*Artificial Intelligence for Engineering Design, Analysis and Manufacturing* (2000), *14*, 359–372. Printed in the USA.
Copyright © 2000 Cambridge University Press 0890-0604/00 $12.50

# Viewpoint-based shape recovery from multiple views

ALI AKGUNDUZ, DAN ZETU, AND PAT BANERJEE

Department of Mechanical Engineering, University of Illinois at Chicago, 842 W. Taylor St., 2039 Engineering Research Facility, M/C 251, Chicago, IL 60607

(RECEIVED October 2, 1999; ACCEPTED March 21, 2000)

**Abstract**

This paper describes an algorithm for surface reconstruction from a set of scattered three-dimensional points extracted from an image sequence. In this process, additional information (such as location of the viewpoints and the points visible from a particular viewpoint) is available and can be exploited for accurately recovering the shape of the objects portrayed in the images. Initially, the set of points is subjected to a Delaunay triangulation that fills the convex hull of the set of points with disjoint tetrahedra. The key idea of the shape recovery algorithm is to eliminate triangles that obstruct the visibility of points from certain viewpoints, whose locations are known from the image acquisition process. The major contribution of this paper is that we have been able to design an algorithm for surface reconstruction that handles a wide variety of shapes, as opposed to currently existing techniques.

**Keywords:** Delaunay Triangulation; Reverse Engineering; Surface Reconstruction

## 1. INTRODUCTION

In recent years, avenues for using detailed three-dimensional (3D) facility models have been widened with the emergence of Virtual Reality (VR) technology. To make efficient use of this technology, it is necessary to design a procedure for efficiently representing and manipulating a detailed facility. In most instances, 3D object models are not readily available. They need to be created by using commercial CAD software. This is a tedious and time-consuming task. An alternative approach to creating 3D representations of existing objects is to extract object models from camera images automatically. Using multiple camera images, the 3D coordinates of points of interest belonging to an object, extracted from two-dimensional (2D) images, can be measured. A technique for extracting 3D information from 2D images has been described in Zetu et al. (1998, 2000), in which points of interest are detected and their coordinates w. r. t. an object-centered coordinate system are measured. From the set of scattered 3D points, object topology has to be constructed.

There are general cases when no *a priori* information about points connectivity is available. The assumption is that the sets of points are relatively dense. In these cases, some general-purpose techniques for building the topology of the object surfaces are available. Such techniques are described briefly in Section 2. These techniques do not work well in all cases, especially for objects with small cavities, and they can certainly fail when the sets of points are not dense. Such is the case for sets of points recovered as a result of applying stereo vision algorithms to extract 3D information from camera images. But in this case, additional information is available and it can be exploited for accurately recovering the shape of the extracted objects.

A useful geometric structure for topology construction from sets of scattered points is *Delaunay triangulation*, which fills the space occupied by the 3D points with disjoint tetrahedra such that each point of the given set is a vertex of at least one tetrahedron in the triangulation. Constructing the Delaunay triangulation of the set of points is not sufficient, as the cavities of the object will be filled with redundant triangles. To preserve the "true" shape, the redundant triangles have to be eliminated. The main idea is to use the viewpoints from which the images were taken to define imaginary lines [called virtual lines (VL)], from each viewpoint, to the points of the object visible from that particular viewpoint. In this way, redundant triangles are intersected by the VL and eliminated from the triangulation.

In this article, we assume that the 3D points have been accurately reconstructed and that the noise in the images

Reprint requests to: Ali Akgunduz, Department of Mechanical Engineering, University of Illinois at Chicago, 842 W. Taylor St., 2039 Engineering Research Facility, M/C 251, Chicago IL 60607. E-mail: ali@virtual.me.uic.edu



https://doi.org/10.1017/S0890060400145020 Published online by Cambridge University Press

does not have a destructive effect on the accuracy of shape recovery. This can be ensured by using an adequate method for 3D reconstruction (such as the one described in Zetu et al., 1998, and others). The topic of recovering 3D structure from camera images is beyond the scope of this paper.

The remainder of this paper is organized as follows. In Section 2, related work is discussed. Section 3 presents some preliminary background in Delaunay triangulation. The shape recovery algorithm is described in Section 4. Results are presented in Section 5 and conclusions are drawn in Section 6.

## 2. RELATED WORK

Most techniques for surface reconstruction from a set of scattered points start with the Delaunay triangulation of the set. The final product of the Delaunay triangulation is the convex hull of the set of points. The convex hull cannot capture any information about the cavities of an object. Therefore, we need to "carve" the convex hull until we are left with a set of triangles that best approximates the real shape of the object.

Boissonnat (1984) suggested a Delaunay triangulation-based technique for shape recovery. In this technique, the convex hull of an object is sculptured by iteratively eliminating the tetrahedra that lie in the exterior of the object (redundant tetrahedra), according to a specific rule. Each tetrahedron belonging to the Delaunay triangulation of the object points has associated with it a value $V$ that is given by the maximum distance between the faces of the tetrahedron and the corresponding sectors of its circumscribing sphere. Boissonnat's rule is to select for elimination, at each iteration, only tetrahedra with exactly one face or exactly two faces on the convex hull. At each iteration, the tetrahedron with the largest $V$ is eliminated from the triangulation. This iterative process stops when no more tetrahedra can be eliminated. When the sets of points are relatively dense and cavities are relatively large, the exterior tetrahedra always have a larger maximum distance from their faces to the corresponding sectors of the circumscribing sphere than the interior ones and, therefore, Boissonnat's technique works well and does not fail. Not all sets of points enjoy the above-mentioned properties. There can be special cases where, according to Boissonnat's rule, an interior tetrahedron could have a higher priority than an exterior one. For example, in Figure 1, which shows a desk with a few tetrahedra resulting from the Delaunay triangulation superimposed on it, the tetrahedron ABEF (with base ABE drawn with dark lines) is an interior one, and the tetrahedron ABCD (with base ACD drawn with white lines) is exterior to the object. The angles at F and B are right angles, so both tetrahedra are enclosed by a hemisphere. Since BF $>$ BD and AF $>$ AD, the radius of the sphere circumscribing ABEF is greater than the radius of the sphere circumscribing ABCD, so the value ($V$) associated with ABEF (as defined above)

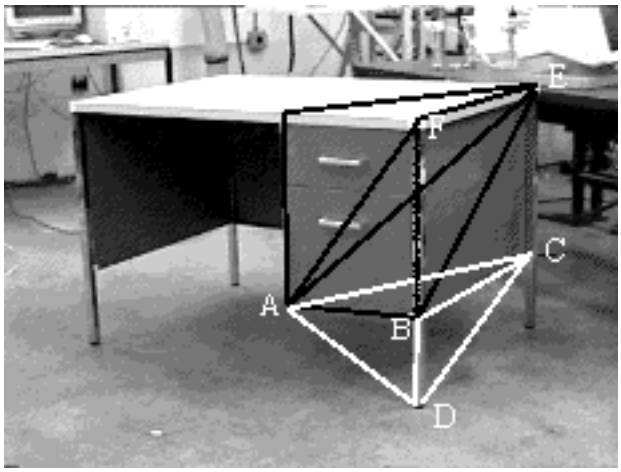



**Fig. 1.** Example of a part of an object (desk) for which Boissonnat's technique fails.

is greater than the value associated with ABCD. Both tetrahedra have two faces on the convex hull formed by the set of points defining the object. So, according to Boissonnat's rule, the tetrahedron ABEF will be eliminated first and consequently the shape of the object will be destroyed.

More recently, a new geometric structure proposed to define the shape of certain 3D objects is the *alpha-shape* concept, described in Edelsbrunner and Mucke (1994) and Mucke (1993). Conceptually, $\alpha$-shapes are a family of shapes, parameterized by a real $\alpha$, ranging from 0 to $\infty$. The $\alpha$-shape of a point set is a polytope that is neither necessarily convex nor necessarily connected. When $\alpha = \infty$, the $\alpha$-shape is identical to the convex hull of the point set. When $\alpha = 0$, the $\alpha$-shape is the point set itself, with no connection between any two points. When $\alpha$ decreases, certain triangles or edges disappear from the triangulation. Intuitively, a geometrical entity (triangle or edge) disappears from a triangulation when $\alpha$ becomes small enough so that a sphere with radius $\alpha$ can occupy the space of the geometrical entity, without enclosing any of its vertices.

As with Boissonnat's technique, the problem with $\alpha$-shapes is that they work well only for dense sets of points. Figures 2 and 3 illustrate two cases when the $\alpha$-shapes technique cannot provide an accurate representation of the "real" object. The two objects shown in these figures are defined by nondense sets of points and there are cavities that are "smaller" than the density of points. In Figure 2, the example of a desk is provided. No suitable $\alpha$-value was found to accurately represent the desk. The same situation occurs with the cubicles shown in Figure 3. A few $\alpha$-shapes are shown, but no value of $\alpha$ can recover the true shape of the object.

It is obvious, from previous examples, that a different approach is needed to accurately recover the shape of objects in a wider variety of cases, possibly with additional information. Such a technique is described later in this article.

The problem of reconstructing surfaces from range images has received a great deal of attention. Again, in these

https://doi.org/10.1017/S0890060400145020 Published online by Cambridge University Press

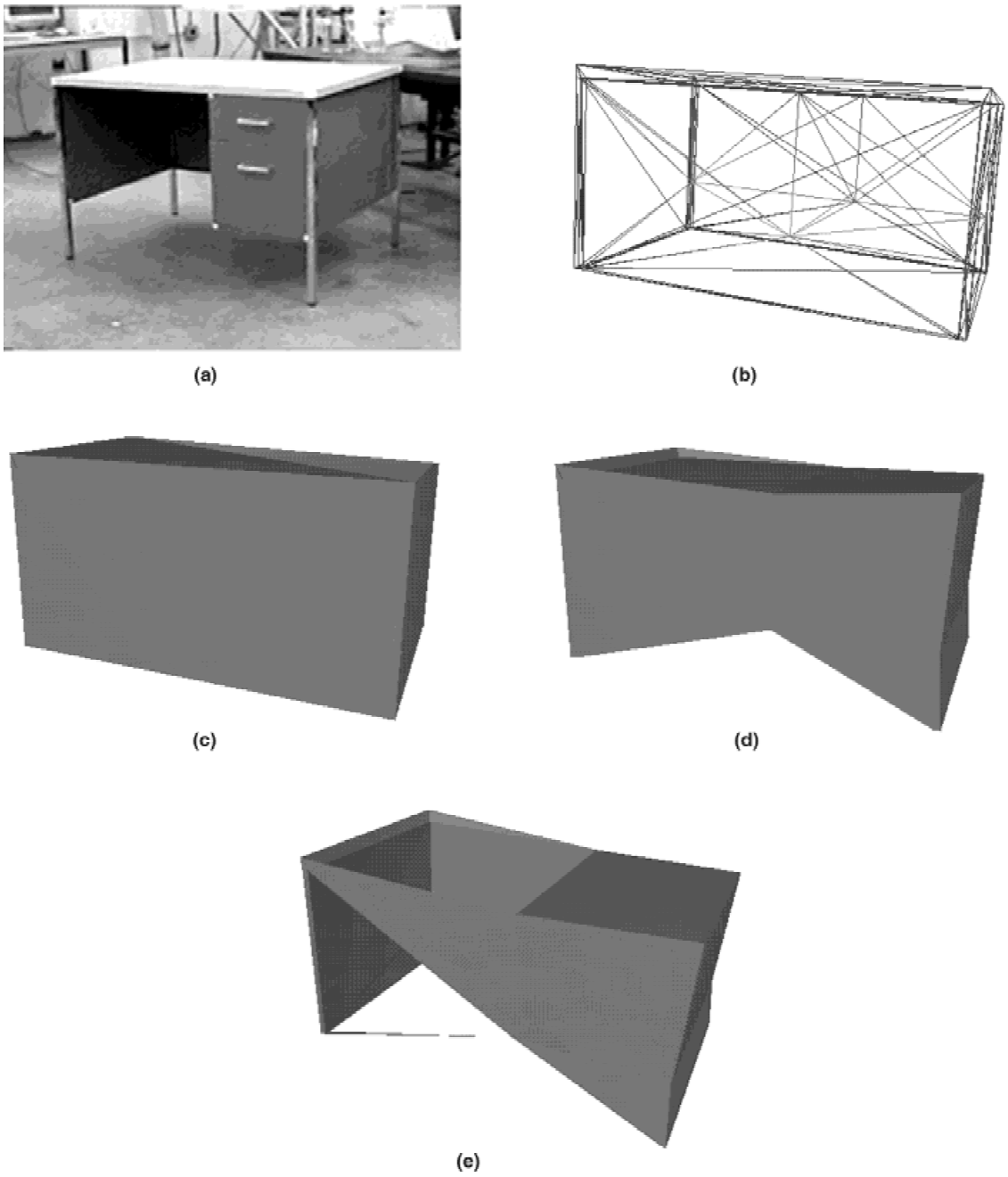



**Fig. 2.** $\alpha$-shapes of a desk in various cases. Note that the true shape of the desk cannot be recovered for any value of $\alpha$; (a) Original image; (b) Delaunay triangulation of the set of points; (c) Convex hull; (d) $\alpha = 50$; (e) $\alpha = 40$.

techniques, the basic assumption is that the sets of points are dense. There are two types of reconstruction from range images: reconstructing parametric surfaces (Turk & Levoy, 1994; Soucy & Laurendeau, 1995) or reconstructing an implicit function, usually a signed or weighted signed distance function (Chen & Medioni, 1992; Higuchi et al., 1995; Curless & Levoy, 1996; Hope et al., 1992). These techniques deal with additional issues, such as the registration between different range images. In our case, the set of points is obtained globally, after tracking the object points at the image level; therefore registration is not an issue.

## 3. DELAUNAY TRIANGULATION PRELIMINARIES

A triangulation $T$ of a 3D point set $S$ is a collection of disjoint tetrahedra filling the space covered by $S$, such that each point of $S$ is a vertex of at least one tetrahedron in $T$. The

https://doi.org/10.1017/S0890060400145020 Published online by Cambridge University Press

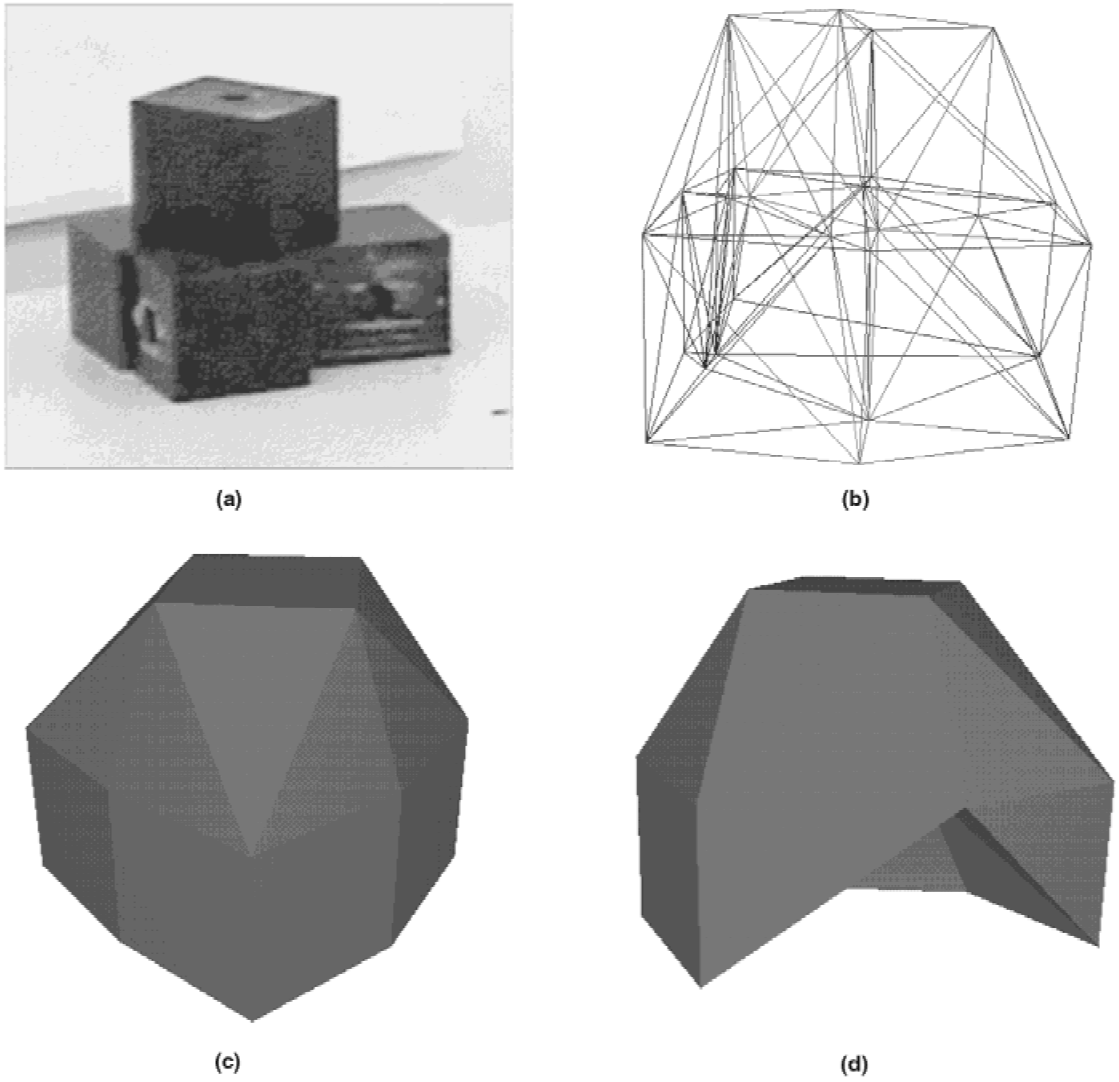



**Fig. 3.** Another example when $\alpha$-shapes do not work. (a) Original object; (b) Delaunay triangulation of the set of points; (c) Convex hull; (d) $\alpha = 95$.

triangulation $T$ is called a Delaunay triangulation, denoted by $D$, if the circumsphere of each tetrahedron *abcd* contains no point of $S$ other than its endpoints $a$, $b$, $c$, and $d$. This property is called *in-sphere criterion* and is illustrated in Figure 4.

In Figure 4, the points $a$, $b$, $c$, and $d$ form a tetrahedron that is a part of an already constructed triangulation. The point $e$ is not included in the interior of the circumsphere of the tetrahedron *abcd*, so the newly formed local triangulation will be Delaunay according to the definition. The point $e_1$ is inside the sphere, so the Delaunayhood is not preserved. The points $a$, $b$, $c$, $d$, and $e_1$ have to be retriangulated to obtain a local Delaunayhood. The point $e_2$ is located exactly on the sphere. In this case the set of five points is degenerate and the triangulation is not unique.

Delaunay triangulation is directly related to another geometric structure: the *Voronoi diagram* of the set of points. Let $S = \{s_1, s_2, \ldots, s_n\}$ be a set of points in the 3D Euclidean space. The space is partitioned by assigning every point in space to its nearest point of the set $S$. All those points assigned to $s_i$ form the *Voronoi region* (*Voronoi polyhedron*) $V(s_i)$. $V(s_i)$ consists of all the points at least as close to $s_i$ as to any other point of $S$:

$$V(s_i) = \{x : |s_i - x| \leq |s_i - x|, \quad \forall j \neq i\} \tag{1}$$

Some point sets do not have a unique nearest neighbor from $S$. The set of all points that have more than one nearest neighbor form the Voronoi diagram for the set $S$. The Delaunay triangulation is the dual of the Voronoi diagram, that is, it is obtained by joining the points of the set $S$ whose Voronoi polyhedra are adjacent (have a face in common). For a better illustration, in Figure 5, an example of Voronoi diagram and Delaunay triangulation of a small set of 2D points are shown.

https://doi.org/10.1017/S0890060400145020 Published online by Cambridge University Press

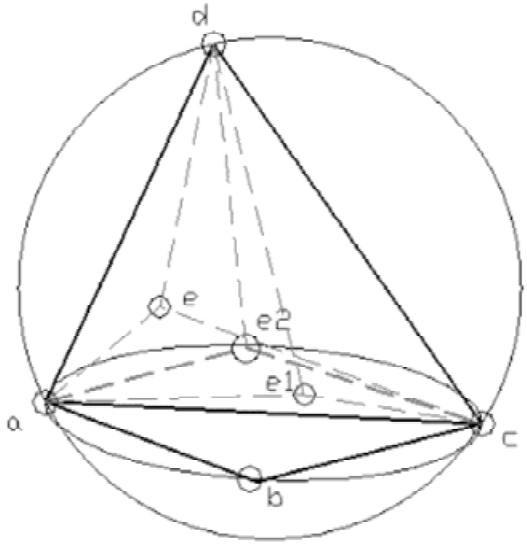



**Fig. 4.** In-sphere criterion for preserving the Delaunay triangulation at the local level of five points.

The design and implementation of robust Delaunay triangulation algorithms have been an area of extensive research (for reviews, see O'Rourke, 1994; Preparata & Shamos, 1985). We use the algorithm described in Edelsbrunner and Shah (1992) and Mucke (1993).

## 4. VIEWPOINT-BASED APPROACH FOR SHAPE RECOVERY

There are situations when more than the set of points is available. Such is the situation when the points have been extracted from camera images. A characteristic of this process is that the camera location is known at the time of image capture. As we describe in this section, we can exploit this additional information in order to improve the process of shape recovery.

A 2D image of an object portrays that object as seen from a certain viewpoint. It is not necessary to see all the points that completely define that particular object. In fact,

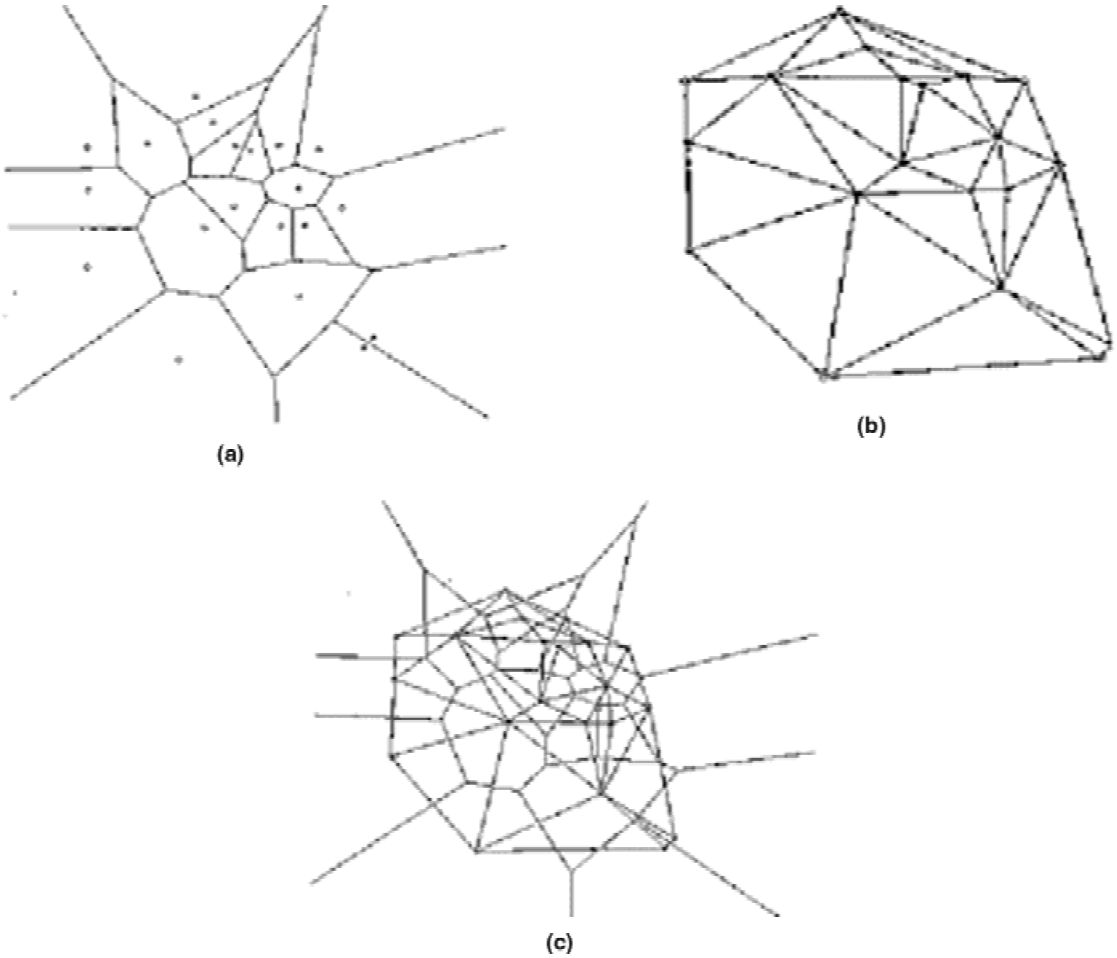



**Fig. 5.** (a) Voronoi diagram of a set of 2D points; (b) Delaunay triangulation of the points in (a); (c) Delaunay triangulation and Voronoi diagram of the same points.

https://doi.org/10.1017/S0890060400145020 Published online by Cambridge University Press

in a particular image, only a subset of the entire set of points is visible from the viewpoint from where the image was taken. When applying a Delaunay triangulation on the set of points, the convex hull is obtained. The convex hull fills the cavities of the object with unwanted triangles. Our approach is intuitively simple: define VL joining a viewpoint with the points visible from that viewpoint. Then those lines will intersect some exterior triangles. The triangles that are intersected by VL at least once are eliminated from the triangulation. This simple yet powerful concept is illustrated in Figure 6. In this figure, the VL intersects the triangles *ace*, *cde*, *ceg*, and *dhe* (for simplicity, only the intersections between the VL and triangles *ace* and *ceg* have been marked with circles). Consequently, triangles intersected by the VL are considered redundant (exterior) and are therefore eliminated from the triangulation.

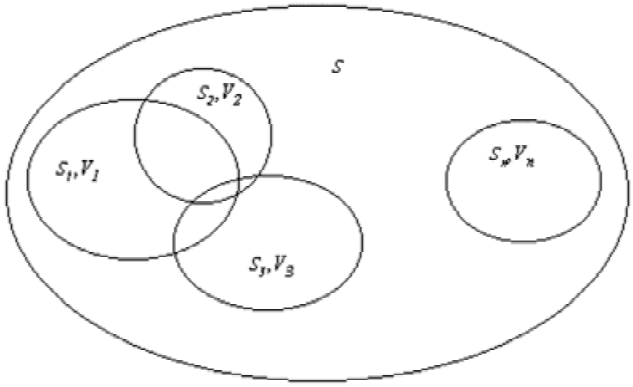



**Fig. 7.** Grouping the set of points $S$ into subsets $S_i$, associated with viewpoint $V_i$.

### 4.1. The algorithm

The set of points $S$ is obtained by tracking the points (extracted from images) from image to image, until all the available images are visited. For each image, we know the camera location (determined, e.g., by using the technique described in Zetu et al., 1998) and we also know the object points that are visible in that particular image.

The shape recovery procedure starts by grouping the point set $S$ into subsets $S_i$, $i = 1,2,\ldots,n$. Each subset contains points that were extracted from a particular image, corresponding to viewpoint $V_i$. The subsets can overlap. The grouping of the point set into subsets is illustrated in Figure 7.

The flowchart of shape recovery algorithm is shown in Figure 8 and the pseudo-code looks as follows:

```
–For( all viewpoints Vi and subsets Si)
 {
 • Define virtual lines between Vi and ev-
   ery point in Si
 • Mark all triangles intersected by vir-
   tual lines for deletion
 }
–Remove all tetrahedra that contain at
least one marked triangle
–Eliminate all triangles that lie in the
interior of the object
–Correct face normals
```

The key point of the algorithm is to find triangles obstructing the visibility of points from certain viewpoints. If there is such a triangle, that triangle is classified as redundant and is consequently marked for deletion. To determine whether a VL intersects a triangle, we need to determine whether the point of intersection between the line and the plane formed by the vertices of the triangle is located in the interior of the triangle.

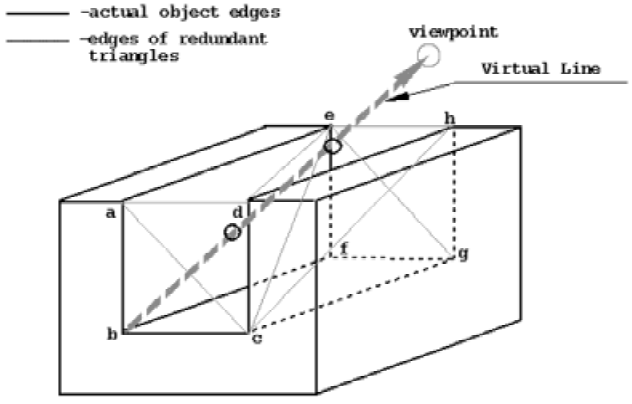



**Fig. 6.** Virtual line intersecting redundant triangles.

### 4.2. Point-in-polygon testing

The problem of determining whether a point lies inside or outside a polygon is an important one in computational geometry applications (O'Rourke, 1994; Preparata & Shamos, 1985). A very popular algorithm for solving this problem is known as Jordan curve theorem (Glassner, 1989). This algorithm works by shooting a ray in an arbitrary direction and counts the number of polygon edges crossed. If the number of crossings is odd, the point is inside the polygon; otherwise, it is outside. An example is given in Figure 9. In Figure 9, the ray R1 crosses the polygon twice, so its point of origin is outside the polygon. Ray R2 crosses only once, so it originates from a point that is inside the polygon. This technique gets complicated when a ray intersects a vertex of the polygon (R3 in Fig. 9), or the point of origin is located on an edge of the polygon (R4 in Fig. 9).

When the polygons are triangles, a simpler technique exists. First project the triangle in question onto the planes

https://doi.org/10.1017/S0890060400145020 Published online by Cambridge University Press

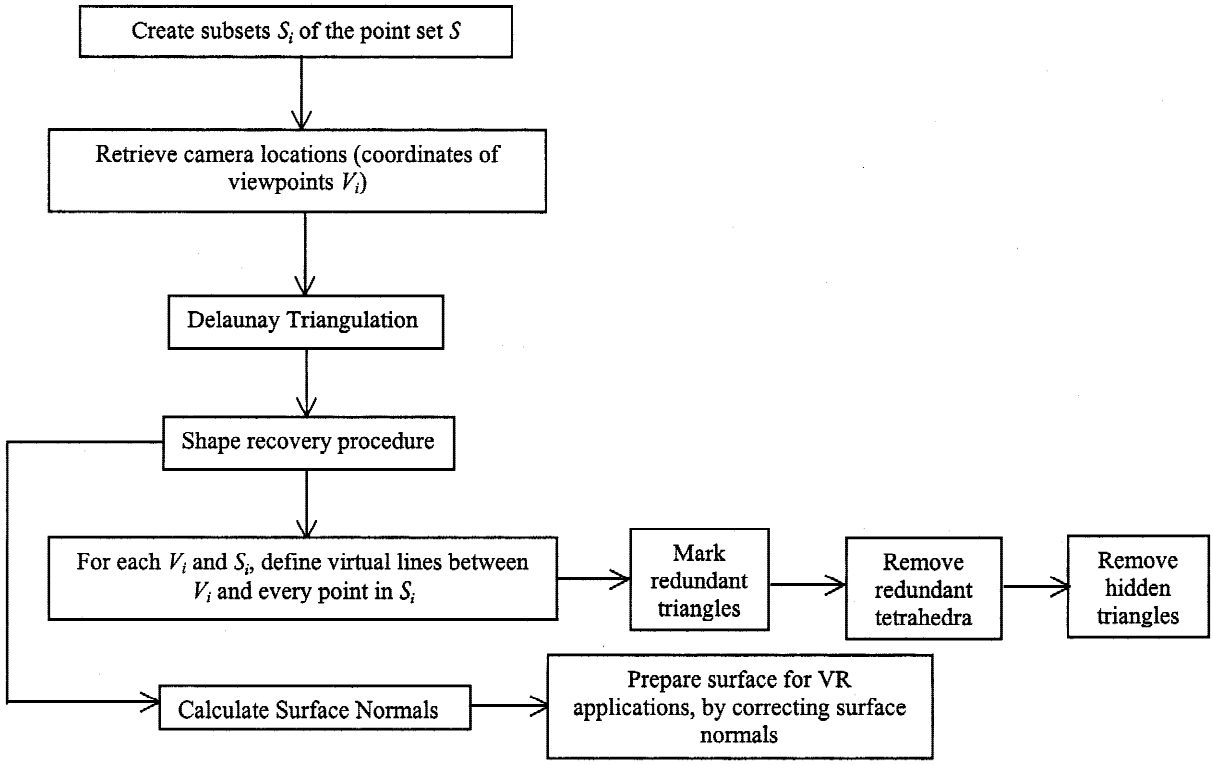



**Fig. 8.** Shape recovery procedure.

formed by the axes of the cartesian coordinate system. Thus, three projections are obtained: onto the $(xy)$, $(zy)$, and $(zx)$ planes, respectively (see Fig. 10). Out of the three projections, the one with the largest area is selected.

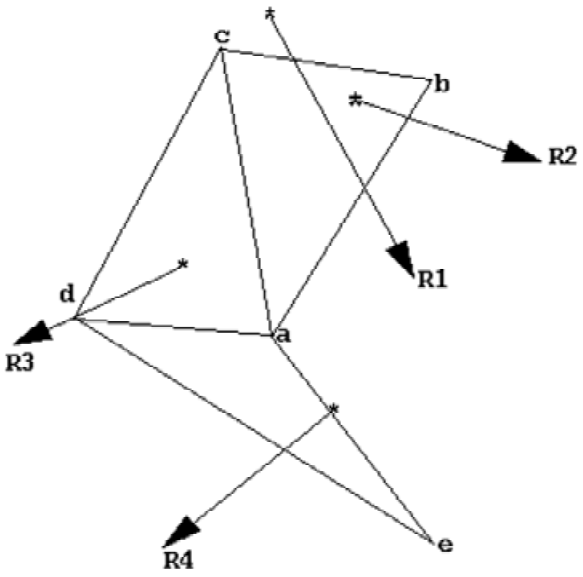



**Fig. 9.** Jordan curve theorem.

The rationale for selecting the projection with the largest area is to avoid degenerate cases, such as when two vertices of a triangle have the same coordinate along a coordinate axis (if two vertices have the same $x$ coordinate, for example, the projection onto $(xy)$ plane will have zero area and we cannot ascertain whether or not a point is inside or outside the triangle—see Fig. 11, where vertices $a$, $b$, and $c$ of triangle $abc$ have the same $Z$ coordinate, so projection onto $(zx)$ plane is reduced to the line $c'a'$).

Once the largest possible 2D projection of a triangle is selected, there can be two different relationships between the intersection point and triangle (Fig. 12). As is seen in Figure 12, if we connect the three vertices of the triangle with the intersection point, we obtain three angles, and if one of these angles is larger than or equal to 180°, we can conclude that the point is outside the triangle. If all three angles are less than 180°, we conclude that the intersection occurs inside of the triangle, and this triangle shall be marked as redundant.

In Figure 12, the point I is inside the triangle $abc$, and the point I′ is outside the triangle $a'b'c'$ (angle $Q_1'$ is greater than 180°).

To avoid ambiguities generated by rounding errors, we add a tolerance $(\epsilon)$. If an angle is greater than $(180° + \epsilon)$, the intersection point is exterior to the triangle (if one angle is exactly 180°, the intersection point is located on an edge of a triangle and this could generate an ambiguity due to rounding errors). In our experiments, $\epsilon = 0.01$.

https://doi.org/10.1017/S0890060400145020 Published online by Cambridge University Press

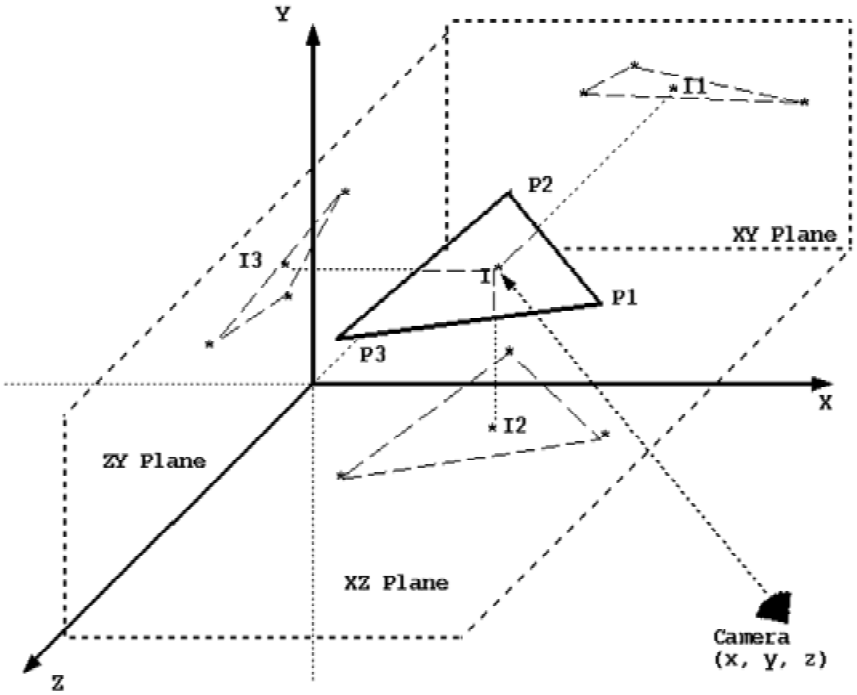



**Fig. 10.** Projections of a triangle onto coordinate planes.

### 4.3. Removing redundant tetrahedra

The goal of shape recovery algorithm is to remove exterior triangles, which obstruct the visibility of object points from a particular viewpoint. The algorithm demonstrated above is able to detect triangles that fall between camera and object points. It is not guaranteed that all the exterior triangles will be intersected by a VL at least once. This depends on the number of images available. To ensure that all the exterior triangles will be intersected, the operator in charge of image acquisition should consider all possible situations and capture images from various viewpoints. One case is studied in Figure 13, where the triangle formed by points 1, 4, and 5 is not intersected by any VL defined by two cameras oriented towards the object. When the size of the triangle is small relative to the object, or the number of viewpoints available is not sufficient, it is possible to have exterior triangles left undetected in the first stage of our algorithm. Two camera images are sufficient to recover the depth of the extracted points, but this number may not be sufficient,

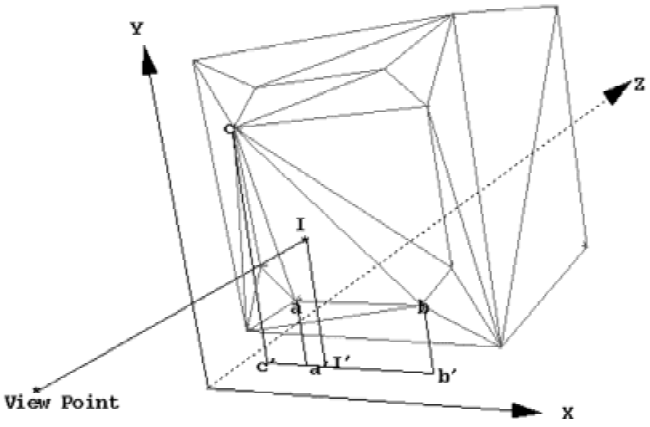



**Fig. 11.** Triangle *abc* is projected onto the plane (*zx*) into a line.

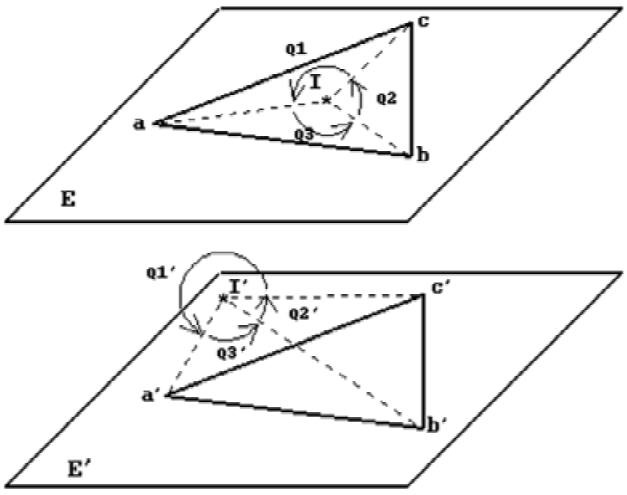



**Fig. 12.** Intersection point inside or outside a triangle.

https://doi.org/10.1017/S0890060400145020 Published online by Cambridge University Press

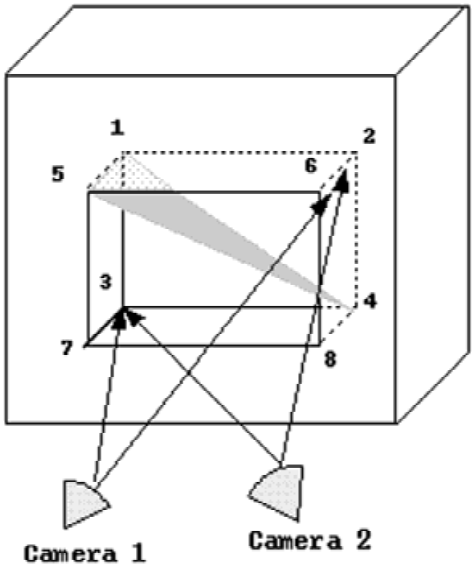



**Fig. 13.** Some exterior triangles might not get detected in the first stage of the algorithm.

especially if we are dealing with small triangles, to recover the shape of the object. Even though it cannot guarantee the removal of all the exterior triangles, an alternative approach exists to improve the shape recovery process: working with tetrahedra instead of triangles. For example, in Figure 13, even though the triangle *t*, formed by the points 1, 4, and 5, is not intersected by any VL, those lines intersect other triangles, adjacent to *t* in the Delaunay triangulation, that are faces of the same tetrahedra as *t*. Such triangles can be the ones formed by points 4, 5, and 6 and 2, 4, and 5 (these triangles are not drawn in Figure 13, for reasons of simplicity).

If one tetrahedron contains at least one triangle that has been previously marked as redundant, it is removed from the triangulation.

### 4.4. Removing hidden triangles

The first two steps of the shape recovery procedure are sufficient for providing the final shape of the object, but they are not able to remove unnecessary triangles located in the interior of the object. These triangles could cause a computational overhead especially for VR applications, where rendering speed is of utmost importance. We do not really need the interior triangles for most of the applications. All that is needed is a suitable representation of the surface of an object. Therefore, the interior (hidden) triangles should be removed.

The removal of hidden triangles is conceptually simple. At this point, we have eliminated the triangles located in the exterior of the object, so we are left with a collection of triangles that best preserves the shape of the object. Intuitively, we can imagine that a triangle that belongs to the surface of the object can be a facet of only one tetrahedron of the triangulation. On the other hand, the hidden triangles are common faces of two tetrahedra. Therefore, the triangles that, after removal of the exterior triangles, are common faces of two tetrahedra are considered interior and therefore eliminated from the triangulation.

After removal of exterior and interior triangles, we are left with a collection of triangles that represent the object's surface.

### 4.5. Correcting face normals

After all the redundant triangles (exterior and interior) have been removed, the surface of the object has to be properly prepared for visualization in VR applications. This entails the computation of the normals to the triangular facets for proper visualization. Typically, in VR applications, Inventor™ or Performer™ compilers are used to view 3D models. For the VR simulation, Performer format for the CAVE™ (Cruz-Neira et al., 1993) is used. All these visualization techniques require calculation of the normal to the surface. In our technique, surfaces are created from scattered point sets, and initially directions of the normals are not known.

After retrieving the surface of the object, the normals to the triangular facets are set to point towards the positive half-space of each triangle, as stored by the Delaunay triangulation algorithm. The problem with this approach is that Delaunay triangulation stores the vertices of each triangle to serve the visibility purposes as internally required by the algorithm. This order might not be suitable for visualization purposes. It is well known from computer graphics (e.g., see Foley, 1995) that a surface patch is visible from a particular viewpoint if the normal to the patch points towards

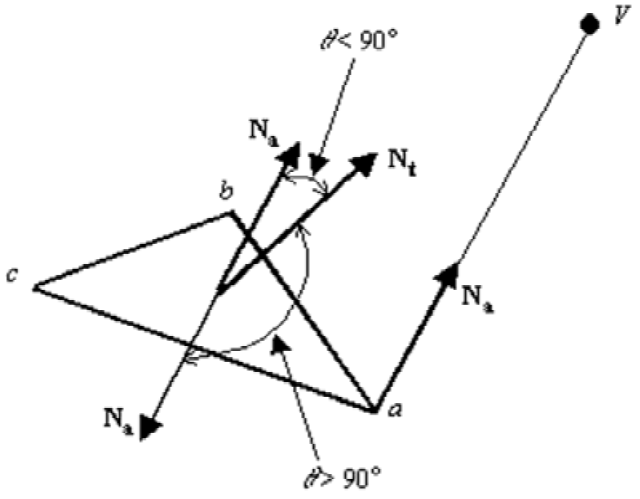



**Fig. 14.** Correcting face normal.

https://doi.org/10.1017/S0890060400145020 Published online by Cambridge University Press

**Table 1.** *CPU cycle distribution for the modules of the shape recovery algorithm*

| | Time [s] | | | | |
|---|---|---|---|---|---|
| Number of viewpoints | Delaunay triangulation | Check for redundant triangles | Removal of redundant tetrahedra | Removal of hidden triangles | Correcting face normals |
| 7 | 0.641 | 0.5 | 0.0719 | 0.05 | 0.1 |
| 10 | 0.641 | 0.635 | 0.072 | 0.057 | 0.11 |
| 15 | 0.641 | 0.79 | 0.083 | 0.055 | 0.12 |
| 23 | 0.641 | 1.1 | 0.08 | 0.062 | 0.12 |
| Percentage CPU time | 39 | 45 | 4.8 | 3.4 | 7.8 |

the viewpoint. As a result of the shape recovery algorithm, this might not be the case with all facets and all available viewpoints (the viewpoints are the camera locations when capturing images of the scene).

To correct the visualization problem, the following algorithm is designed. Consider the same viewpoints as the ones used for shape recovery and consider all triangular facets of the object surface (already known) visible from each viewpoint (the triangular facets that are visible from a particular viewpoint are defined by points visible from that viewpoint). For each viewpoint, we check whether the normal to each visible triangular facet points towards the viewpoint.

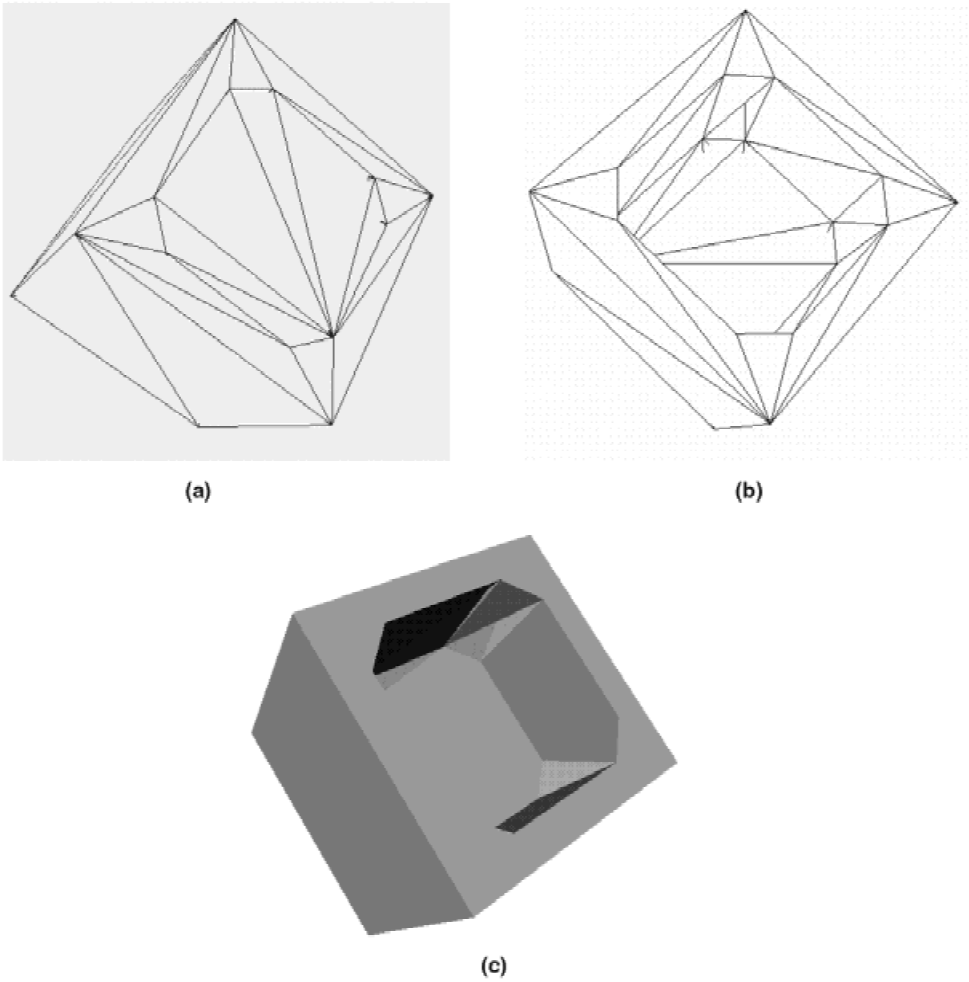



**Fig. 15.** Cube with one cavity recovered with our procedure. (a) Delaunay triangulation of the convex hull; (b) Delaunay triangulation of the convex shape; (c) Recovered shape (four viewpoints).

https://doi.org/10.1017/S0890060400145020 Published online by Cambridge University Press

This is done as follows (the illustration is provided in Fig. 14). Consider the triangular facet defined by the points *a*, *b*, and *c* and a viewpoint *V* (Fig. 14). Suppose that the normal to the triangle $t = abc$ (denoted $\mathbf{N_t}$) points towards the positive half-space of *t*. To check whether $\mathbf{N_t}$ points towards *V*, we simply check whether the angle ($\theta$) defined by the positive direction of $\mathbf{N_t}$ and the direction vector ($\mathbf{N_a}$) of the line that joins *V* with an arbitrary vertex of *t* (say, *a*) is less than 90°. The vector $\mathbf{N}_a$ points from *a* to *V*. If $\theta < 90°$, then *t* is visible from *V* in any VR modeling software. Otherwise, the direction of $\mathbf{N_t}$ should be reversed. The pseudocode of the above-mentioned algorithm looks as follows:

```
For( all viewpoints V_i)
  For( all surface triangular facets t_i
      visible from V_i)
  {
  • define triangle normal N_t
  • define direction vector N_a from ver-
    tex a of t_i to V
  • if the angle θ between N_t and N_a is
    greater than 90°
    —revert the direction of N_t
  }
```

### 4.6. Performance and complexity analysis

Our shape recovery algorithm has five major components that can have an impact on the overall CPU time required by the algorithm. These components are:

1. Delaunay triangulation;
2. Checking for redundant triangles;
3. Removing redundant tetrahedra;

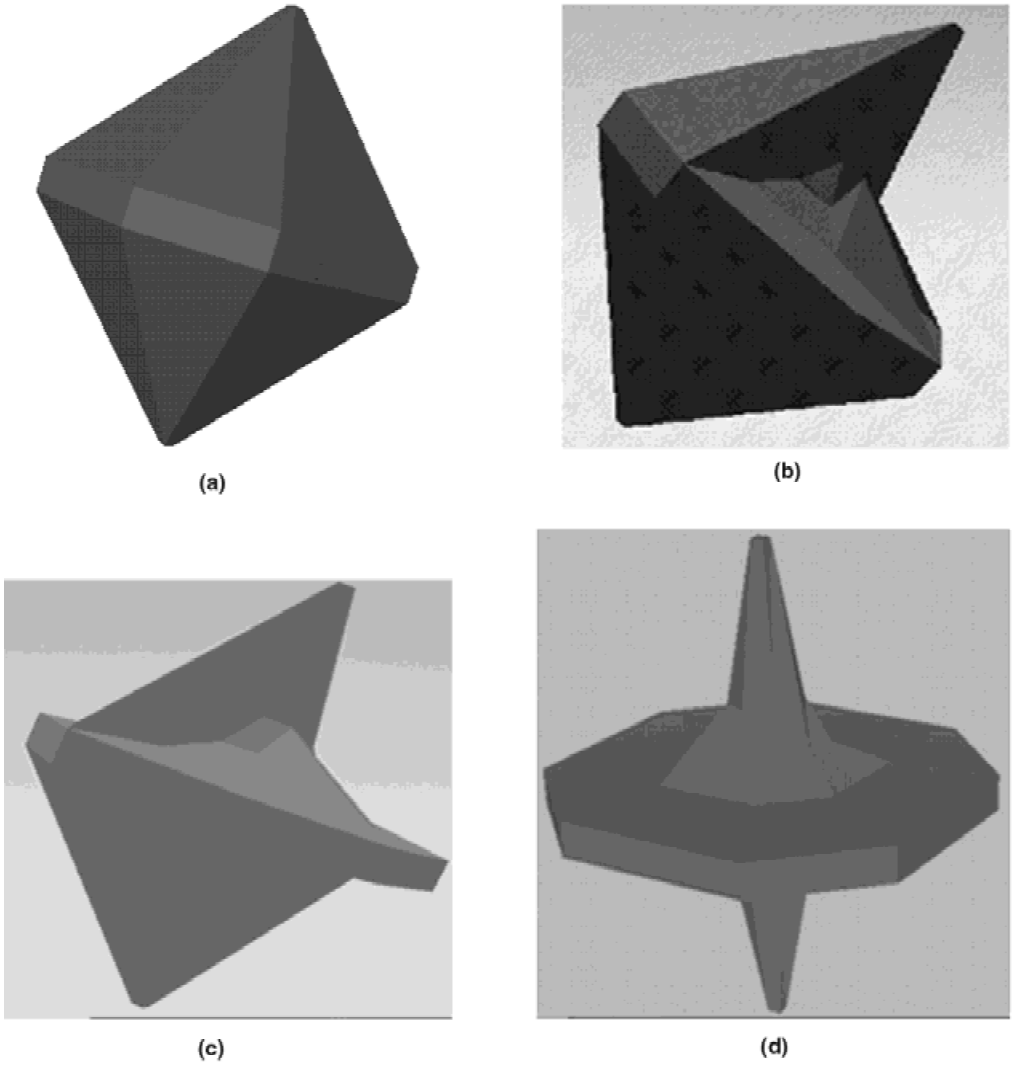



**Fig. 16.** Synthetic object recovered with our procedure. (a) Convex hull; (b) Intermediate shape from 3 viewpoints; (c) Intermediate shape from 6 viewpoints; (d) Final shape (12 viewpoints).

https://doi.org/10.1017/S0890060400145020 Published online by Cambridge University Press

4. Removing hidden triangles;
5. Correcting normals.

Delaunay triangulation performance is dependent on the number of points in the set and independent from the number of viewpoints. The other operations (2–5 above) depend on both the number of vertices and the number of viewpoints. We will analyze the CPU cycle distribution only as a function of the number of viewpoints (it is anticipated that, as the number of points in the set increases, CPU time increases as well). The analyses below have been performed on a 200 MHz SGI O2 workstation, running IRIX 6.5, with 256 Mb of main memory. The number of points in the set is 29. The times required by each operation, when the number of viewpoints varies, and CPU cycle distribution are given in Table 1.

From Table 1, we can see that most of the CPU time is taken by Delaunay triangulation and the detection of redundant triangles. When the number of viewpoints increases, CPU time and percentage of CPU cycle used by redundant triangle check increases also.

The computational complexity of our algorithm depends on the number $n$ of points in the set and the number $p$ of viewpoints used for shape recovery. The complexity of the Incremental Flip algorithm for Delaunay triangulation is $O(n^2)$ (Edelsbrunner & Shah, 1992; Mucke, 1993). The expected number of triangles created by Delaunay triangulation is at most $9n + 1$ (de Berg, 1997). The worst case scenario for our algorithm is when all triangles are visible from every viewpoint (although this never happens in practice). The point-in-triangle testing is being performed in $O(1)$ time. Therefore, the process of marking redundant triangles

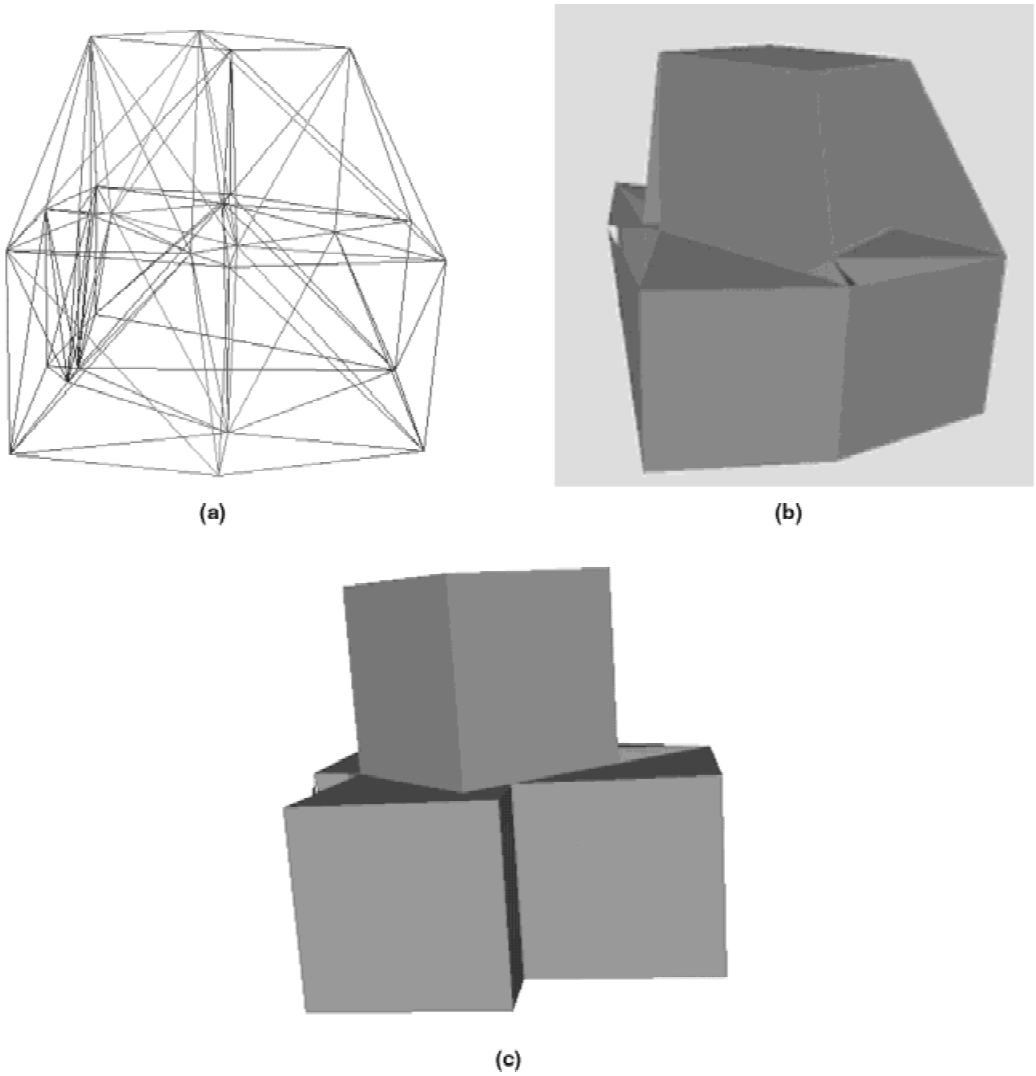



**Fig. 17.** Object from Figure 3 recovered with our methodology. (a) Delaunay triangulation; (b) Intermediate shape from 4 viewpoints; (c) Final shape (7 viewpoints).

https://doi.org/10.1017/S0890060400145020 Published online by Cambridge University Press

has a worst case complexity of $O((9n + 1)p) = O(np + p)$. The other operations of the algorithm are performed in constant time. Consequently, the total computational complexity of our shape recovery algorithm is $O(n^2 + np + p)$. We can see that the complexity of the algorithm is dominated by the complexity of the Delaunay triangulation for small $p$. Since typically $p < n$, our algorithm can be performed in a reasonable amount of time.

## 5. EXAMPLES

In this section, a few examples of shapes recovered with the algorithm described in this paper are provided. Figures 15 and 16 show two artificially generated objects whose shapes have been recovered by applying our shape recovery methodology. The object shown in Figure 16 has also been used by Boissonnat (1984). The same methodology is applied on two real objects, shown in Figures 17 and 18. The object in Figure 17 is the same as the one shown in Figure 3. It can be seen that, unlike Boissonnat's technique or $\alpha$-shapes, our method can accurately recover the shape of this object, provided a sufficient number of images (and, implicitly, viewpoints) is available.

The examples provided in this section show that our methodology can handle a wider variety of shapes (in particular shapes recovered from nondense sets of points) than current techniques, by using specific information derived from the methodology from which the sets of points were obtained (i.e., extracted from 2D images).

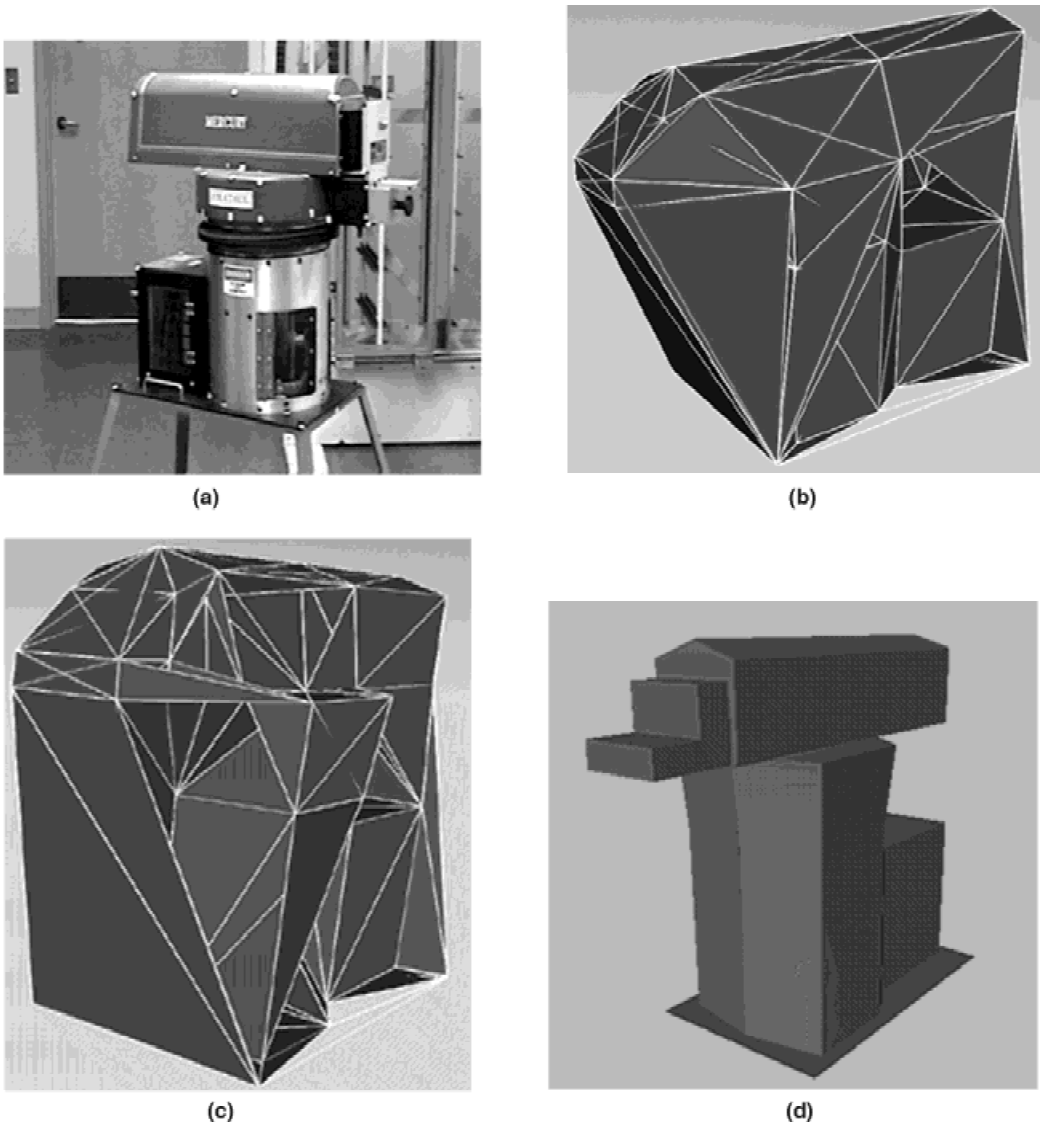



**Fig. 18.** Another object recovered with our methodology. (a) Original object; (b) Intermediate shape from 1 viewpoint; (c) Intermediate shape from 7 viewpoints; (d) Final shape (17 viewpoints).

https://doi.org/10.1017/S0890060400145020 Published online by Cambridge University Press

## 6. CONCLUSIONS

In this article, an algorithm for shape recovery from a scattered set of 3D points, extracted from camera images, has been described. The algorithm exploits the information provided by the image capture process (namely, the location of viewpoints) to carve the convex hull of the point set in order to recover the shape of the object the points belong to. The locations of the viewpoints are used to define VL from the viewpoints to the points of the given set, visible from any viewpoint. The triangles that obstruct the visibility of points from any viewpoint are eliminated from the triangulation of the point set.

The viewpoint-based technique for shape recovery has the advantage of being able to handle well nondense sets of points and objects with small cavities. The only limitation of the technique is that it is dependent on the number of images (viewpoints) provided by the image capture process. A future research direction would be to artificially generate additional viewpoints and automatically determine which points of the set are visible from the artificially generated viewpoints.

## REFERENCES


Boissonnat, J.D. (1984). Geometric structures for three-dimensional shape representation. *ACM Trans. on Graphics 3(4)*, 266–286.

Chen, Y., & Medioni, G. (1992). Object modeling by registration of multiple range images. *Image and Vision Computing 10(3)*, 145–155.

Cruz-Neira, C., Sandin, D.J., & DeFanti, T.A. (1993). Surround-screen projection-based virtual reality: The design and implementation of the CAVE. *ACM SIGGRAPH 93 Proceedings*, 313–319.

Curless, B., & Levoy, M. (1996). A volumetric method for building complex models from range images. *ACM Computer Graphics, SIGGRAPH '96*, 303–312.

de Berg, M. (1997). *Computational Geometry: Algorithms and Applications*. Springer-Verlag, New York.

Edelsbrunner, H., & Mucke, E.P. (1994). Three-dimensional alpha-shapes. *ACM Trans. on Graphics 13(1)*, 43–72.

Edelsbrunner, H., & Shah, N.R. (1992). Incremental topological flipping works for regular triangulations. *Proc. Eighth Annual Symp. Computational Geometry*, 43–52.

Foley, J.D. (1995). *Computer Graphics; Principles and Practice*. Addison Wesley, Reading, MA.

Glassner, A. (1989). *An Introduction to Ray Tracing*. Academic Press, New York.

Higuchi, K., Hebert, M., & Ikeuchi, K.(1995). Building 3D models from unregistered range images. *Graphical Models and Image Processing 57(4)*, 315–333.

Hope, H., DeRose, T., Duchamp, T., McDonald, J., & Stuetzle, W. (1992). Surface reconstruction from unorganized points. *ACM Computer Graphics, SIGGRAPH '92 26*, 71–78.

Mucke, E.P. (1993). Shapes and implementations in three-dimensional geometry. PhD Thesis, University of Illinois at Urbana-Champaign, Urbana, Illinois.

O'Rourke, J. (1994). *Computational Geometry in C*. Cambridge University Press, New York.

Preparata, F.P., & Shamos, M.I. (1985). *Computational Geometry—An Introduction*. Springer-Verlag, New York.

Soucy, M., & Laurendeau, D. (1995). A general surface approach to the integration of a set of range views. *IEEE Transactions on Pattern Analysis and Machine Intelligence 17(4)*, 344–358.

Turk, G. & Levoy, M. (1994). Zippered polygon meshes from range images. *Proc. SIGGRAPH '94 Computer Graphics Conference*, ACM SIGGRAPH, 311–318.

Zetu, D., Banerjee, P., & Schneider, P. (1998). Data input model for virtual reality-aided facility layout design. *IIE Transactions 30(7)*, 597–620.

Zetu, D., Banerjee, P., & Thompson, D. (2000). Extended-range hybrid tracker and applications to motion and camera tracking in manufacturing systems. *IEEE Transactions on Robotics and Automation 16(3)*, 281–293.


---


**Ali Akgunduz** is a Research Assistant and Ph.D. candidate in Industrial Engineering and Operations Research at the University of Illinois at Chicago. He holds a B.S. in Industrial Engineering from Gazi University, Turkey (1992) and an M.B.A from Illinois Institute of Technology (1996).

**Dan Zetu** is currently a Research Engineer at General Motors Technical Center in Warren, MI. He received his Ph.D. degree in Industrial Engineering from the University of Illinois at Chicago in 2000 and a B.Sc. in Mechanical Engineering from Technical University of Iasi, Romania, in 1993. His primary research interests are in customer-driven design and engineering, computer vision, computational geometry and their applications to virtual manufacturing, virtual reality-based factory design, and motion capture from virtual environment.

**Prashant Banerjee** is currently an Associate Professor in the Department of Mechanical Engineering at UIC, serving as the Director of Industrial Virtual Reality Institute (IVRI), a joint research and development operation comprising UIC, Northwestern University and Argonne National Laboratory. Banerjee's current research interests include virtual reality-based factory design, part design and assembly design models, immersive display interfaces, and linear and nonlinear design optimization models. His research has been supported by NSF, NIST and ONR, and by companies such as Caterpillar, Searle and Motorola. Professional education includes an M.S. and Ph.D. from Purdue University, and a B.Tech. from Indian Institute of Technology, Kanpur. In addition to a number of publications in his area of research expertise, Banerjee is a senior member of IIE and is currently serving as department editor of *IIE Transactions*, and as an associate editor of *IEEE Transactions on Robotics and Automation*.


https://doi.org/10.1017/S0890060400145020 Published online by Cambridge University Press