation of their endpoints. This preprocessing has been described in the two-dimensional case ; it can be extended to any dimension in a straightforward manner.

Proposition 4 *The number of points, we add in this process is in the worst case* $O(n)$.

Each segment s_i in the original set has a distance $d_i > 0$ attached to it and computed as described above. If we define $d = \min_{1 \le i \le n} \{d_i\}$, $d > 0$. And for any segment AB of length

D in the band parallel to AB of width $2d$ there are no other points belonging to other segments (see Figure 6). We then only need to add $\lceil D/2d \rceil - 1$ points to segment AB so that the corresponding segments thus created satisfy the constraint.

But we may outline the fact that $\lim_{d \to 0} \lceil D/2d \rceil = +\infty$.

3.3 Questions of thresholds

In practice depending on the kind of preprocessing, two segments may not be exactly connected. Therefore, d can be very small. To distinguish the two cases, AB and CD are connected or not, we use a threshold ε depending on the resolution ; if two segments are parallel or close to parallel and if the parallel distance is δ, ε must be less than or equal to δ.

Consequently, if the distance between AB and CD is less than or equal to ε, two situations may occur :

- AB and CD are connected. The distance between two endpoints is less than or equal to ε.

- The value of the distance $d(AB, CD)$ is reached for two points $\alpha \in AB$ and $\beta \in CD$ which are not the endpoints of the segments AB and CD.

If we are in the plane and because two segments must not intersect, α or β is an endpoint (say α). We may then divide CD into two subsegments $C\beta$, βD and apply the process described above to the two new subsegments.

If we are not in the plane, we may divide the two segments AB and CD into two subsegments $A\alpha$, αB, and $C\beta$, βD.

The number of points, we add, $\lceil D/2d \rceil$ is consequently greater than or equal to $\lceil D/2\varepsilon \rceil$ and depends on the value of the threshold.

We now present the algorithm, based on the above ideas, that computes a Delaunay Triangulation constrained to contain a set of segments S :

Algorithm 2 :

For any edge, say AB, in S do :
1 If AB is connected to one or two segments, then preprocess it the way it is described in Section 3.2.
2 Find $i \in [1, n]$ such that the distance d_i between e_i and AB is minimal ;
1 Add the corresponding new points (as described above) in a list of new points ;
Compute the Delaunay Triangulation of the new set of points.

3.4 Complexity Analysis

The complexity of this algorithm is the sum of two contributions:

1. that of computing the Delaunay triangulation of $O(n)$ points and we have seen in Section 2.2 that it is worst case $O(n^2)$ in the planar case and $O(n^3)$ in the three-dimensional case, if we use the sequential algorithm for the Delaunay triangulation. In practice, we have seen that this algorithm is in fact linear (see also [BT86]).

2. that of steps 1 and 2 of the previous algorithm which are a priori $O(n^2)$.

Therefore these steps may dominate the overall complexity if some care is not taken. In order to reduce its complexity, we use *bucketing techniques.* Bucketing techniques in Computational Geometry have been studied when the data are points. Edahiro, Kokubo and Asano proved the efficiency of such techniques for the Point-Location Algorithm ([EKA84]) ; Asano, Edahiro, Imai and Iri analyzed the complexity of the problems of minimum weight perfect matching in the plane, the two-dimensional Voronoi diagrams, the range search in the plane and the shortest paths in networks ([AEII85]), but did not analyze the problems related to segments. The idea of using buckets is clearly motivated by the fact that we do not need to compare for each edge AB, the distance between AB and every other segment. The segments which must be considered are those intersecting the sphere whose diameter is the segment AB.

The bucketing technique used here consists in dividing the space into cubic or parallelepipedic buckets and in computing for each sphere the buckets which cover it and for each segment the buckets it intersects.

We then compute the distances between the segment AB and each segment which may intersect the sphere of radius AB, i.e. the segments which have at least a bucket in common with the sphere.

The worst case is $O(n^2)$ but let us look at the average case which is in practice, more interesting. We assume that the line segments can be considered as being produced by a Poisson Point process (see the Appendix) of parameter θ, the density of points. Most of the results which follow are true in theory only for an unbounded image which means that basically we are ignoring boundary effects. This is also the case for the results of Section 4.3. After this caveat, we have the following proposition :

Proposition 5 *Let δ be a stationary Poisson distribution of points in the plane whose parameter θ is the density of points. If $\bar{l}$ is the average length of the segments, the average number $\overline{ns}$ of segments intersecting a given circle, whose radius is r is given by the following formula :*

$$\overline{ns} = \theta r(2\,\bar{l} + \pi r)$$

We have a similar proposition in the three-dimensional case :

Proposition 6 *Let δ be a stationary Poisson distribution of points in space whose parameter θ is the density of points. If $\bar{l}$ is the average length of the segments, the average number $\overline{ns}$ of segments intersecting a given sphere, whose radius is r is given by the following formula :*

$$\overline{ns} = \theta \pi r(2\,\bar{l} + \frac{4}{3}r^2)$$

Therefore, if ns is the number of segments, the average number of comparisons that need to be made in steps 1 and 2 is :

$$ns \times 2\theta\bar{l}^2(1 + \frac{2\pi}{3})$$

in the two-dimensional case and

$$ns \times 2\pi\theta\bar{l}^2(1 + \frac{4}{3}\bar{l})$$

in the three-dimensional case ; this means that *the average number of comparisons that need to be made is* $\theta(n)$.

The careful reader will have noticed that since the density θ is estimated as n/L^2 (n/L^3) in the two-dimensional case (the three-dimensional case), where L is the size of the image (working volume), the complexity is in fact still $O(n^2)$. But our estimation of the constant is now accurate. Also, in practice, n is less than 1000 and L is 512 pixels which means that θ is of the order of .4%, quite a small number compared to n. This remark also applies to the other analysis of algorithm complexity which are presented later in the paper.

We have applied this algorithm to a set of segments coming from a real image. One result is shown in Figure 9.

We now proceed to describe how we mark the empty tetrahedra by using the visibility property.

4 Marking Empty Tetrahedra

Figure 10 shows a two-dimensional example of the basic principle of the method. Point P represents one of the cameras optical center, points A, B, C, D, M, E, F, and G are points in the set S, measured by the stereo process. We have computed the Delaunay triangulation of the points in S and we are considering point M. This point is by definition visible from P and therefore all triangles in the Delaunay triangulation intersected by the segment PM should be marked as empty space. These are the textured triangles in Figure 10. By considering more points and the second camera center, more triangles can be marked as empty space. If we find that it is a little hazardous to mark a triangle as empty if it crossed only by one ray, we can keep for each triangle a count of the number of times it has been crossed by a ray. Only triangles which have been crossed by a sufficient number of rays (where sufficient depends on the quality of the data and on the application) are marked as empty.

In the three-dimensional space, things are very similar except for the fact that we have to deal with line segments in 3D rather than points. We assume therefore that the Delaunay triangulation is built for the set S of the segments endpoints and each segment is a Delaunay edge.

As shown in Figure 11, for a given stereo segment AB, the visibility property means that all tetrahedra intersected by triangle PAB can be marked as *empty* space, where P is one of the camera optical center. We call such a triangle a *stereo triangle*. For example, the

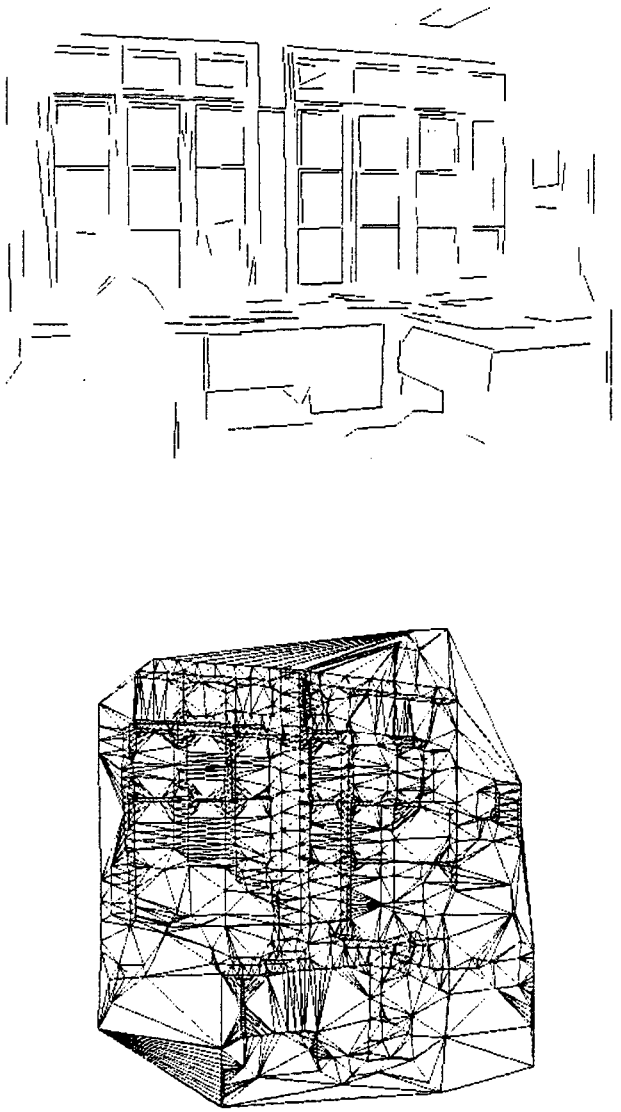

Figure 9: A set of line segments of a real scene and the 2D Constrained Delaunay Triangulation on this set of segments

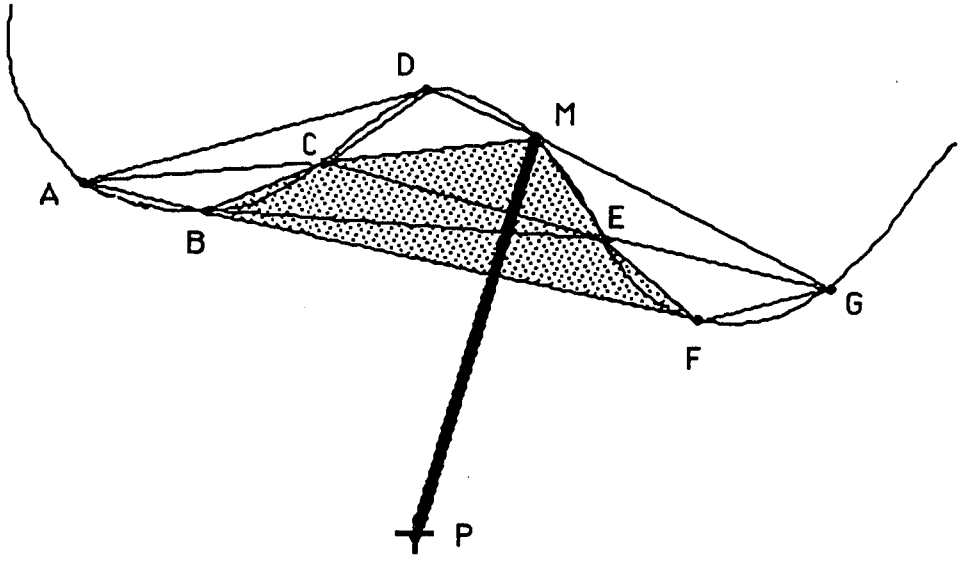

Figure 10: Basic principle of using the visibility property to mark empty tetrahedra

stereo triangle PAB intersects tetrahedron $QRST$, the intersection being triangle HIJ. Therefore, tetrahedron $QRST$ can be marked as empty space.

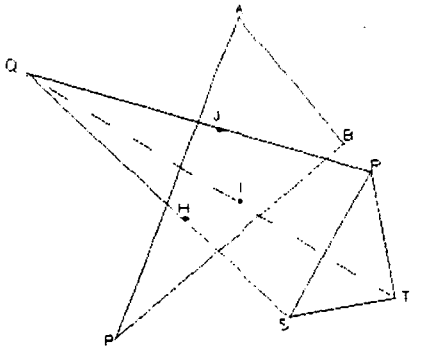

Figure 11: The stereo triangle PAB intersects tetrahedron $QRST$ which can then be marked as empty

A remark similar to the previous one on the number of triangle-ray intersections can be made on the number of tetrahedron-triangle intersections.

The next question then is, given the Delaunay triangulation of the endpoints of the stereo segments, how can we efficiently detect those tetrahedra which are intersected by at least one stereo triangle PAB.

The obvious way of going is by performing the following algorithm :

Algorithm 3 :

For all stereo segments AB
  For all unmarked tetrahedra $\mathcal{T}$
    Determine if the intersection of triangle PAB with tetrahedron $\mathcal{T}$ is not empty.
    If it is not empty then mark tetrahedron $\mathcal{T}$

Clearly, the computation time of this algorithm depends heavily on the number of intersections which are going to be computed. There are several ways of minimizing this number and all imply some preprocessing of the data. One obvious idea is to use bucketing techniques. These can be used either in 3D or in 2D by projecting the tetrahedra and the segments in the image, as shown next.

4.1 Bucketing in 3D

The idea is to divide the whole space (assumed to be included in a parallelepiped) into cubic or parallelepipedic buckets and to compute for each stereo triangle PAB the list of buckets it intersects and for each bucket the list of tetrahedra $\mathcal{T}$ of the Delaunay triangulation that it intersects. This preprocessing having been performed, the previous algorithm can be made slightly more precise :

For each stereo segment AB
  Get the list of buckets associated with the stereo triangle PAB
  For each bucket $\mathcal{B}$ in that list
    Get its list of associated tetrahedra
    For each unmarked tetrahedron $\mathcal{T}$ in that list
      Determine if the intersection of triangle PAB with tetrahedron $\mathcal{T}$ is not empty
      If it is not empty then mark tetrahedron $\mathcal{T}$.

Clearly, we may be able to save time if the tetrahedra are not too big with respect to the buckets size by performing less intersection tests in the innermost loop of this algorithm. The disadvantage of this approach is that the preprocessing time of computing the buckets and their associated structures may be long compared to the savings. This is why we propose a second alternative which uses a two-dimensional structure which has previously been computed by the stereovision matcher.

4.2 Bucketing in 2D

The basic principle is shown in Figure 12.

In this figure, face QRS of the tetrahedron $\mathcal{T}$ projects itself as triangle qrs on the retina of the camera with optical center P. In the same way, stereo segment AB projects itself as segment ab. The following propositions are self-evident:

Proposition 7 *If for all faces $F_i, i = 1, \ldots, 4$ of tetrahedron $\mathcal{T}$, their perspective projections f_i in retina plane R do not intersect line segment ab, then tetrahedron $\mathcal{T}$ does not intersect triangle PAB.*

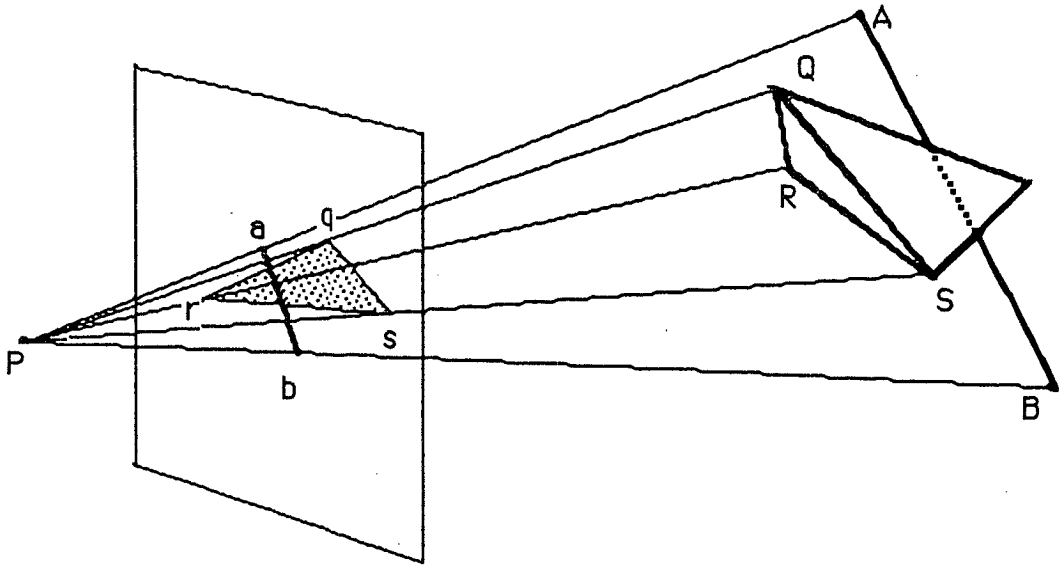

Figure 12: Testing the 3D intersection between a stereo triangle and a tetrahedron can most of the time be performed in the image plane

Proposition 8 *If the intersection between tetrahedron $\mathcal{T}$ and triangle PAB is not empty, then a face F_i intersects PAB iff its projection f_i on retina plane R intersects segment ab.*

Proposition 9 *Reciprocally, if there exists an i such that the projection f_i of face F_i of tetrahedron $\mathcal{T}$ intersects segment ab, then there may exist an intersection between tetrahedron $\mathcal{T}$ and triangle PAB.*

It is in fact possible to be slightly more precise with very little extra computation in the case described by Proposition 9. Let us consider an axis Ox and let us project A, B, and P on that axis on three points with abscissa x_a, x_b, and x_p. Let us denote by min (resp. max) the smallest (resp. the largest) of the quantities $\mid x_a - x_p \mid$ and $\mid x_b - x_p \mid$. Projecting the vertices of $\mathcal{T}$ on Ox yields four points of abscissa x_i, where $1 \leq i \leq 4$. Let us denote by Min and Max the minimum and maximum value of $\mid x_i - x_p \mid$'s, respectively. The situation is shown in Figure 13.

The following propositions are self-evident :

Proposition 10 *If $Min > max$ then tetrahedron $\mathcal{T}$ does not intersect triangle PAB*

Proposition 11 *If $Max < min$ then tetrahedron $\mathcal{T}$ intersects triangle PAB.*

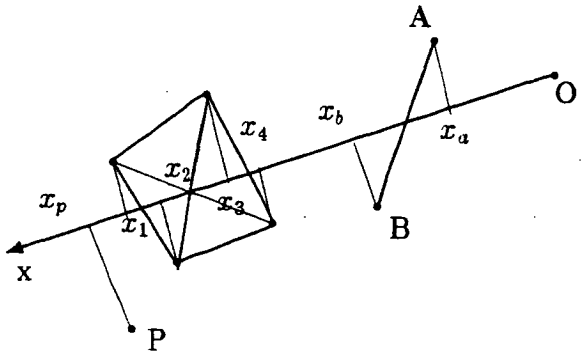

Figure 13: Projecting points on an axis can often tell wether the stereo triangle and tetrahedron intersection exists.

In the stereo matching algorithm described in [AF87c], [AL87a] and [AL87b], square buckets are used in the image planes to get fast access to the line segments neighbors. Therefore we already have, attached to each projection of a stereo segment in the image planes, a list of buckets intersected by this projection. The only remaining computation that needs to be done is to attach to each bucket the list of triangles it intersects (a triangle is the perspective projection in the retina of the face of a tetrahedron in the Delaunay triangulation).

All this allows us to write another version of our algorithm as follows.

Algorithm 4 :

For each stereo segment AB, let ab be its projection in the retina
  Get the list of buckets associated with ab
  For each bucket $\mathcal{B}$ in that list
    Get its list of associated triangles
    For each unmarked triangle t in that list
      Determine if ab intersects t (test 2D intersection)
      If it does, then
        if $Max < min$
        then mark tetrahedron $\mathcal{T}$ corresponding to t
      else
        if $Min \leq max$ then
          if triangle PAB intersects tetrahedron $\mathcal{T}$
          then mark $\mathcal{T}$ (test 3D intersection)

Discussion

All this is very fine but the question is, do we save computation time by doing this. A formal answer is hard to give. What can be said is that the 2D segment/triangle test is a lot cheaper than the 3D triangle/tetrahedron test since most of it can be easily done by look-up table and that the tests suggested by propositions 10 and 11 considerably reduce the number of full 3D tests which are actually performed on the examples that we have treated so far.

We now propose an analysis of the algorithm complexity.

4.3 Analysis of the algorithm complexity

We will first analyze the worst case and then the average case. Let ns be the number of segments and nt be the number of tetrahedra.

4.3.1 Worst case Analysis

If there is only one bucket, the study of this algorithm is equivalent to the study of the first algorithm, without buckets. For each segment s, we compute its intersection with every unmarked tetrahedron $\mathcal{T}$ whereas the introduction of buckets leads us to compute its intersection with only the tetrahedra intersecting the buckets intersected by its corresponding stereo triangle. So, we will study the case where there is only one bucket. The worst case happens in some special configurations of segments : the segments except one are all in the plane, perpendicular to the viewing direction and the remainder is behind the other ones. Moreover, the segments are sorted in such a way that the last considered is the farthest from the position of the camera. Each tetrahedron of the scene connects an endpoint of the last segment with an endpoint of a segment of the plane. Therefore, for the $(ns-1)$ st segments we have to consider every tetrahedron (none has been marked as empty), and at the ns th step of the algorithm, some tetrahedra will be marked but we have to consider all of them. The complexity of this case is $nt \times ns$.

Boissonnat and Teillaud proved in [BT86] that in case of homogeneous distributions of points on a surface, the number of tetrahedra is no more than $8n$. Therefore, the worst case analysis leads us to a $O(n^2)$ algorithm, if n is the number of points.

4.3.2 Average case Analysis

Let ns be the number of segments, $\overline{nb}$ be the average number of buckets intersected by a segment and $\overline{nt}$ the average number of triangles intersecting a bucket. The complexity of the part of the algorithm that tests the 2D intersections between a segment and a triangle is $ns \times \overline{nb} \times \overline{nt}$.

Therefore we must compute $\overline{nb}$ and $\overline{nt}$. These values depend of course on the distribution we give to the segments, and for this analysis, as in Section 3.4, we use an homogeneous distribution of points in the image.

The average number of buckets intersected by a segment :

Let I be the image. The number nb of buckets intersected by a segment s depends on its length l, the number b^2 of buckets and the dimensions of I. We may consider without loss of generality that I is square: results in the general case will be similar up to a scale factor. Let L be its size. The number of buckets intersected by s lies in the interval

$[\frac{b \times l}{L}, 2\ \frac{b \times l}{L}]$. Therefore, we have :

$$\frac{\bar{l}}{\lambda} \leq \overline{nb} \leq 2\ \frac{\bar{l}}{\lambda}$$

Where $\bar{x}$ is the average of the quantity x and λ the width of the bucket $\lambda = L/b$

The average number of segments which intersect a given bucket :

We still assume that our segments are produced by a Poisson point process (see Section 3.4 and Appendix) of parameter $\theta = \frac{n}{L^2}$.

Proposition 12 *The average number $\overline{ns}$ of segments intersecting a bucket $\mathcal{B}$ is given by :*

$$\overline{ns} = \theta\ \lambda \left(\bar{l}\ \sqrt{2} + \lambda\right)$$

This result is included here, even though it is not used in the rest of this article, because its proof, given in the Appendix, helps understanding the next result.

The average number of triangles intersecting a given bucket : Let T be a triangle. We have the following proposition:

Proposition 13 *Let $\bar{x}$ and $\bar{y}$ be the average dimensions of the minimal enclosing rectangle whose sides are parallel to the coordinates axes for the triangles we consider.*

The average number $\overline{nt}$ of triangles intersecting a bucket is given by :

$$\overline{nt} = \theta(\overline{\mathcal{A}} + \lambda\ (\bar{x} + \bar{y}) + \lambda^2).$$

Where $\overline{\mathcal{A}}$ is the average surface of the triangles and θ is the parameter of the Poisson distribution of triangles.

The proof is given in the Appendix.

Consequently, if C is the complexity of the part of the algorithm concerned with the 2D test, and $c = ns \times \frac{\bar{l}}{\lambda} \times \theta\ (\overline{\mathcal{A}} + \lambda(\bar{x} + \bar{y}) + \lambda^2)$, we have:

$$c \leq C \leq 2c$$

The main result is that C is $\theta(n)$ (see the comment in Section 3.4). If we neglect the part of the algorithm concerned with the 3D test, the average complexity is linear in the number of segments. In practice, about 50% of the marked tetrahedra are marked using the 2D intersection test and about 25% of the tetrahedra which fail to be marked fail thanks to the 2D intersection test.

5 Merging different views

We are now dealing with several sets of data points, each of them corresponding to a different viewpoint. Figure 14 shows a two-dimensional example of the method's principle, based on the work carried out on only one view. Points P_1 and P_2 represent the two camera optical centers, whereas points A, B, C, D, E, F, G and H are points in the set S, measured by the stereo process. Points A, B, C and D are seen by P_1, when points D, E, F, G, H are seen by P_2. We have computed the Delaunay Triangulation of the points in S with P_1 and P_2 and we consider points B and G. The triangles FGH and ABC have to be removed because the point B is visible from P_1 and G from P_2.

We assume that the environment is static and that only the stereo rig is moving and taking different snapshots. A system that relates the different 3D snapshots, computes the rigid displacements between any two of them, and fuses the segments which have been seen from several viewpoints has been developed by Ayache, Faugeras, and Faverjon ([FAF86,AF87a,AF87b]). For each pair of 3D snapshots, this system produces the best estimate of the rigid displacement between them, a list of segments matched, and the fused segments. This allows us to put all segments and camera centers in the same coordinate system. In practice, the 3D snapshots arrive sequentially in time while the stereo rig is moving. The fact that we are using a sequential algorithm for computing the Delaunay transformation could play an essential role since, in principle, it could be possible to start computing the constrained Delaunay triangulation after the first 3D snapshot has been obtained without waiting for the last one. We come back to this point in the conclusion.

In the currently implemented version, we do wait until all 3D snapshots have been merged by the system described in [FAF86,AF87a,AF87b]. We compute the constrained Delaunay triangulation of the resulting set of segments and the camera centers. Each segment is labelled by a list corresponding to the cameras seeing it. For a given stereo segment AB visible from the cameras $P_1, \ldots, P_p$, the **visibility property** means that all tetrahedra intersected by one of the triangles P_iAB $(\forall i, 1 \leq i \leq p)$ can be marked as *empty space*. In the previous example, point D must be considered twice, once for the first position of the camera and once for the second position.

We now present the algorithm, based on the above ideas, marking empty tetrahedra when the set of segments has been obtained from several viewpoints.

Algorithm 5 :

For all stereo segment AB
  Get the list of cameras seeing AB
  For each camera i in that list, let $(ab)_i$ be its projection in the retina i
  Get the list of buckets associated with $(ab)_i$
  For each bucket $\mathcal{B}$ in that list
    Get its list of associated triangles
    For each unmarked triangle t in that list
      Determine if $(ab)_i$ intersects t (test 2D intersection)
      If it does, then

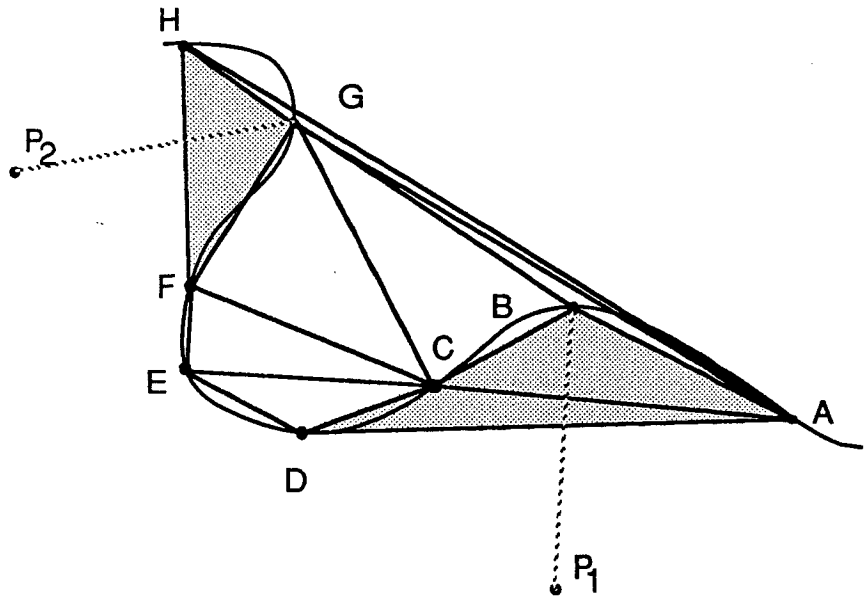

Figure 14: Because several views are available, triangles not removed, when only one viewpoint was considered, may now be removed.

```
    if Max < min
    then mark tetrahedron T corresponding to t
  Else
    if Min ≤ max then
      if triangle P_i AB intersects tetrahedron T
      then mark T (test 3D intersection)
```

The complexity of this algorithm is the same as that of the one considering only one viewpoint. It is linear in the number of segments (modulo the remark of Section 3.4).

6 Producing Valid Representations of the Obstacles

At this stage, we have marked a number of tetrahedra from the Delaunay Triangulation that are known to be part of free space and not of obstacles. However this is necessary but not sufficient, in general, to recover a valid representation of the obstacles. Indeed some tetrahedra of free space may not be intersected by any triangle joining a camera center to a measured line segment. In particular, some of the tetrahedra that remain as part of obstacles, at the end of the above procedure, can create singularities on the surface of the obstacles: i.e. the surface of the obstacles is not a polyhedron as defined in section 2. These singularities do not exist on the actual obstacles ; thus, by eliminating

the tetrahedra creating the singularities, we eliminate tetrahedra of free space and get a valid representation of the obstacles.

The singularities can be eliminated by some kind of combinatorial region growing, we called Modelling in a previous paper [Boi85b]. The principle is the following : suppose that O is a set of tetrahedra connected face to face in such a way that the boundary of O is a polyhedron as defined in section 2. We grow O by adding tetrahedra, one at a time, so that at each step the boundary of O remains a polyhedron. This is possible iff, at each step, O and the tetrahedron we add have exactly one, two or three faces in common. Once the tetrahedra of the triangulation have been marked by the visibility test, steps 1, 2 and 3 of the following algorithm separate the non-marked tetrahedra into different subsets without singularity, using the combinatorial region growing. Though this procedure prevents the creation of singularities, it also prevents the generation of obstacles with holes. This is why a second region growing (step 4) is performed on the subsets found by the first one.

Algorithm 6 :

1. Pick up a non-marked non yet considered tetrahedron T ;
2. Create a new subset $O := T$;
3. Until this is no longer possible
   3.1. Grow O by adding a tetrahedron T' of the triangulation which shares exactly one, or two or three faces with O;
   3.2. Mark T' as considered;
4. If there remain non marked not yet considered tetrahedra then goto 1;
   Else
   Merge any two subsets whose common intersection consists of (possibly several disjoint) connected sets of triangles.

This algorithm can easily be implemented in time proportional to the number of tetrahedra non-marked by the visibility test.

The result of this algorithm is a number of subsets which either belong to the obstacles or to free space. However it is not possible, from a theoretical point of view, to decide which is which. The only thing we know is that two subsets that have common vertices cannot belong either to the obstacles or to free space. In order to decide what subsets represent the obstacles and what subsets belong to free space, we need to perform new measurements or to use heuristics : for example a subset with a large number of vertices is most probably an obstacle.

7 Results

We have applied the previous algorithms to a number of synthetic and real scenes.

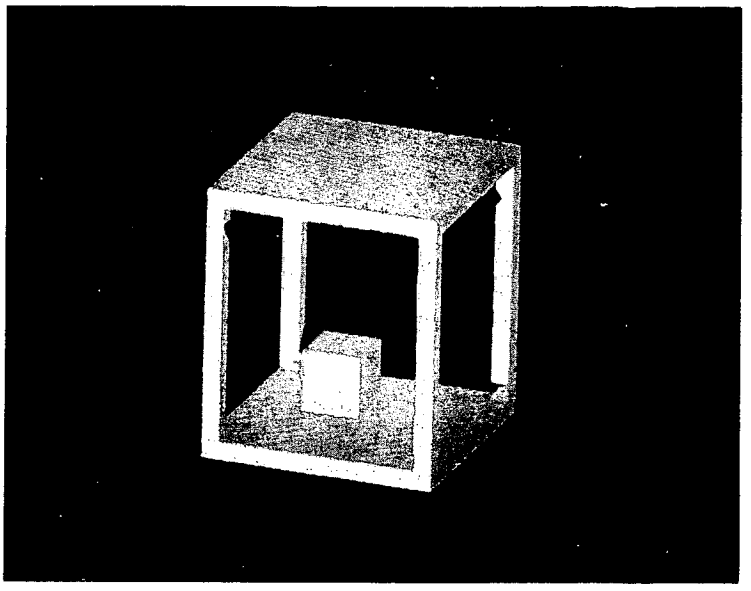

Figure 15: The result obtained by combining four viewpoints on a synthetic scene

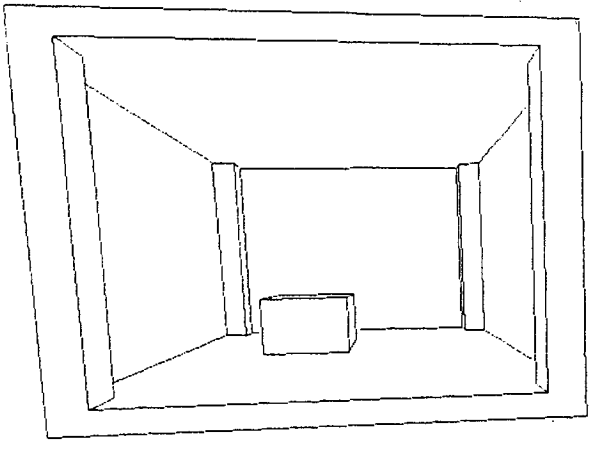

Figure 16: The synthetic scene as seen from on camera

7.1 Synthetic scenes

We show one example of such a scene in figure 15, which is in fact the result of our processing. The scene is made of a cube inside another one. The big cube has a floor, a ceiling, and four pillars. We have simulated four viewpoints such as the one shown in figure 16. The initial scene is made of 80 points, after splitting the segments so that they are all Delaunay edges, it is made of 1166 points. Figure 18 shows the result after taking into account one viewpoint only and it is to be compared with figure 15 which is perfect in this case where no noise is present. As another way of displaying what is going on, we show in figure 17 a cross-section of the Delaunay triangulation before and after marking the empty tetrahedra when using only one viewpoint, and in figure 19 the same thing when using the four viewpoints.

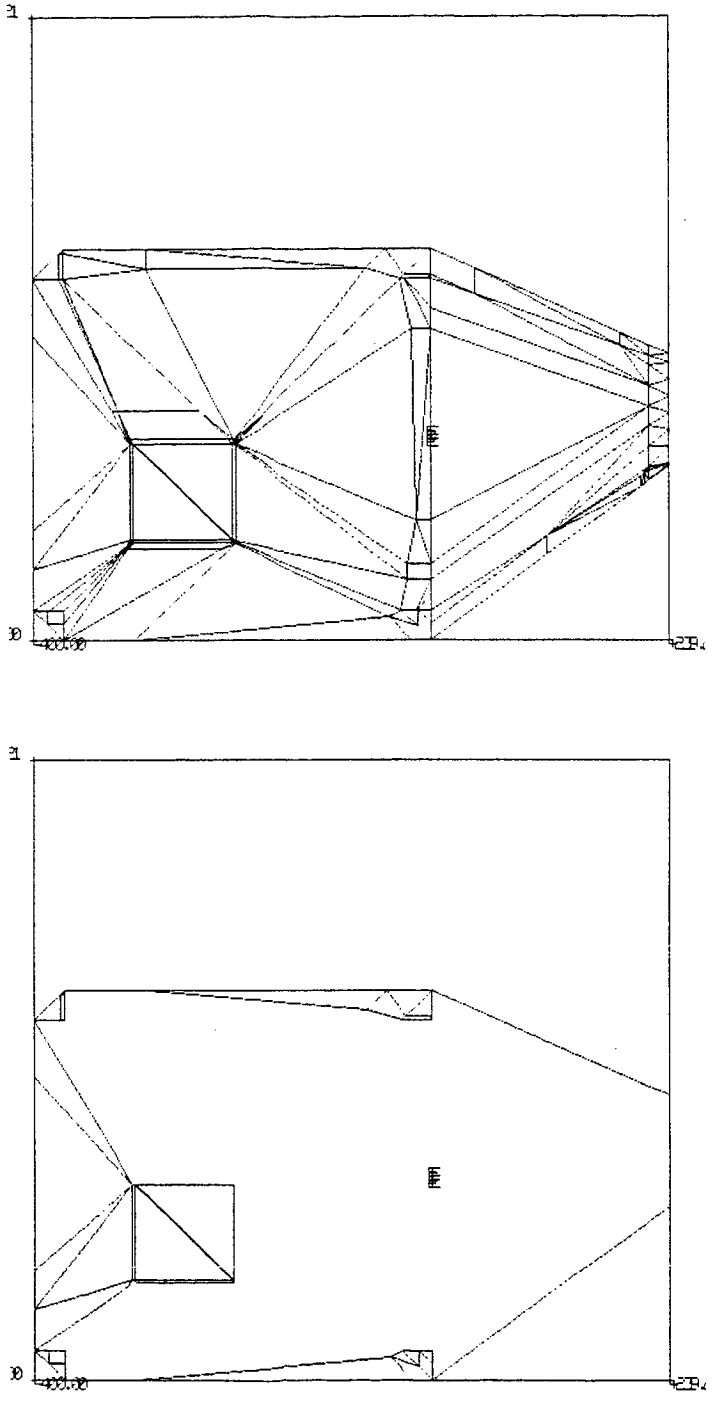

Figure 17: A cross-section of the Delaunay triangulation with a plane parallel to the floor before and after the removal of empty tetrahedra (using one viewpoint).

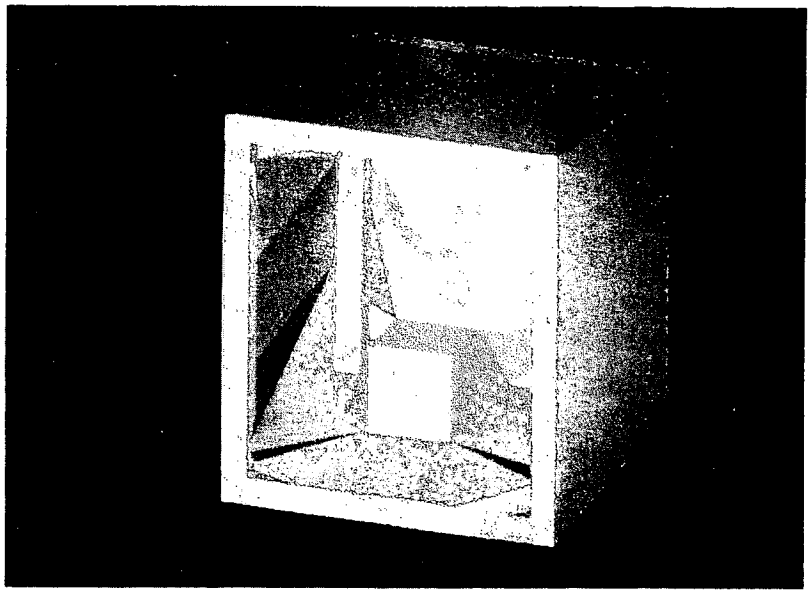

Figure 18: Using the visibility criterion for only one camera

7.2 Real scenes

We have applied these ideas to a number of outputs of the stereo matcher described in [AF87c], and processed by the system described in [FAF86,AF87a,AF87b]. We present some of them when taking one viewpoint into account and when taking several viewpoints into account.

7.2.1 Considering one viewpoint

The first original image is shown in Figure 20. The typical result of the stereo matcher of this scene is shown in Figure 21 where the line segments present in the triplet images are those which have been matched. In order to display the results of the visibility algorithm, we show cross-sections of the Delaunay triangulation before and after the visibility algorithm has been applied.

In Figure 22 we show such a cross-section with a plane parallel to the floor at about the table height before the visibility algorithm has been applied and after the removal of empty tetrahedra. The table is clearly visible.

In order to give a more expressive representation, we have projected each visible face of the unmarked tetrahedra on the image plane, computed the average grey level of each projected triangle on the original image, and created a new image where each visible triangle is drawn and filled with the grey level we obtained. Such a result is given in Figure 23 which should be compared with Figure 20.

Similar results are presented for the image of Figure 24. The salient features of this scene are a table in the foreground and, on the right hand side, or hallway extending to about 20 meters from the camera. In the middle of the hallway, a door is half open. On the wall behind the table there is a fire extinguisher. Figure 25 shows the segments which have been matched by stereo, figure 26 a cross-section of the Delaunay Triangulation before and after eliminating empty tetrahedra. The plane is parallel to the floor and approximately at the height of the table which is again clearly visible as is the door halfway down the

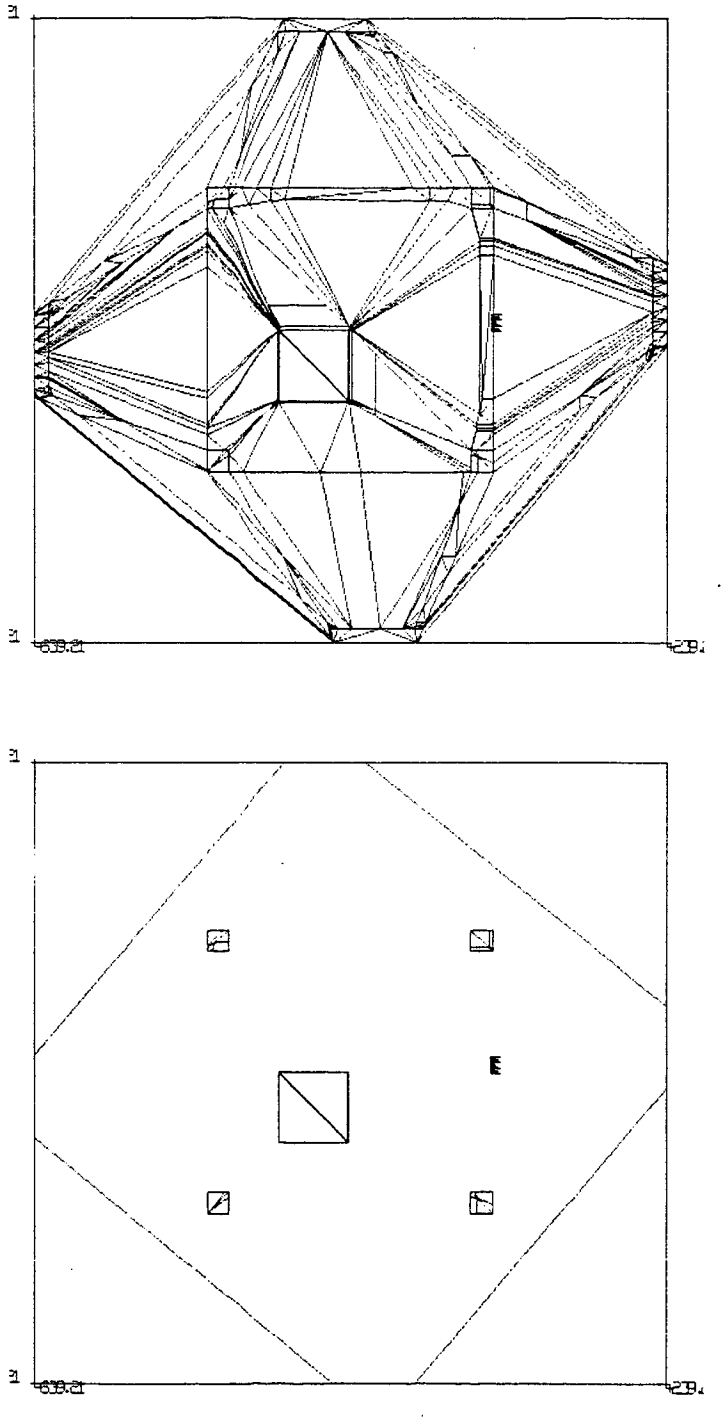

Figure 19: A cross-section of the Delaunay triangulation with a plane parallel to the floor before and after the removal of empty tetrahedra (using four viewpoints).

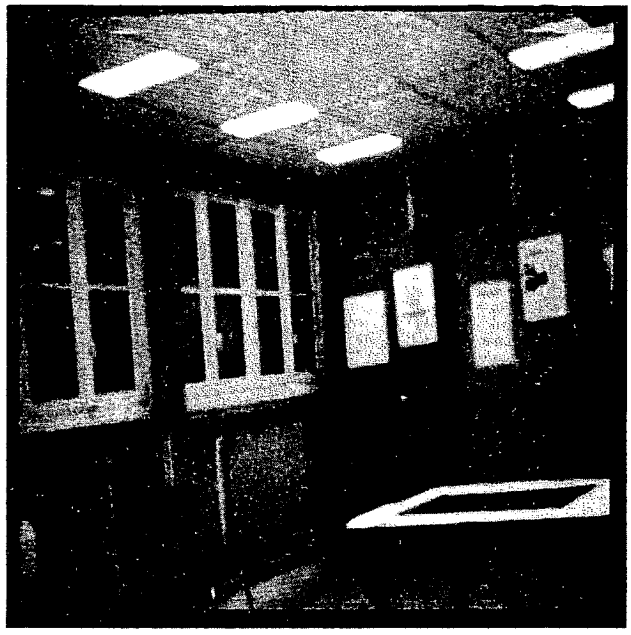

Figure 20: The original image

hallway ! (the camera is on the right handside of the figure). Figure 27 shows a shaded graphics representation of this scene.

7.2.2 Considering several viewpoints

We consider a second image taken from another viewpoint of the scene shown in figure 20 (figure 28), with the three sets of segments in Figure 29. The difference is a rotation of five degrees of the cameras with respect to a vertical axis. We also show in figure 30 a cross-section of the Delaunay triangulation of the combination of the two views before and after marking the empty tetrahedra. As shown by comparing with figure 22, we have a somewhat better definition of the scene. We have also built a "synthetic" image from the new polyhedral representation obtained in combining the two viewpoints as explained previously. This is shown in figure 31. Since we have a 3D model of the scene, we have displayed two slightly different viewpoints. Comparing this with figure 23 shows an improvement in the definition of the windows, the chair and the table, as well as of the details on the back wall.

Results are presented for three viewpoints by combining figures 24 and 32. The two viewpoints differ from that of Figure 24 by two rotations of plus and minus five degrees with respect to a vertical axis. Figure 33 shows the sets of line segments for these two images, figure 34 shows a cross-section of the Delaunay triangulation of the combination of the three views before and after marking the empty tetrahedra. Figure 35 shows the corresponding

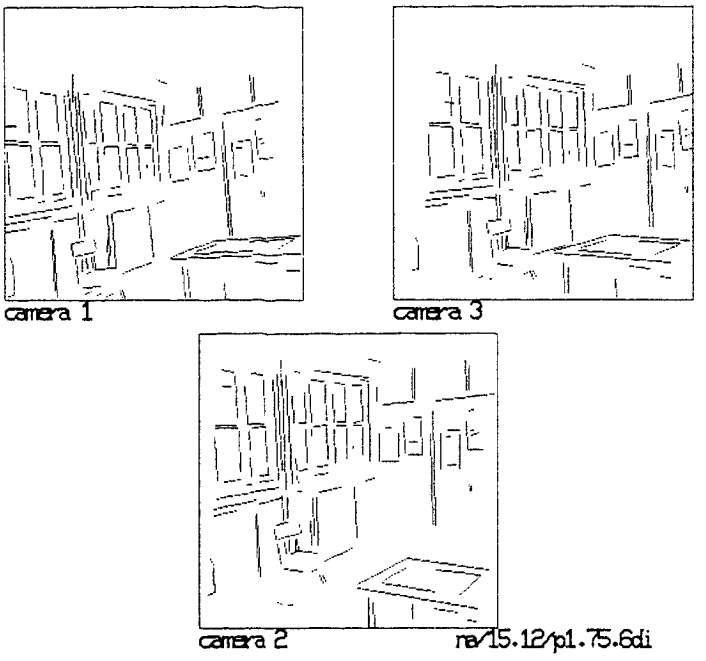

Figure 21: The matched line segments of the three images

"synthetic" image after merging the three views displayed from two viewpoints. This should be compared with Figure 26. The definition has improved everywhere, including near the door.

The careful reader will have noticed that it looks as if we had been able to look behind the wall or the door ! In fact, this is readily explained by the fact that for the door, for example there are extended tetrahedra which link it to details in the back of the hallway ; these extended tetrahedra are then intersected by one or two stereo triangles and therefore removed. This relates to a remark we made in Section 4 on the possibility of counting for each tetrahedron the number of stereo triangles it intersects and removing only the tetrahedra which intersect a sufficiently large number of stereo triangles. Here the number is one.

8 Conclusions and comparisons with previous work

We have proposed in this article a way to represent stereo data based on the use of the Delaunay triangulation. The proposed scheme provides an explicit volume representation of free space and a polyhedral representation of obstacles and objects. We have shown that the representation could be efficiently computed and updated by using the properties of the Delaunay triangulation and a simple visibility constraint.

These ideas are by no means limited to stereo data and in particular for mobile robots

applications there is no problem in throwing in range data obtained either by sonars or any kind of active sensing device, for example a laser range finder.

Nonetheless many questions remain open :

1. Can the visibility test be included in the computation of the Delaunay triangulation
2. Can we make better use of the fact that the algorithm for computing the Delaunay triangulation is sequential to process the 3D snapshots as they become available.
3. Can the polyhedral representation of the objects, which is inherently discontinuity preserving, be used to come up with efficient and accurate interpolation schemes.

The answer to questions 1 and 2 is probably yes, and the answer to 3 is yes, very likely so.

It may be interesting here to compare our approach to previous approaches to the problem of interpolating Stereo data. In his pionneering work on the subject, Grimson ([Gri81,Gri82,Gri83]) proposed to interpolate range data by a function which was minimizing the quadratic variation in gradient. he was not concerned with the integration of several views but was nonetheless confronted to the problem of depth discontinuities. Indeed, his technique implicitely assumes that the pieces of surface are in fact pieces of a single surface and, when discontinuities are present, they are considerably smoothed by the interpolation process.

Terzopoulos ([Ter83]) addressed the same problem but used different numerical techniques. In particular, he used multigrid techniques to speed up the computation time of Grimson's method. Nonetheless, the problem of discontinuities was still unsolved and he did not tackle either the problem of integrating several viewpoints.

Our solution to the problem of 3D data interpolation using the Delaunay triangulation explicitely solves the problem of depth discontinuities since the Delaunay triangulation preserves these discontinuities, and also solves the problem of integrating a variety of viewpoints thus allowing us to deal naturally with multiple valued functions of depth whereas with one view one deals only with single valued functions.

Another related work is that of Herman, Kanade, and Kuroe ([HKK84]) who developed a system to incrementally acquire a 3D model of a complex urban scene from stereo pairs obtained from a variety of viewpoints. In their work, there was no automatic matching of the different views as in [FAF86,AF87a,AF87b], the building of the surface model from one stereo pair was somewhat heuristic as was the combination of the 3D models obtained from different places. We believe that our geometric approach is a step forward a systematic treatment of these problems

Acknowledgements:

We want to thank Michel Schmitt for pointing out to us the results described in the Appendix and their possible application to the analysis of our algorithms.

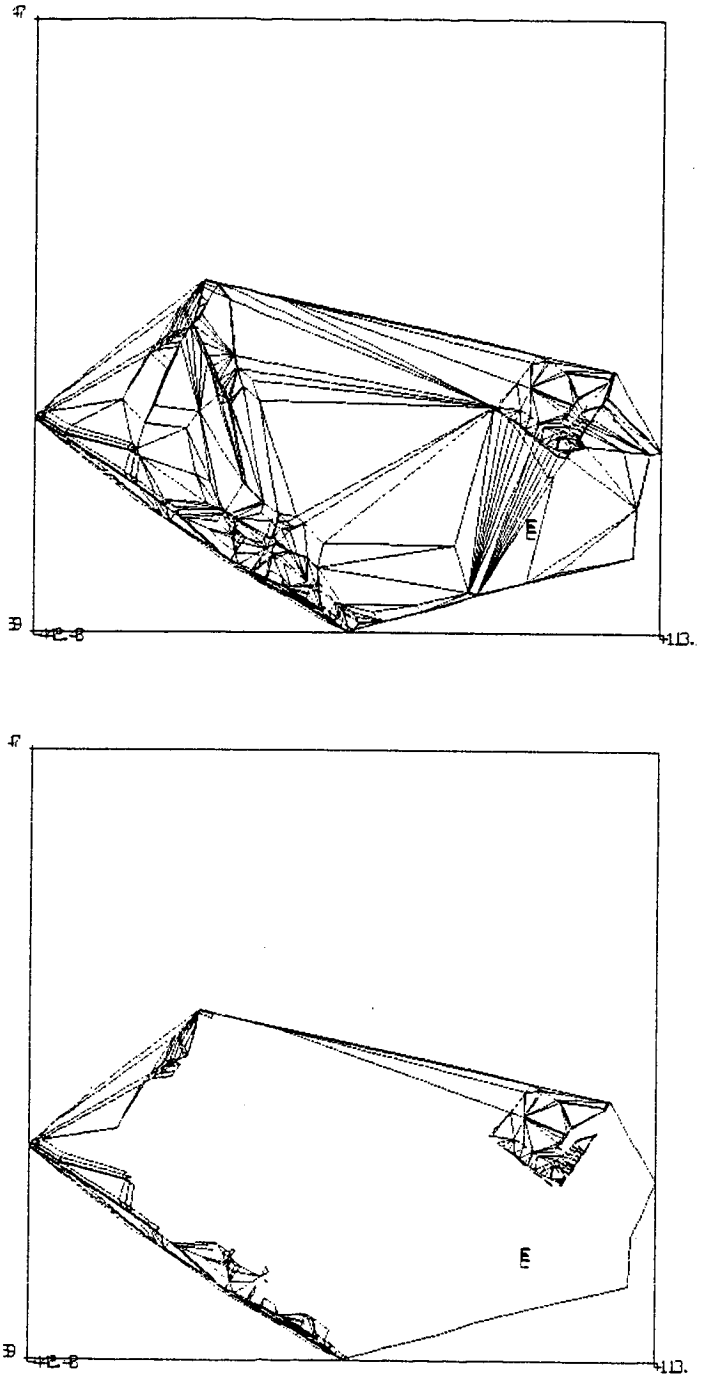

Figure 22: A cross-section of the Delaunay triangulation with a plane parallel to the floor before and after the removal of empty tetrahedra (using one viewpoint).

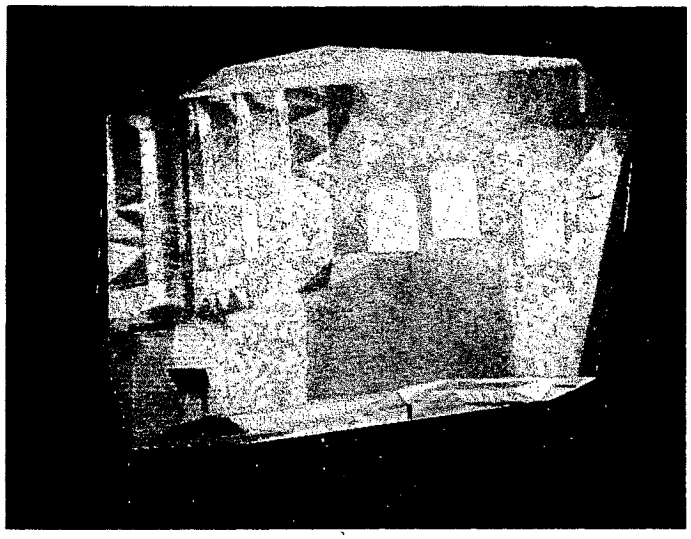

Figure 23: A "synthetic" image created from the polyhedral model of the scene shown in figure 20

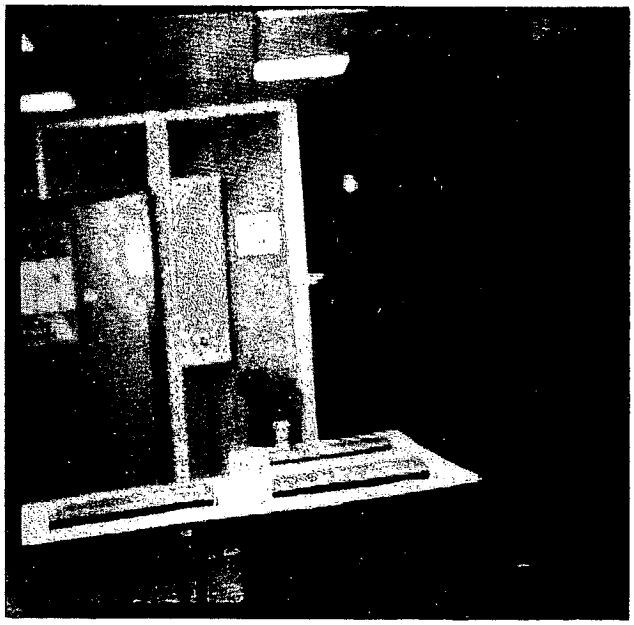

Figure 24: The original image

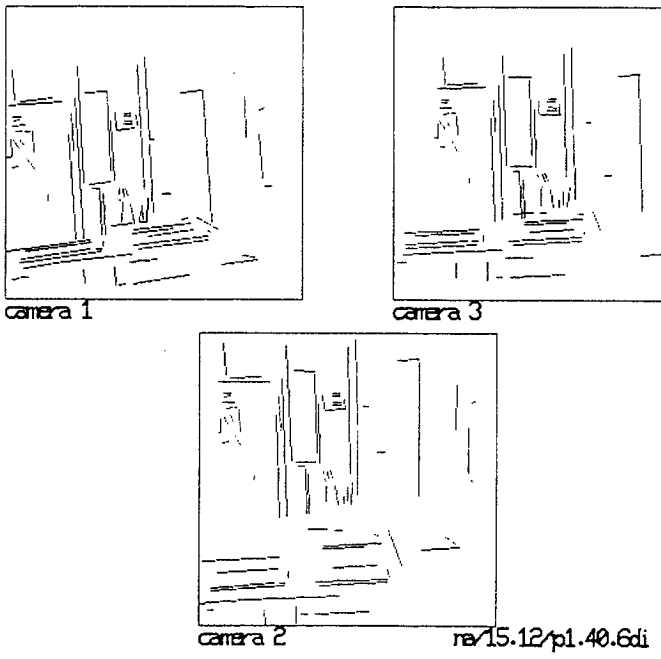

Figure 25: The matched segments

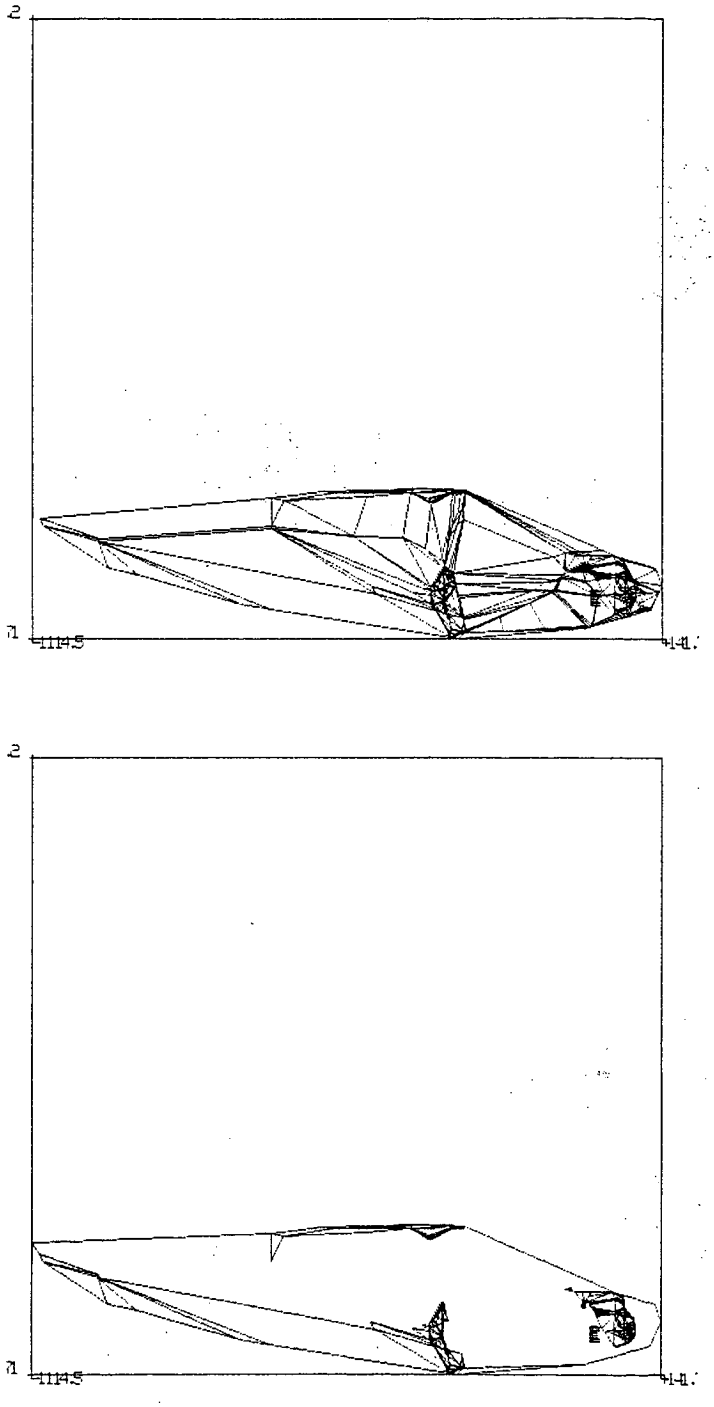

Figure 26: A cross-section of the Delaunay triangulation with a plane parallel to the floor before and after the removal of empty tetrahedra (using one viewpoint).

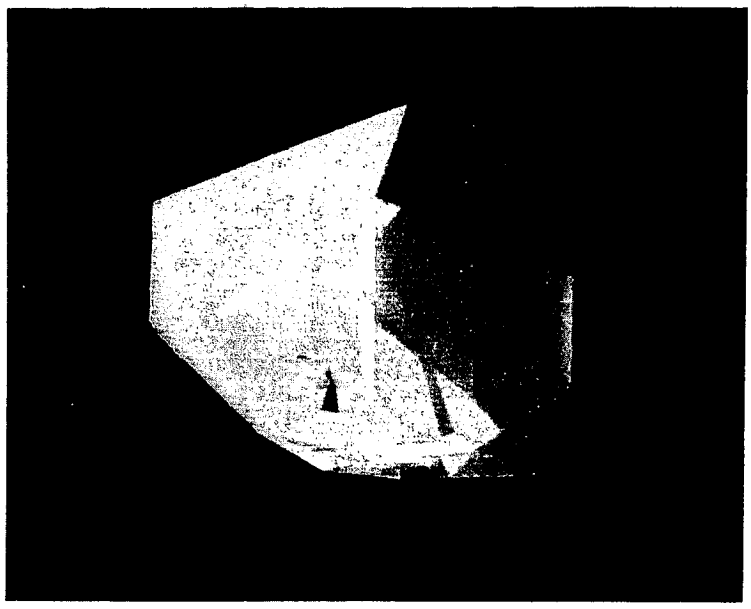

Figure 27: A "synthetic" image created from the polyhedral model of the scene shown in figure 24

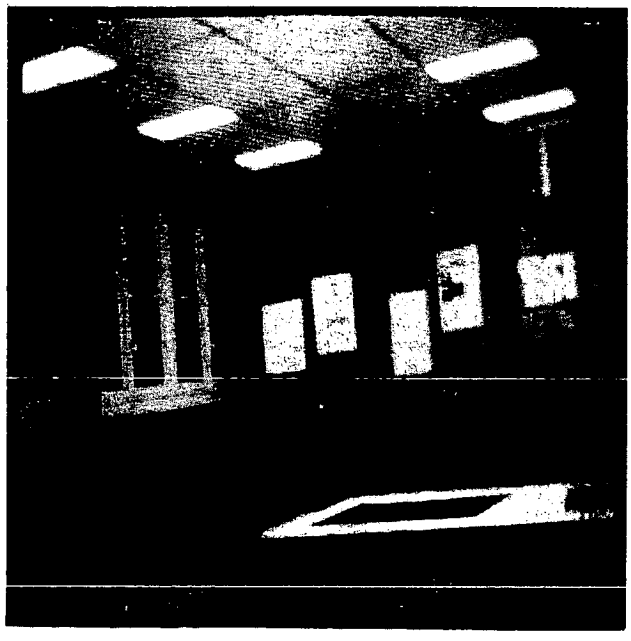

Figure 28: The second viewpoint corresponding to figure 20

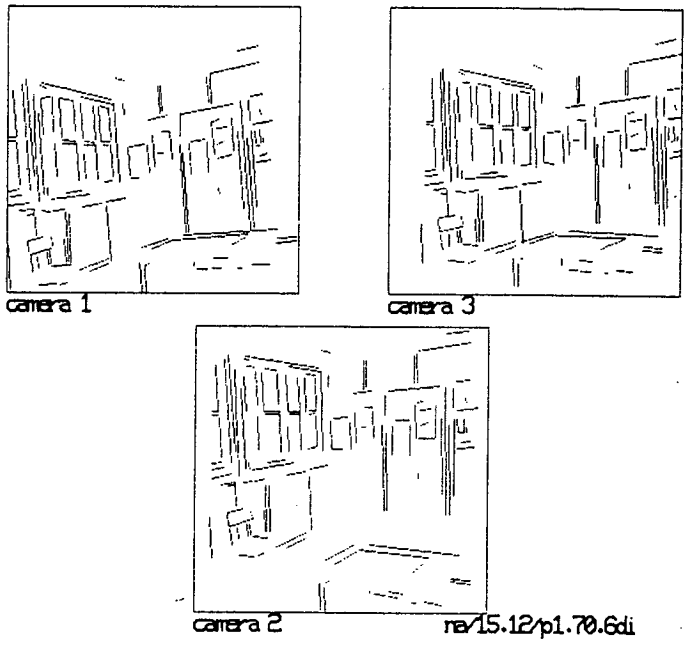

Figure 29: The three sets of segments corresponding to figure 28

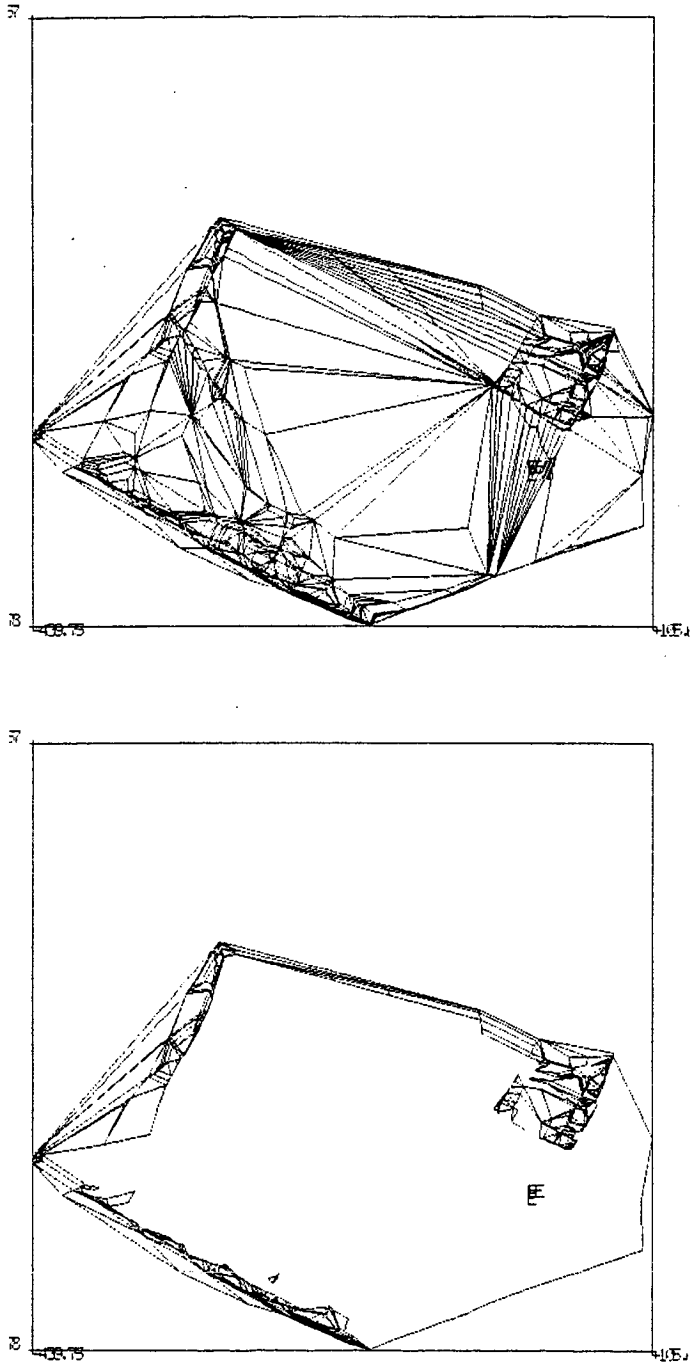

Figure 30: A cross-section of the Delaunay triangulation with a plane parallel to the floor before and after the removal of empty tetrahedra (using two viewpoints).

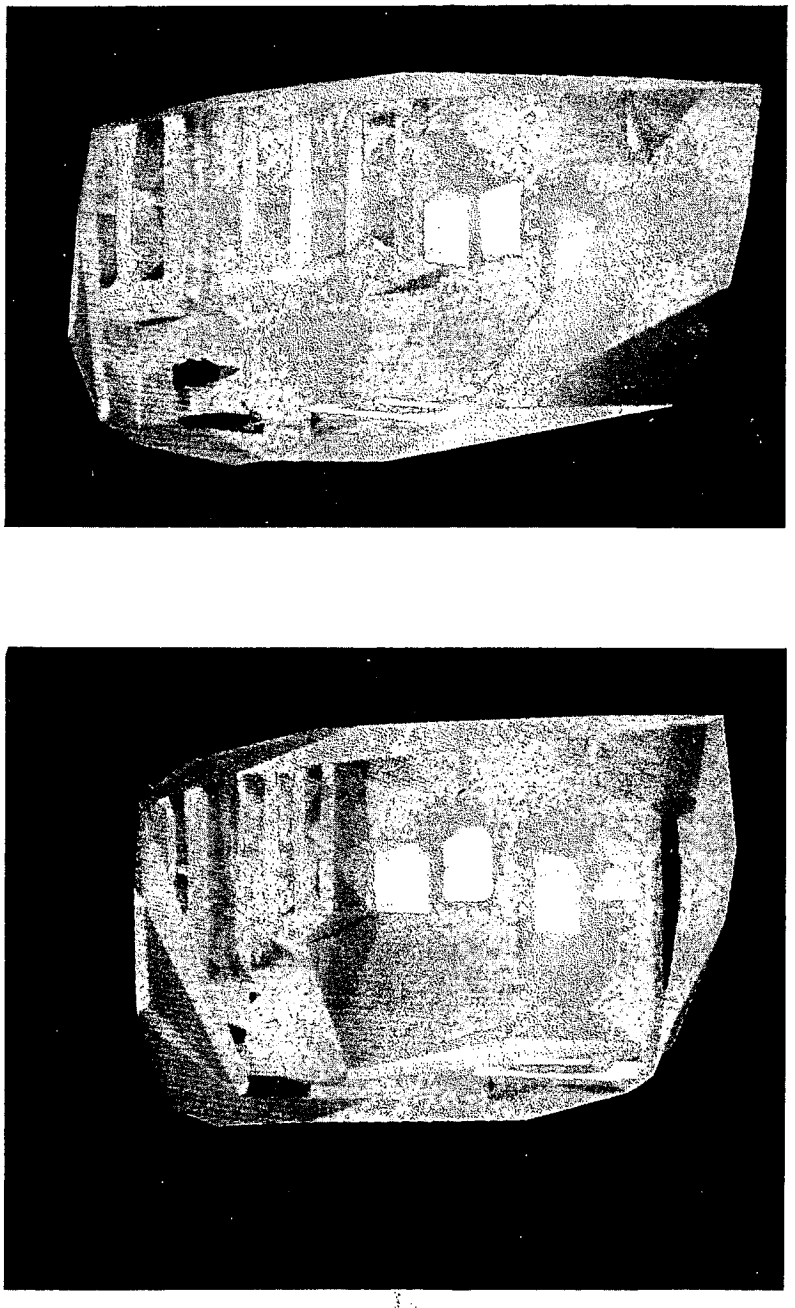

Figure 31: The "synthetic" image obtained by combining figures 20 and 28 shown from two different viewpoints.

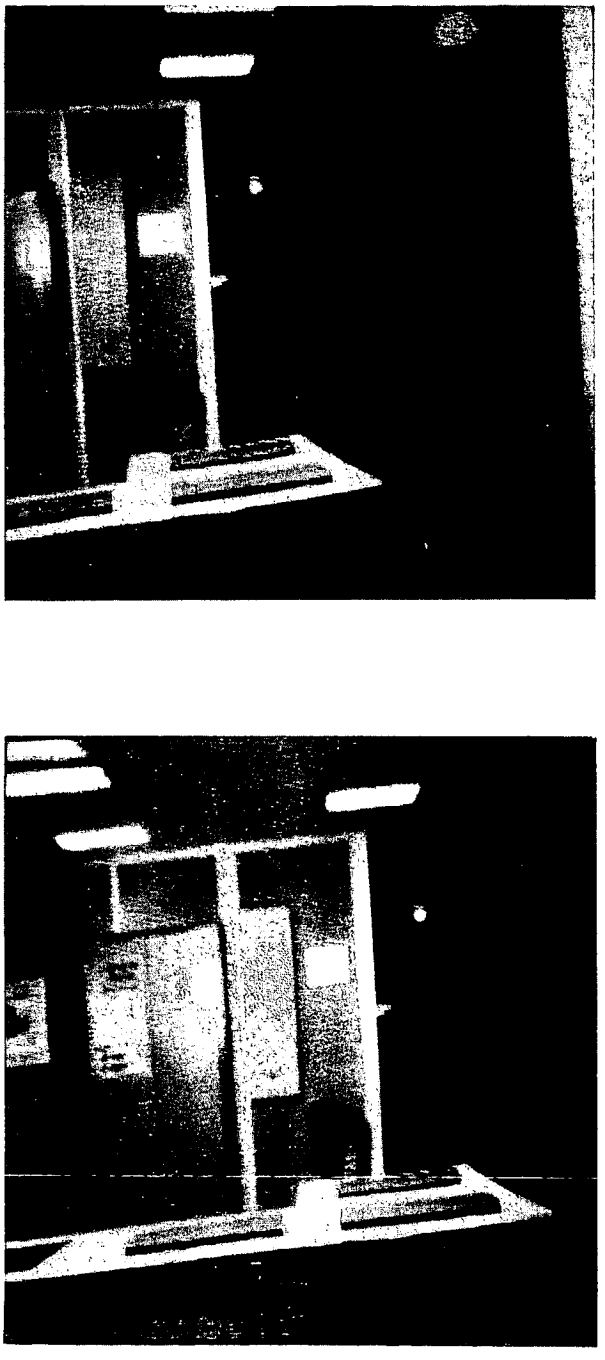

Figure 32: The two extra viewpoints corresponding to figure 24

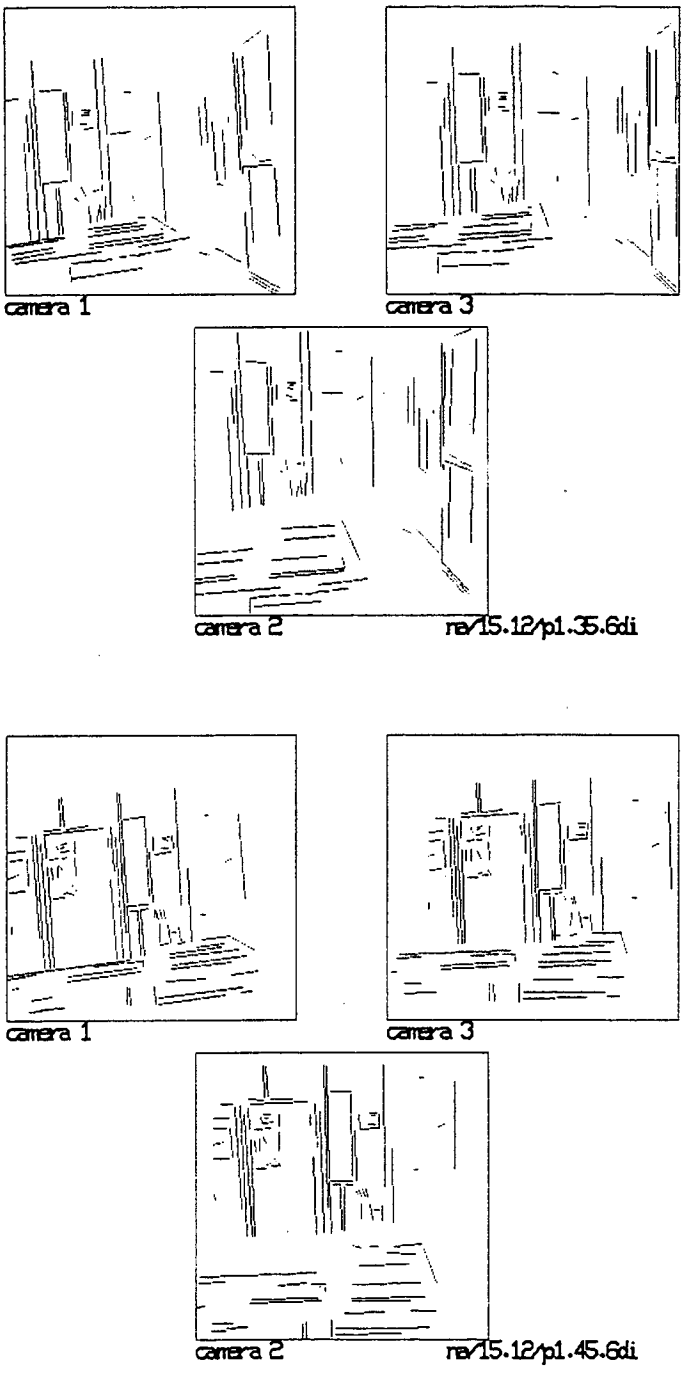

Figure 33: The three sets of segments corresponding to images shown in figure 32

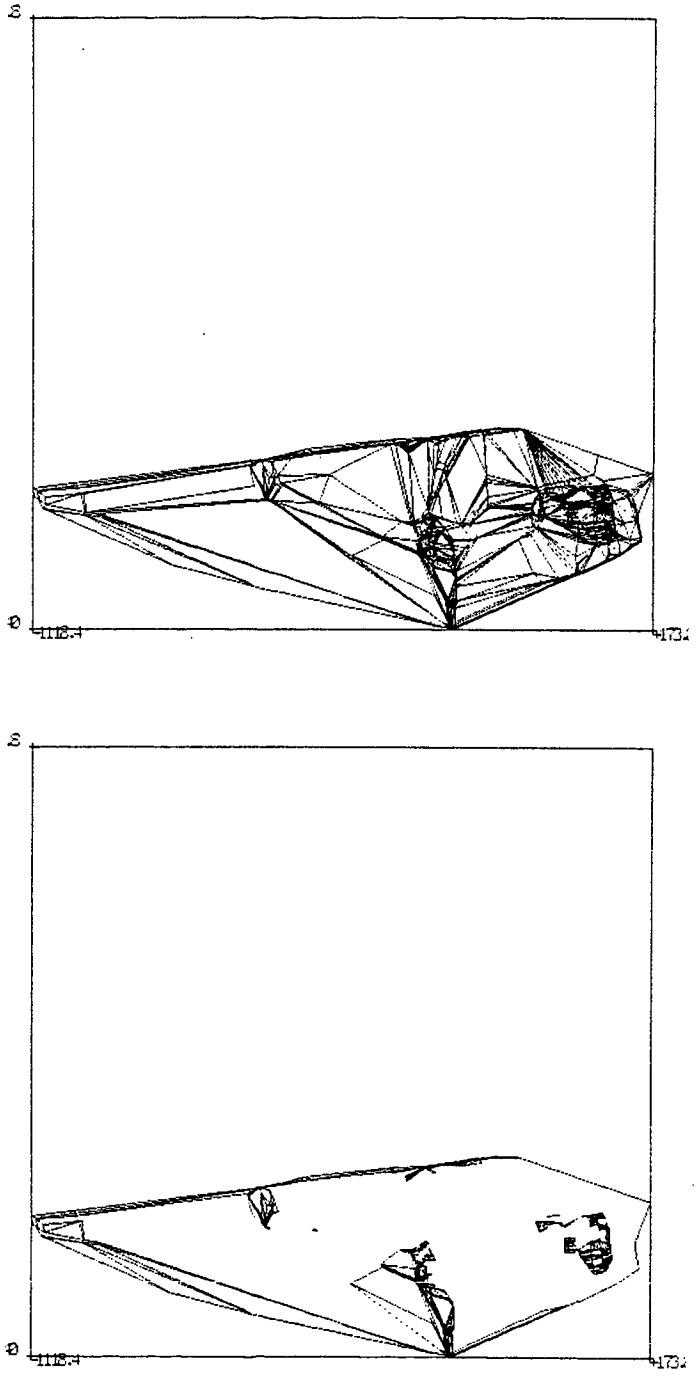

Figure 34: A cross-section of the Delaunay triangulation with a plane parallel to the floor before and after the removal of empty tetrahedra (using three viewpoints).

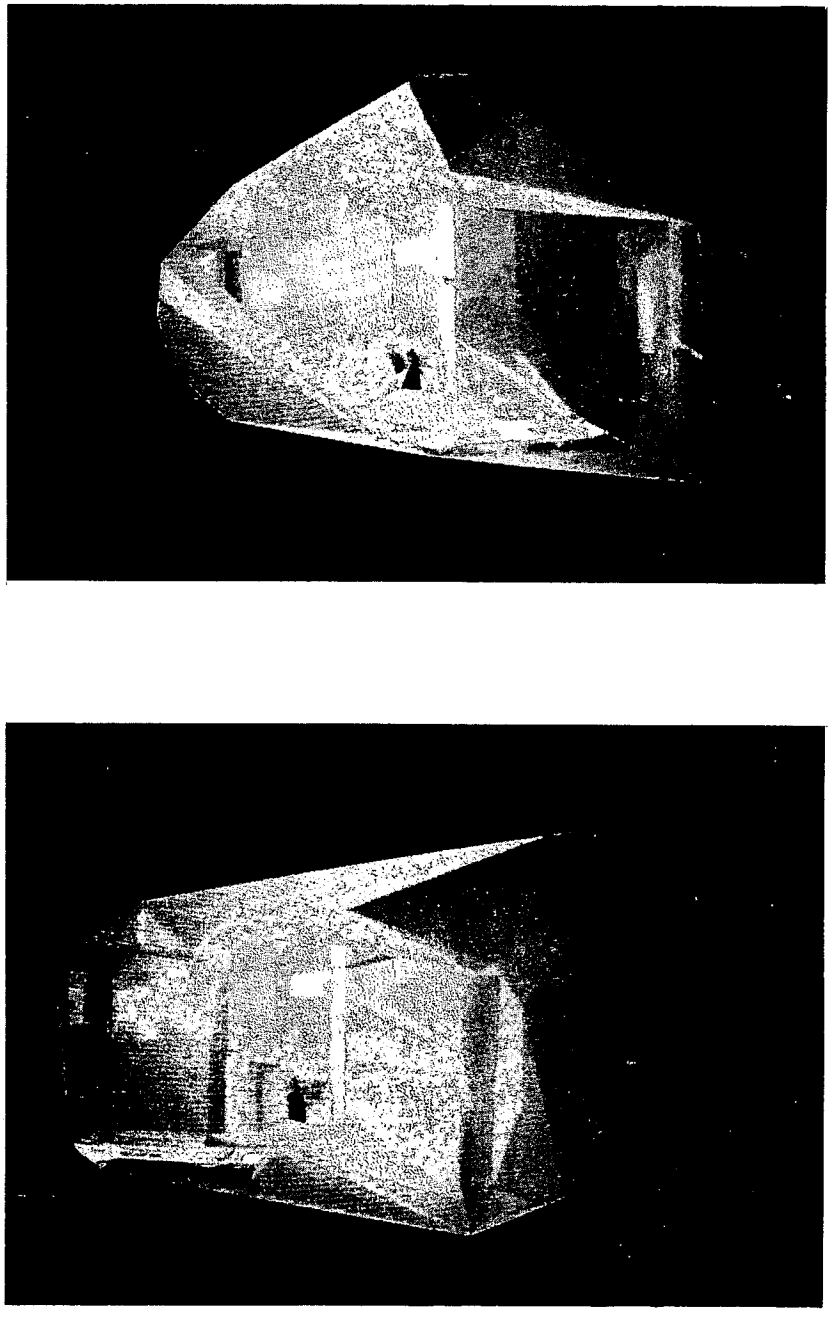

Figure 35: The "synthetic" image obtained by combining figures 24 and 32

A Appendix

Let δ be a stationary Poisson distribution of points in space, whose parameter θ is the density of points. For example, in the plane, this density can be estimated by counting the number n of points falling inside a square of size L:

$$\theta = \frac{n}{L^2}$$

To each of these points x, we associate a random closed set almost surely compact X_x called the *primary grain*. In what follows, X will be either a line segment or a triangle.

Let $\mathcal{B}$ be a deterministic compact. We consider for each x, the dilated set of X_x by $\mathcal{B}$:

$$X_x \oplus \mathcal{B} = \{y + b,\ y \in X_x,\ b \in \mathcal{B}\}$$

and we denote $Z = \bigcup_x X_x$.

We have the following lemma :

lemma 2 *The probability that no compact X_x intersects $\mathcal{B}$, is defined if and only if the area of $X_x \oplus \mathcal{B}$ is finite $\forall \mathcal{B}$, and in this case, is equal to :*

$$P(\mathcal{B} \subset Z^c) = \exp(-\theta E\left[\mu(X \oplus \mathcal{B})\right])$$

where μ is the Lebesgue measure and $E[\phi]$ is the expectation of the random variable ϕ.

The proof is given in [Mat75].

We can apply this result to a set of segments or triangles. F. Prêteux and M. Schmitt have shown that the probability $P(\mathcal{B} \subset Z^c)$ is independent of the location of the point in X, see [PS87b] and pages 16-21 of [PS87a].

A.1 The case of segments

For each random point x, X_x is a segment whose midpoint is x. Its length l and, in the planar case, its orientation β are randomly and uniformly distributed. Let $\bar{l}$ be the average value of l and we choose, in the planar case, for the angle β a uniform distribution between $-\pi/4$ and $\pi/4$.

A.1.1 $\mathcal{B}$ is a circle

$\mathcal{B}$ is a circle whose radius is ρ. The Lebesgue measure of $X_x \oplus \mathcal{B}$ is the area of $X_x \oplus \mathcal{B}$.

$$\mu(X_x \oplus \mathcal{B}) = 2\,\rho l + \pi\rho^2$$

According to lemma 2:

$$P(\mathcal{B} \subset Z^c) = \exp\left(-\theta\ \rho\left(2\,\bar{l} + \pi\rho\right)\right)$$

Let $\alpha = \theta\rho(2\,\bar{l} + \pi\rho)$, it can be shown ([PS87a], and [PS87b]) that the probability ϱ that p segments intersect $\mathcal{B}$ is a Poisson probability, and $\varrho = e^{-\alpha}\,\alpha^p/p!$. The average number $\overline{ns}$ of segments intersecting $\mathcal{B}$ equals α and therefore :

$$\overline{ns} = \theta\,\rho\left(2\,\bar{l} + \pi\rho\right)$$

A.1.2 $\mathcal{B}$ is a sphere

$\mathcal{B}$ is a sphere whose radius is ρ. The Lebesgue measure of $X_x \oplus \mathcal{B}$ is the volume of $X_x \oplus \mathcal{B}$:

$$\mu(X_x \oplus \mathcal{B}) = 2\pi\,\rho l + \frac{4}{3}\pi\rho^3$$

Similarly to the 2D case we find that:

$$\overline{ns} = \theta\,\rho\left(2\pi\,\bar{l} + \frac{4}{3}\pi\rho^2\right)$$

A.1.3 $\mathcal{B}$ is a square

Let λ be the width of $\mathcal{B}$.

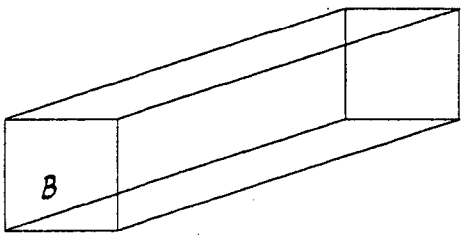

Figure 36: $X_x \oplus \mathcal{B}$

The Figure 36 shows the shape of $X_x \oplus \mathcal{B}$ and its area is given by :

$$\mu(X_x \oplus \mathcal{B}) = \lambda\left(\sqrt{2}\,\cos(\pi/4 - \beta) + \lambda\right)$$

Taking into account the fact that β is uniformly distributed, according to lemma 2:

$$P(\mathcal{B} \subset Z^c) = \exp\left(-\theta\,\lambda\left(\bar{l}\,\sqrt{2} + \lambda\right)\right)$$

From this we deduce the average number $\overline{ns}$ of segments intersecting $\mathcal{B}$:

$$\overline{ns} = \theta\lambda\left(\sqrt{2}\,\bar{l} + \lambda\right)$$

A.2 The case of triangles

Let X be a triangle. We decompose the square $\mathcal{B}$ as a Minkowski's sum of two segments:

$$\mathcal{B} = [O, (\lambda, 0)] \oplus [O, (0, \lambda)] = \mathcal{B}_x \oplus \mathcal{B}_y$$

Since the Minskowski's addition is associative, we have the following result:

$$X \oplus (\mathcal{B}_x \oplus \mathcal{B}_y) = (X \oplus \mathcal{B}_x) \oplus \mathcal{B}_y.$$

There exists a smallest rectangle $\mathcal{R}$ containing X,

$$\mathcal{R} = [(x_a, y_a); (x_b, y_a)] \times [(x_a, y_a); (x_a, y_c)]$$

The area of $X \oplus (\mathcal{B}_x \oplus \mathcal{B}_y)$ is shown in Figure 37.

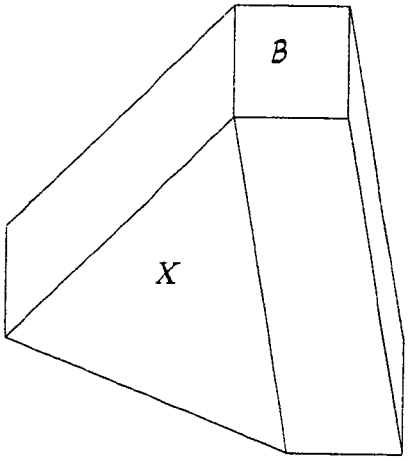

Figure 37: The area of $\mu\left(X \oplus (\mathcal{B}_x \oplus \mathcal{B}_y)\right)$

Using the associativity property, we write:

$$\mu\left(X \oplus (\mathcal{B}_x \oplus \mathcal{B}_y)\right) = \mu\left((X \oplus \mathcal{B}_x) \oplus \mathcal{B}_y\right) = \mu(X) + \lambda\left(y_c - y_a\right) + \lambda\left(x_b + \lambda - x_a\right)$$

which can be rewritten as:

$$\mu(X) + \lambda\left(x_b - x_a + y_c - y_a\right) + \lambda^2$$

The mathematical expectation of $\mu(X \oplus (\mathcal{B}_x \oplus \mathcal{B}_y))$ can be written as a fonction of the average value of the area of the triangles $\overline{A}$, and of the dimensions $\overline{x}$ and $\overline{y}$ of the rectangles.

$$E[\mu(X \oplus \mathcal{B})] = \overline{A} + \lambda(\overline{y} + \overline{x}) + \lambda^2$$

Applying again lemma 2, we deduce the average number $\overline{nt}$ of triangles intersecting a bucket :

$$\overline{nt} = \theta\left(\overline{\mathcal{A}} + \lambda\left(\overline{x} + \overline{y}\right) + \lambda^2\right)$$

References

[ABY87] P.D. Alevizos, J.D. Boissonnat, and M. Yvinec. An Optimal O(nlogn) Algorithm for Contour Reconstruction from Rays. In *Third Symposium on Computational Geometry*, June 1987.

[AEII85] T. Asano, M. Edahiro, H. Imai, and M. Iri. *Computational Geometry*, chapter Practical Use of Bucketing Techniques in Computational Geometry, pages 153–195. Elsevier Science Publishers B.V. (North Holland), 1985.

[AF87a] N. Ayache and O.D Faugeras. Building, registrating and fusing noisy visual maps. In *Proc. 1st ICCV*, June 1987.

[AF87b] N. Ayache and O.D Faugeras. Maintaining representations of the environment of a mobile robot. In *Proc. 4th Symposium on Robotics Research*, Santa-Cruz, USA, aug 1987.

[AF87c] Nicholas Ayache and Bernard Faverjon. Efficient Registration of Stereo Images by Matching Graph Descriptions of Edges Segments. *International Journal of Computer Vision*, 1, 1987.

[AL87a] N. Ayache and F. Lustman. Fast and reliable passive trinocular stereovision. In *Proc. First International Conference on Computer Vision*, pages 422–427, IEEE, June 1987. London, U.K.

[AL87b] N. Ayache and F. Lustman. Trinocular stereovision, recent results. In *Proc. International Joint Conference on Artificial Intelligence*, aug 1987. Milano, Italy.

[BN78] Harris Blum and Roger N. Nagel. Shape description using weighted symmetric axis features. *Pattern Recognition*, 10:167–180, 1978.

[Boi84] J.D. Boissonnat. Geometric Structures for Three-Dimensional Shape Representation. *ACM Transactions on Graphics*, 3:266–286, October 1984.

[Boi85a] J.D. Boissonnat. An Automatic Solid Modeler for Robotics Applications. In O. Faugeras and G. Giralt, editors, *Robotics Research : The Third International Symposium*, pages 65–73, 1985.

[Boi85b] J.D. Boissonnat. Reconstruction of Solids. In *First ACM Symposium on Computational Geometry*, Baltimore, June 1985.

[Bow81] A. Bowyer. Computing Dirichlet Tessellations. *Computer Journal*, 24:162–166, 1981.

[BT86] J.D. Boissonnat and M. Teillaud. A Hierarchical Representation of Objects : the Delaunay Tree. In *Second ACM Symposium on Computational Geometry*, June 1986.

[CA86] C.H. Chien and J.K. Aggarwal. Volume/Surface Octrees for Representation of Three- Dimensional Objects. *Computer Vision, Graphics and Image Processing*, 36:100–113, 1986.

[DC76] Manfredo P Do Carmo. *Differential Geometry of Curves and Surfaces.* Prentice Hall, Englewood Cliffs, NJ, 1976.

[DH73] R.O. Duda and P.U. Hart. *Pattern Recognition and Scene Analysis.* Wiley Interscience, NEW-YORK, 1973.

[EKA84] M. Edahiro, I. Kokubo, and T. Asano. A New Point-Location Algorithm and its Practical Efficiency - Comparison with Existing Algorithms. *ACM Transactions on Graphics*, 3:86–109, April 1984.

[ES85] H. Edelsbrunner and R. Seidel. Voronoi Diagrams and Arrangements. In *First ACM Symposium on Computational Geometry*, pages 251–262, Baltimore, June 1985.

[FAF86] O.D. Faugeras, N. Ayache, and B. Faverjon. Building visual maps by combining noisy stereo measurements. In *Proceedings 1986 IEEE Conference on Robotics and Automation*, pages 1433–1438, San Francisco, USA, April 7-10 1986.

[Gri81] W. E. L. Grimson. *From Images to Surfaces: A Computational Study of the Human Early Vision System.* M.I.T. Press, Cambridge, MA, 1981.

[Gri82] W. E. L. Grimson. A computational theory of visual surface interpolation. *Phil. Trans. Roy. Soc. London*, B298:395–427, 1982.

[Gri83] W. E. L. Grimson. An implementation of a computational theory of visual surface interpolation. *Computer Vision, Graphics, and Image Processing*, 22:39–69, 1983.

[Her82] F. Hermeline. Triangulation Automatique d'un Polyèdre en dimension n. *RAIRO Analyse Numérique*, 16:211–242, 1982.

[HKK84] Martin Herman, Takeo Kanade, and Shigeru Kuroe. Incremental Acquisition of a Three-Dimensional Scene Model from Images. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, PAMI-6(3):331–340, 1984.

[JT80] C.L. Jackins and S.L. Tanimoto. Oct-Trees and their Use in Representing Three-Dimensional Objects. *Computer Graphics and Image Processing*, 14:249–270, 1980.

[Kle80] V. Klee. On the Complexity of d-dimensional Voronoi Diagramm. In *Arch. Der Math.*, pages 75–80, 1980.

[LL86] D.T. Lee and A.K. Lin. Generalized Delaunay Triangulation for Planar Graph. *Discrete Computational Geometry*, 1:201–217, 1986.

[Mat75] G. Matheron. *Random Sets and Integral Geometry.* Wiley, 1975.

[Nac82] Lee R. Nackman. Curvature Relations in Three-Dimensional Symmetric Axes. *Computer Graphics and Image Processing*, 20:43–57, 1982.

[PS86] F. Preparata and M.I. Shamos. *Computational Geometry, An Introduction.* Springer-Verlag, 1986.

[PS87a] F. Prêteux and M. Schmitt. *Analyse et Synthese de Fonctions Booleennes : Théorèmes de Caractérisation et Démonstrations.* Technical Report, Ecole des Mines, Fontainebleau, May 1987.

[PS87b] F. Prêteux and M. Schmitt. *Analyse et Synthese des Fonctions Booleennes.* Technical Report, Ecole des Mines, Fontainebleau, May 1987.

[Rog64] C.A. Rogers. *Packing and Covering.* Cambridge University Press, 1964.

[Sam84] H. Samet. The Quadtree and Related Hierarchical Data Structures. *Computing Surveys*, 16, June 1984.

[Sib78] R. Sibson. Locally Equiangular Triangulation. *Computer Journal*, 21:243–245, 1978.

[Ter83] D. Terzopoulos. Multilevel computational processes for visual surface reconstruction. *Computer Graphics and Image Processing*, 24:52–96, 1983.

[Wat81] P.P. Watson. Computing the n-dimensional Delaunay Triangulation with application to Voronoi Polytopes. *Computer Journal*, 24:167–172, 1981.

Imprimé en France
par
l'Institut National de Recherche en Informatique et en Automatique