Retrieval of shape from silhouettes

Author's version

Published in: Advances in Imaging and Electron Physics, Volume 139, 2006, Pages 1-73

Retrieval of shape from silhouette

Andrea Bottino, Aldo Laurentini

Dipartimento di Automatica e Informatica

Politecnico di Torino

Corso Duca degli Abruzzi, 24. 10129, Torino, Italy

andrea.bottino@polito.it, aldo.laurentini@polito.it

I. Introduction

A. Reconstructing 3D shapes: general techniques

The reconstruction of three-dimensional shapes is a fundamental research field in computer vision. A large number of techniques have been discussed and implemented, depending first on the type of sensors used. Active 3D scanning is currently a leading technology. Most active 3D scanners use ranging techniques (time of flight, phase comparison) or triangulation with laser beams. Their main feature is producing depth maps that are close to the final 3D shape (Figure 1).

Figure 1: 3D scanners

However, 3D scanners are expensive, and affected by several limitations. They are rather invasive, different equipments are required for scanning objects of different sizes, scanning outdoor large scenes presents several problems, scanning times are not adequate to real time applications, as capturing the shape of objects in motion.

For these reasons, in several practical situations traditional image-based computer vision approaches are more effective. For instance, an important emerging area usually requiring image-based approaches is the analysis of the human body movements (Figure 2).

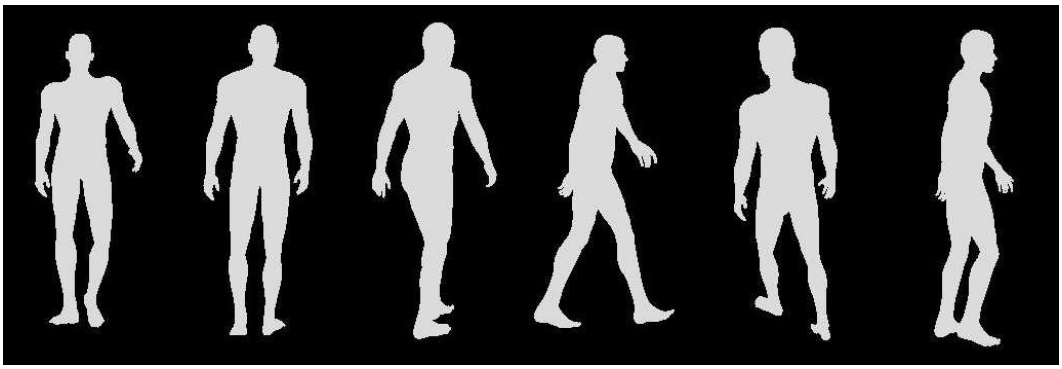

Figure 2: the analysis of posture and motion of the human body is a popular computer vision area

Several passive approaches, based on 2D images captured by inexpensive cameras, have been proposed, usually referred to as *shape from X*, where X stands for some image feature which can be used as 3D shape cue (Aloimonos, 1988). Among them, *stereo* (*disparity*), *texture*, *motion*, *focus*, *shading* and *silhouettes*. An advantage of image-based approaches is that they can easily supply 3D textured reconstructed objects. Actually, most of these approaches require textured objects, or lines called *image contours*.

Some of these approaches, as shape from shading (Zhang and Tsai, 1999), require a number of hypotheses and conditions that seldom take place in practice. The most effective approaches are shape from stereo and shape from contours. Shape from stereo requires images from (at least) two cameras and matching algorithms for determining corresponding points in the two images (Trucco and Verri, 1998). From the different positions in the images of the corresponding points (stereo

disparity), and camera geometry, it is easy to find the 3D position of a point. Shape from motion in principle is an extension of shape from stereo, since it considers multiple images of the same objects taken with the same camera from different relative positions, and also requires finding correspondences in the various images. The purpose of this work is to discuss theory and practice of shape from silhouettes.

B. Shape from silhouettes

Many algorithms for reconstructing 3-D objects from 2-D image features are based on particular lines called image contours. Actually, these lines can be also used for recognizing 3D objects. This approach mimics to some extent the human vision, since we are often able to understand the solid shape of an object from a few image lines laid out on a plane (Gibson, 1951, Koenderink and van Doorn, 1979).

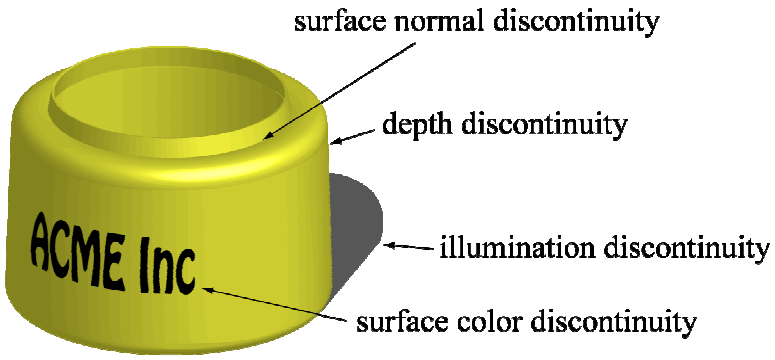

Figure 3: Different types of image lines

In general, from an image of a 3D object we can extract various kinds of lines, separating areas of different intensities or colors. Some of them are not directly related to the 3D shape, and correspond to different surface properties, or to abrupt changes of the incident light. Other lines are directly related to the surface of the object. They are usually called *edges* and *occluding contours* (see Figure 3).

The edges are the projections of the *creases* of the object (surface normal discontinuities), and are not present in images of smooth surface objects. The lines called *occluding contours* (*apparent contours*, *limbs*) are the projection onto the image plane of the *contour generators* of the objects. A contour generator is a locus of points of the object surface where there is a depth discontinuity along the line of sight.

The lines related to the 3D shape (occluding contours and edges) produce line drawings that can be organized into the *aspect graph*, a user-centered representation of 3D objects first proposed in Koenderink and van Doorn (1979).

With the word *silhouette* we indicate the area of the image bounded by the occluding contours that occlude the background. Sometimes the word silhouette is used for the boundary of this area. The idea of using silhouettes to reconstruct 3D shapes was first introduced in the pioneering work of Baumgart (1974). Since then, several variations of the shape from silhouettes methods have been proposed. For example, Martin and Aggarwal (1983) and Kim and Aggarwal (1986) used volumetric descriptions to represent the reconstructed shape. Potemesil (1987), Noborio et al. (1988) and Ahuja and Veenstra (1989) used octree data structure to accelerate the reconstruction process. Shape from silhouettes has also been used in Szeliski (1993) to build a non-invasive 3D digitizer using a turntable and a single camera. Other recent techniques (Kutulakos and Seitz, 2000, Matusik et al., 2001, Slabaugh et al., 2003) stem from this idea.

In principle, reconstruction from silhouette requires back-projecting the silhouettes from the corresponding viewpoint for obtaining solid cones. They are bounded by the circumscribed cones formed by the half-lines tangent to the object. Since the object must lie inside each cone, it must also lie inside their intersection **R**, which summarizes the information provided by all silhouettes and viewpoints (Figure 4). This reconstruction technique is called *volume intersection* (VI)

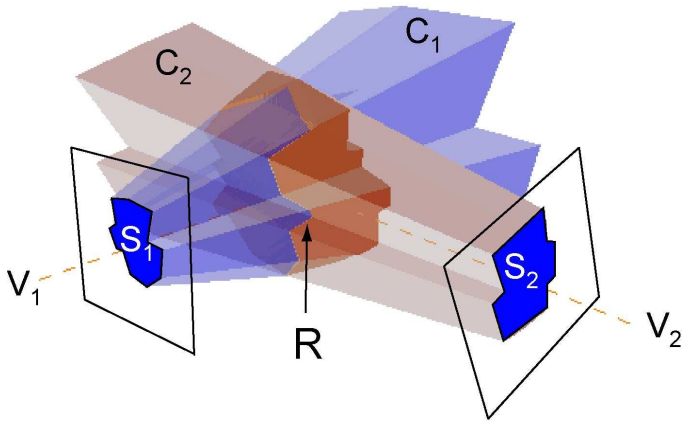

Figure 4: Volume intersection. © 2003 IEEE

A reason that makes silhouette-based techniques popular is that silhouettes are usually easy to extract from images. This operation is particularly fast and robust with controlled background, or for objects in motion with respect to the background. Another important feature of silhouette-based technique is that they are convenient for real time applications.

However, one drawback of this approach is that, depending on the object, the information provided by its silhouettes could be insufficient for fully understanding a 3D concave shape. The problem, qualitatively perceived by several researchers in the field, received a full quantitative solution in Laurentini (1994).

The content of this chapter is as follows. First we discuss the theoretical problems raised by silhouette-based image understanding. In particular, we present and develop the concept of visual hull of an object, which allows answering questions such as: can the 3D shape of an object be fully understood from its silhouettes? We will see that this geometric entity allows answering this and other basic questions related to silhouettes approach. It will be shown that the visual hull and the aspect graph of an object are strictly related. The computation of the visual hull will be discussed for polyhedra, object of revolution and smooth surface objects. In general, VI supplies an object which is not coincident with the object to reconstruct: the problem of inferring the real shape from the object reconstructed will be also discussed, introducing the concepts of *hard* and *soft* points. On the base of these concepts, we will show how an interactive reconstruction approach can be

outlined. Finally we will deal with the problem of reconstructing an object when we have its silhouettes, but no information about the relative positioning of the corresponding viewplanes.

The last part of the chapter is devoted to survey practical shape from silhouettes.

II. The Visual Hull

Reconstructing as well as recognizing 3D objects using silhouettes requires facing some problems inherent to the approach. An intuitive analysis of some examples will help us to focus the problems raised by this technique. Let us consider the concave objects $\mathbf{O}_1$ and $\mathbf{O}_2$ in Figure 5. The difference between them consists of a small cube inside the concavity. It is not difficult to intuitively realize that this cube does not affect any of the silhouettes of the object, provided that the corresponding viewpoint is not “too close” to the object (a precise formulation of this statement will be given later). The example shows that we could be unable to distinguish different concave objects using their silhouettes.

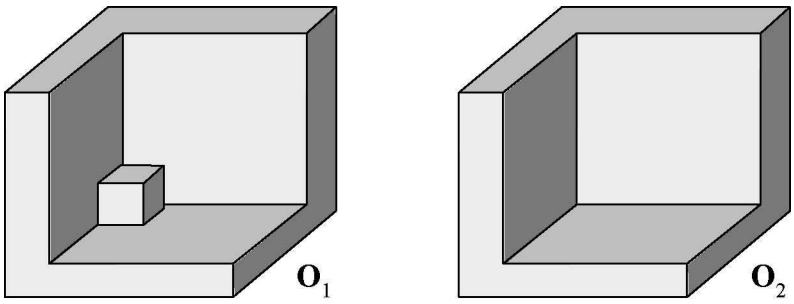

Figure 5: O_1 and O_2 cannot be neither exactly reconstructed nor distinguished using their silhouettes. © 1994 IEEE

Let us consider now the reconstruction from silhouettes of $\mathbf{O}_1$ and $\mathbf{O}_2$. Since the silhouettes relative to viewpoints not ”too close” are equal for the two objects, we cannot reconstruct exactly neither $\mathbf{O}_1$ nor $\mathbf{O}_2$ from these viewpoints. On the contrary, assuming viewpoints equally positioned with respect the two objects, the same object is reconstructed by VI.

The examples show that the silhouette-based approach raises a series of question, such as:

1. can we reconstruct exactly a given 3D object with VI?
2. if not, which is the closest approximation of the object that can be obtained using VI?
3. can two objects be distinguished from their silhouettes?

4. which parts of the surface of an object can affect its silhouettes, and can be reconstructed by VI?
5. how much a concave object can change its shape without affecting its silhouettes?

Before dealing with precise geometric definitions and theorems, it will helpful to consider another example of VI reconstruction. The object in Figure 6(a) is sufficiently simple to allow answering the previous questions concerning reconstruction by an intuitive inspection. We understand that it cannot be exactly reconstructed. The closest possible reconstruction is shown in Figure 6(b). It can be obtained from two ideal viewpoints. Several more viewpoints-at least six-are required if located at finite distance. The surface that can be reconstructed is shown in Figure 6(c). The object can assume any shape inside the polyhedron P without affecting any silhouette obtained by parallel projection, or perspective silhouettes obtained by viewpoints not “too close” to the object Figure 6(d).

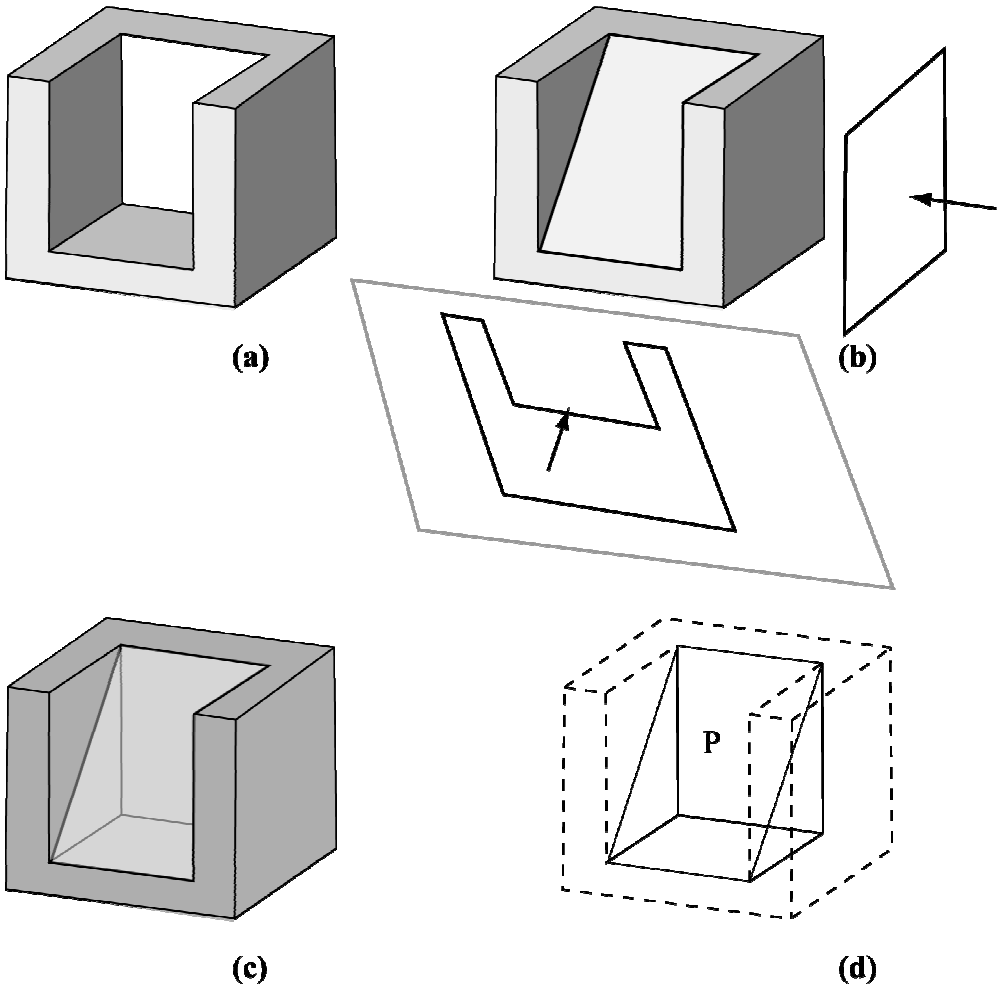

Figure 6: reconstructing an object using parallel projection

In the following we will introduce the geometric concept of *Visual Hull,* which provides precise quantitative answer to the question raised by silhouette-based shape understanding (Laurentini 1994). In the next sections, we will describe the visual hull concept (II.A) and the algorithms for its computation for several classes of objects (II.B). In II.C, we will show what can be inferred of the originating object from the reconstructed object. Finally, two particular problems are discussed: interactive reconstruction of objects, in II.E, and reconstruction with unknown viewpoint, in II.F.

A. Definitions and general properties

Broadly speaking, the visual hull of an object **O** is the maximal object that gives the same silhouettes of **O** from a given set of viewpoints, or the envelope of all the possible cones circumscribed to **O** generated by these viewpoints. We will first define the visual hull relative to a set of viewpoints lying in a particular space region that we call the viewing region. Then we will show that there is a unique visual hull for any viewing region completely enclosing the object and not sharing any point with its convex hull. This will be referred as the *external visual hull*, or *visual hull* without any other specification, since it is relevant to most practical applications. Dropping any restriction for the viewpoint, we obtain the *internal visual hull*, which also provides information on the part of the surface of the objects that cannot be seen by a viewing region outside its convex hull.

1. The Visual Hull of an object relative to a viewing region

Let **VR** be a region of $\boldsymbol{R}^3$ where we can locate viewpoints to observe an object **O** and to extract its silhouettes.

Definition 1: the visual hull **VH**(**O**,**VR**) of an object **O,** relative to **VR,** is a region of $\boldsymbol{R}^3$ such that for any point $\mathbf{p} \in \mathbf{VH}$ and any viewpoint $\mathbf{V} \in \mathbf{VR}$, the half line starting at **V** and passing through **p** contains at least one point belonging to **O**.

A 2D example is shown in Figure 7.

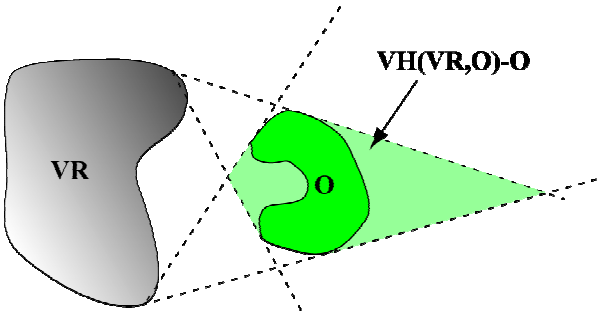

Figure 7: The visual hull of O relative to VR: a 2D example

It is immediate to see that **O** ⊆ **VH**(**O**,**VR**), since any point of **O** satisfies the definition. The following propositions can be obtained from *Definition 1*.

Proposition 1: **VH**(**O**,**VR**) is the maximal object that gives the same silhouette as **O** when observed by any viewpoint **V** belonging to **VR**.

In fact, according to *Definition 1*, the projection of every point **p** ∈ **VH**(**O**,**VR**) from any viewpoint **V** ∈ **VR** belongs to the silhouette of **O** obtained from **V**, as well as the projection of any point **q** belonging to **O**, since **O** ⊆ **VH**(**O**,**VR**). This demonstrates that **VH**(**O**,**VR**) is silhouette-equivalent to **O** for any viewpoint in **VR**. Furthermore, **VH**(**O**,**VR**) is the maximal silhouette-equivalent object, since for any point **p'** not belonging to **VH**(**O**,**VR**) there is at least a line starting from a point **V'** in **VR** and passing through **p'** and not intersecting **O**. Therefore, the projection of **p'** does not belong to the silhouette of **O** obtained from **V'**, and **p'** does not belong to an object silhouette equivalent to **O**. A 2D example is shown in Figure 8.

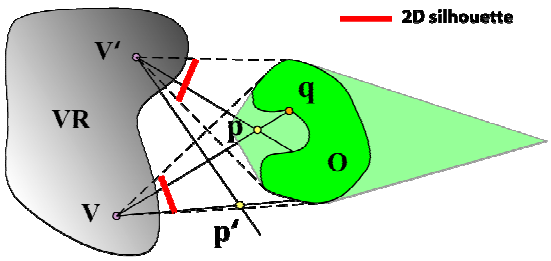

Figure 8: VH(O,VR) is the maximal silhouette-equivalent of O

Proposition 2: **VH**(**O**,**VR**) is the closest approximation that can be obtained using volume intersection techniques with viewpoints **V** belonging to **VR**.

In fact, only points not belonging to **VH**(**O**,**VR**) can be removed intersecting 3D viewing cones, since these are the only points that do not belong to all the silhouettes of **O** that can be obtained from **VR**.

These properties provide quantitative answer to the questions relative to capabilities and limits of silhouette-based techniques:

1. an object **O** can be exactly reconstructed by volume intersection techniques using silhouettes obtained by viewpoints from a viewing region **VR** iff **O** = **VH**(**O**,**VR**)
2. the closest approximation of the object that can be obtained with silhouettes obtained from **VR** is **VH**(**O**,**VR**) (*Proposition 2*)
3. two objects **O** and **O'** can be distinguished using their silhouettes observed from a viewing region **VR** iff **VH**(**O**,**VR**) ≠ **VH**(**O'**,**VR**)
4. the part of surface of the object that affects its silhouettes obtained from **VR** and that can be reconstructed is the part coincident with the surface of the visual hull
5. an object can assume any shape without affecting any silhouette observed from **VR** as far these shape variations are contained in **VH**(**O**,**VR**)-**O**

2. The external Visual Hull

According to *Definition 1*, for every object **O** there is an infinite set of visual hulls, one for each possible viewing region **VR**. This seems to reduce somewhat the effectiveness of the idea. However, in this section we will show that there is a unique object subsuming all the practically important cases, and that this object is related to **CH(O)**, the *convex hull* of **O**.

Observe that, in many practical cases, the object to be recognized or reconstructed can assume any orientation with respect to the viewpoints. Let us therefore consider the visual hull relative to a region **VR'** that *completely enclose* **O**. Let us also suppose that:

$$\mathbf{VR'} \in \mathbf{VR_c} = \boldsymbol{R}^3 \mathbf{- CH(O)}$$

that is, **VR'** is a viewing region outside the convex hull of the object, which appears a practically sensible assumption. The following proposition holds:

Proposition 3: $\mathbf{VH(O,VR')} \subseteq \mathbf{CH(O)}$

To prove this statement, we consider a point $\mathbf{p} \notin \mathbf{CH}$ (**O**), and show that $\mathbf{p} \notin \mathbf{VH}$ (**O, VR'**). We can easily find a viewpoint V such that the line starting at V and passing through **p** does not intersect **O**. A viewpoints of this kind can be found on the plane P passing through **p** and perpendicular to the segment **pq**, where **q** is the point of **CH(O)** closest to **p**. In fact, by contradiction, if the viewline **Vp** intersect **CH(O)** in another point, let's say point **t** in Figure 9, the line **tq** belongs to **CH(O)** and therefore the point of **CH(O)** closest to **p** would be **r** (being **pr** perpendicular to **tq**), and not **q**, contradicting the hypothesis.

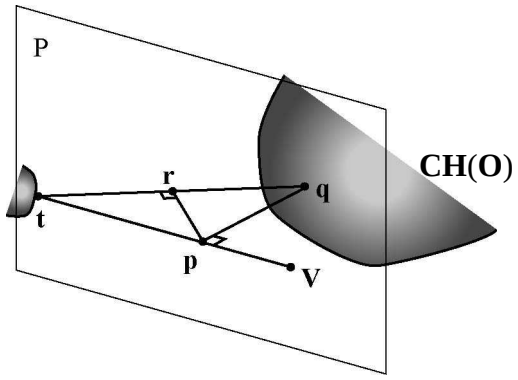

Figure 9: the visual hull relative to viewing regions outside the convex hull of an object O, belongs to CH(O)

Let us consider a second viewing region **VR"**, that, as **VR',** completely encloses **O** and lies outside its convex hull. We have:

Proposition 4: $\mathbf{VH(O,VR')} = \mathbf{VH(O,VR")}$

In fact, consider a point **q**∈**VH**(**O**,**VR'**). According to *Definition 1*, for any viewpoint **V'**∈**VR'** the half line **V'q** intersects **O.** It is clear that also any half line **V"q** , where **V"**∈**VR"** intersects **O** (see Figure 10), since for any **V"** the half line **qV"** intersects the viewing region **VR'.** Then we can find a viewpoint **V'** belonging to the line **qV"**, such that the half line **V"V'q** intersects **O**. Concluding, any point **q** belonging to **VH**(**O**,**VR'**) also belongs to **VH**(**O**,**VR"**). Vice versa, the same argument shows that any point belonging to **VH**(**O**,**VR"**) also belongs to **VH**(**O**,**VR'**), and it follows that the two visual hulls are coincident.

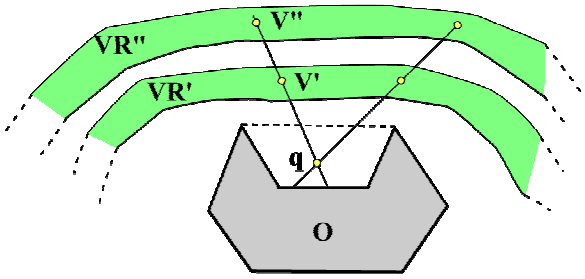

Figure 10: all VHs relative to VRs completely enclosing O and not entering CH(O) are equal

Observe that *Proposition 4* could not hold if a viewing region intersects the convex hull of the object. Consider in fact the 2D case shown in Figure 11, where the region **VR'** has a part inside the convex hull of **O**. Also consider the point **p**∈**VH**(**O**,**VR"**). It does not belong to **VH**(**O**,**VR'**), since for viewpoint as **V'** the half line **V'p** does not intersect **O.** The argument of the previous proof requires that **V'** and **V"** lie on the same side of **p,** which happens if both viewing regions do not share points with the convex hull.

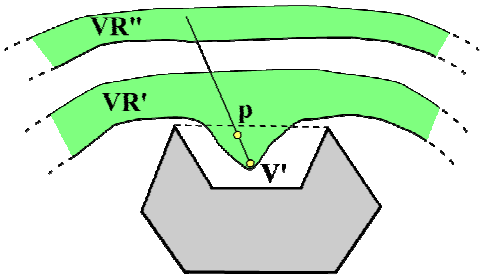

Figure 11: the VHs relative to VR' and VR" are different

In conclusion, there is a unique visual hull, not exceeding the convex hull of **O**, relative to all the viewing regions enclosing **O** and not entering its convex hull. Observe that this precise geometric condition substitutes the intuitive statement "not to near".

Definition 2: **VH**(**O**,$\mathbf{VR_c}$) is the *external visual hull*, or simply the visual hull of **O**, indicated as **VH**(**O**), without any other specification.

It is clear that the (external) visual hull is relevant to most practical situations. The previous answer to the question raised by the silhouette approach can be reformulated in terms of (external) visual hull, making reference to normal viewing regions.

A point **p** belonging to **VH**(**O**) can be characterized in terms of the lines passing through **p**. The following statement holds:

Proposition 5: a point **p** belongs to **VH**(**O**) iff any line passing through **p** contains at least one point of **O**.

This proposition is equivalent to *Definition 1* for the viewing region $\mathbf{VR_c} = \boldsymbol{R}^3$ - **CH(O)**

and it will be used in the following to compute the visual hull. We define *free line* relative to an object **O** a straight line not sharing any point with **O**.

3. The Internal Visual Hull

A limit case takes place when the object itself is the boundary of the viewing region. Let

$$\mathbf{VR_i} = \boldsymbol{R}^3 - \mathbf{O}.$$

Definition 3: **VH**(**O**,$\mathbf{VR_i}$) is the *internal visual hull*, **IVH**(**O**)

The points belonging to the internal visual hull can be simply characterized in terms of half lines:

Proposition 6: a point **p** belongs to **IVH**(**O**) iff any half line passing through **p** contains at least one point of **O**.

This statement, corresponding to *Proposition 5* for the (external) visual hull, is useful to construct the internal visual hull. Finally, since clearly **VH(O,VR)⊆VH(O,VR')** if **VR ⊇VR' and the viewing region relative to IVH (O**) is the largest possible, we have:

Proposition 7: **IVH**(**O**) ⊆ **VH**(**O**)

It also follows that **O** ⊆ **IVH**(**O**) ⊆ **VH**(**O**) ⊆ **CH**(**O**).

For convex objects all these entities are coincident.

The internal visual hull admits another interesting property. It is easy to show that the surface of the internal visual hull not coincident with the surface of the object cannot be observed by viewpoints outside the convex hull. In other words, the **IVH** can tell us which features of a concave object will never appear in any image taken by points not too close to the object.

B. Algorithms for computing the Visual Hull

In general, the surface S_{VH} of the visual hull can be divided into S'_{VH}, coincident with a part of the surface of O, and S''_{VH}, which "covers" some concavities of the object. An example is shown in Figure 12, where **O** is one half of an object of revolution. **CH**(**O**) is its convex hull, where the concavity has been filled. *Proposition 5*, stating that through points not belonging to **VH** must pass free lines, suggests the following intuitive physical construction of **VH**. Fill the concavities of the object with soft material, and scrape off the excess material with a ruler grazing the hard surface of the object in all possible ways. The last image shows the visual hull **VH**(**O**), and in particular S''_{VH}, the surface of the visual hull that covers the concavity, produced by the ruler. A similar intuitive construction holds for the internal visual hull. It consists of "digging" into the concavities with a tool corresponding to a half line, according to *Proposition 6*. In the case of Figure 12, the soft material filling the concavity would be completely scraped off, since **IVH**(**O**) is coincident with **O**.

In the following subsection we will show how the patches composing S''_{VH} can be found for different categories of objects. Then, we will describe algorithms for computing **VH** of 2D objects, polyhedra, solids of revolution, and smooth curved objects.

We will also show that the visual hull is connected to another geometric entity, the aspect graph, and that the visual events surfaces, that we will define in the next sections, can be used to determine the relevant patches for S''_{VH}.

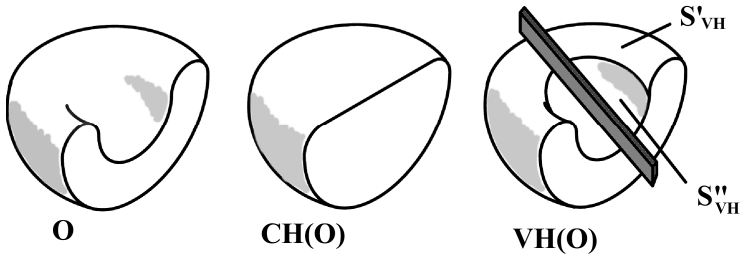

Figure 12: An object O, its convex hull CH(O) and its visual hull VH(O). © 2004 IEEE

1. The Aspect Graph

As already mentioned in the introduction, the contours that occlude both the background and the object itself, together with the image lines which are projections of the *creases* of the object (surface normal discontinuities), produce line drawings that can be organized into the *aspect graph* (Koenderink and van Doorn, 1979). The range of all possible viewpoints can be partitioned into a set of maximal open regions where the topological structure of the line drawing, also called *aspect*, is stable. Crossing the boundaries between these regions produces a topological change in the aspects known as *visual event*. The boundaries are also called the *bifurcation set*. Aspects and visual events can be arranged into the *aspect graph* (AG), where each node is labeled with an aspect and each arc represents a visual event. For the perspective AG, each aspect corresponds to a connected open volume of viewpoints, and each visual event to a boundary surface. For the parallel AG, aspects and visual events correspond to open connected areas and boundary lines on the Gaussian sphere. The parallel aspect graph is a sub-graph of the perspective aspect graph, since not any perspective aspect is also a parallel aspect.

Constructing the AG requires determining the catalogue of the possible visual events and the related boundary surfaces, which is relatively easy for planar faces object (Gigus et al. 1991). Several authors have studied the more complex visual events of curved surfaces. A possible approach is using the *singularity theory* for determining the visual events as the singularities of the *visual mapping*, which maps the surface of the object onto the image plane (Koendrink and van Doorn 1976, Kergosien 1981, Platanova 1984, Callahan and Weiss 1985, Rieger 1987, Rieger 1990,

Petitjean 1996). Other approaches have also been used (Sripradisvarakul and Jain 1989, Eggert and Bowyer 1993). For a comparison of the catalogues presented, the reader is referred to Eggert and Bowyer (1993).

2. The Boundary Surfaces of the Visual Hull

In this section we will derive some necessary condition for a point to belong to $S"_{VH}$, the surface of **VH**(**O**) not coincident with the surface of **O.** In general this surface covers some unreconstructable concavities of **O**, even though in some rather exceptional case there could be parts of **VH**(**O**) not connected to **O (**Laurentini, 1994**).** We will find that $S"_{VH}$ consists of patches of surfaces that belong to the bifurcation set of the aspect graph of **O**.

Let us recall (from *Proposition 6*) that a point **p** does not belong to **VH**(**O**) iff through **p** pass free lines (lines not intersecting **O**). From this statement we can derive a necessary condition for a point to belong to S_{VH}:

Proposition 8 - A necessary condition for a point **p** *to belong to* S_{VH} *is that through* **p** *passes at least one straight line sharing with* **O** *only boundary points.*

This follows from the fact that the visual hull of **O** is the object reconstructed by volume intersection from any possible viewpoint of a region outside the convex hull of **O**. This means that any point of S_{VH} belongs to the surface of (at least) one cone formed by the half lines starting at a viewpoint and tangent to **O**. Then, for finding points of S_{VH} we can restrict ourselves to consider points of straight lines which make contact with the surface S of the object, without intersecting **O** in other parts.

It is clear that, for lines making contact with **O** at one point only, this point belong to S'_{VH}. *Then, for finding* $S"_{VH}$ *we must consider lines making two (*as in the example of Figure 12*) or more*

contacts with S. If we restrict ourselves to generic surfaces without exceptional alignment condition, we need to consider only lines making contact at 3 points. Actually, generic surfaces also admit lines making more than 3 contacts, but these lines are isolated and do not form surfaces (Arnold, 1984, Petitjean et al., 1992). Let us inspect more carefully these lines:

- *Bi-tangent lines*. Straight lines tangent at two different points of **O** do not yield surfaces but fill volumes, so we need a radical pruning of these lines. This will be discussed in the following for two different categories of objects: polyhedra and smooth surface objects.
- *Tri-tangent lines.* In this case the relationship with the aspect graph is immediately established, since tri-tangent lines produce the visual event known as *triple point* (Kergosien 1981, Callahan and Weiss 1985, Petitjean et al. 1992, Eggert and Bowyer 1993, see Figure 13). Lines making three contacts with an object form surfaces that we will discuss in detail for polyhedra and smooth surface objects

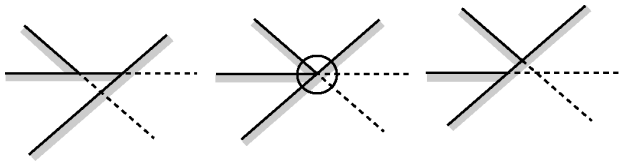

Figure 13: the visual event triple point. © 2004 IEEE

As for the surfaces that bound the internal visual hull, the difference is only that we must consider half-lines making two or three contacts with a generic object.

3. Computing the Visual hull in 2D

The visual hull is essentially a 3D concept. However, algorithm for computing 2D visual hulls are helpful for computing the visual hull of 3D objects obtained by sweeping 2D objects, such as buildings and their interior. In addition, the 2D algorithm for polygonal objects can be used as part of an algorithm for computing the visual hull of polyhedra. For these reasons we will discuss the computation of the **VH** (and briefly of the **IVH**) of polygons in a plane

a) Computing the Visual Hull of a set of polygons.

The **VH** of any simply connected 2D object is trivially coincident with its convex hull, so we will consider a set of polygons SP. It is clear that, to find points which could belong to the boundary of **VH** of generic 2D objects, we must restrict ourselves to lines making two contacts with the object. Therefore, for generic polygons, the possible boundaries of **VH** are the lines passing trough two vertices V_i and V_j and not intersecting elsewhere the polygons.

A line L making two contacts is divided by the vertices into 3 segments. Two cases, shown in Figure 14 can take place. In the first case, Figure 14(a), it is easy to see that through each point **p** of both exterior infinite segments can pass a free line as L', obtained with a small rotation of L about **p**, compatible with the vertices. Then only the interior segment can be a boundary of **VH**. Here and in the following we will call *active* a segment of a line sharing with the object only boundary points, and such that the geometry at the contact point does not allow free lines through the segment.

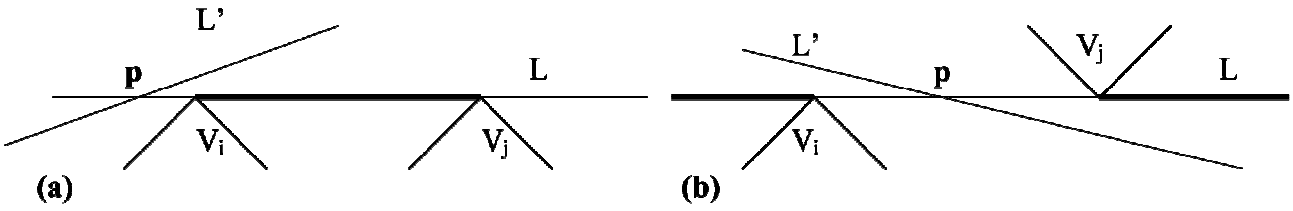

Figure 14: the active segments

In the second case, Figure 14(b), only the exterior segments are active. Observe that, for dealing with exceptional alignment of vertices, the cases of contact at three or more aligned vertices can be decomposed into several two-contact cases, each of which determines active segments. The total active part is obtained as the union of all active segments. Consider for instance the example in Fig.

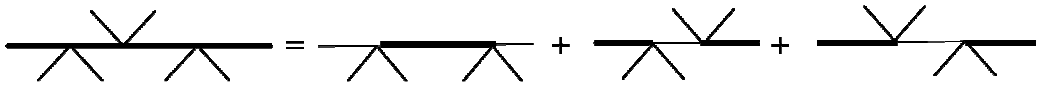

Figure 15: Decomposing a case of three contacts into several two-contact cases

The whole line results active, as it can be verified by OR-ing the active parts due to each pair of vertices.

The general idea of an algorithm for computing **VH** is as follows:

a) find the active segments,

b) intersect these lines to form a partition Π of the plane such that each zone belongs totally or does not belong at all to **VH**.

c) select and merge the zones belonging to **VH**.

Let us consider in more detail each step, and its complexity:

Step a). Let *n* be the number of vertices. The edges are O(*n*). Determining the active segments requires considering O(n^2) pairs of vertices. Each line joining two vertices must be intersected with O(*n*) edges. O(n^2) active segments result in O(n^3) time. Some obvious pruning operations can be performed, as discarding immediately concave vertices, lines that intersect the polygons at the contact vertices, segments outside the convex hull of the polygons that can be computed in O(*n*log*n*) time (Preparata and Samos 1985).

Step b). Observe that, for constructing Π, we must also use the edges of SP that belong to lines not intersecting SP. These edges are also boundaries of **VH**.

Intersecting the active segments to form Π can be performed with the plane sweep algorithm described in Gigus et al. (1991), which makes the computation time sensitive to the size of the output. If *m* is the size of the partition, its construction requires O(*m*log*m*) time. In any case, *m* is O(n^4).

Step c). Efficiently performing this step with a linear visiting algorithm requires introducing the concept of *visual number* of a point VN(**p**). It is defined as the number of families of free lines passing through **p**. Two free lines belong to the same family if one can be obtained from the other with a rotation about **p** without intersecting **O**. For instance, the visual number of point **q** in Figure 16 is 2.

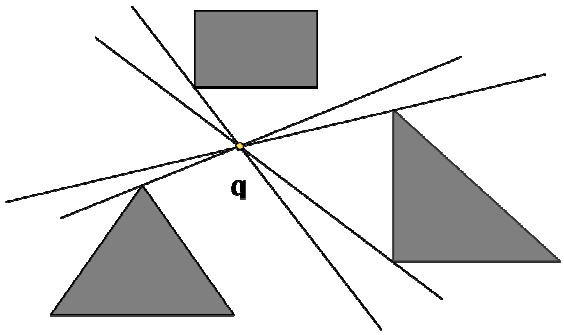

Figure 16: Two families of free lines pass through *q*. © 1994 IEEE

Clearly, the VN of a point belonging to **VH**(**SP**) is zero. The VN is relevant to our case since it is easy to see that all points of each zone of the partition Π have the same visual number. In fact, if we imagine a point **p** moving through the zones of Π, its visual number changes if and only if it crosses an active line. More precisely, one is added or subtracted to visual number according to the crossing direction. In Figure 17 we show the various possible cases.

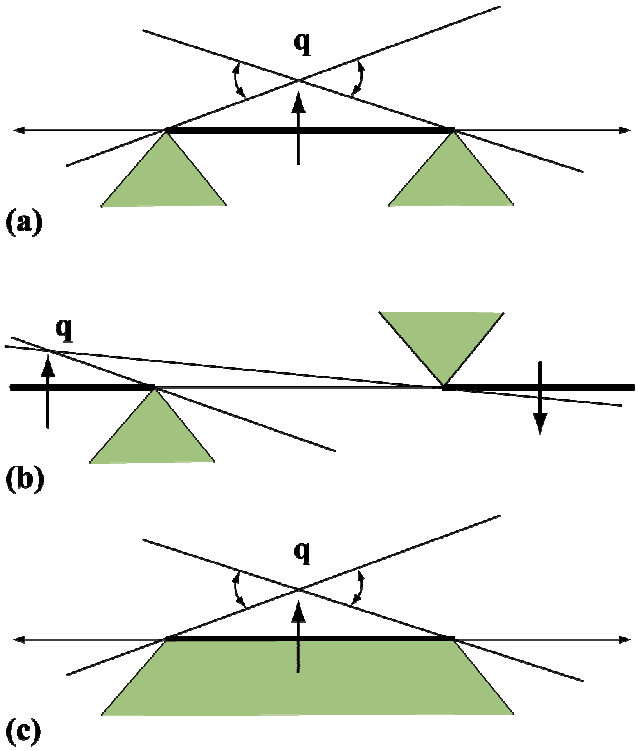

Figure 17: the three cases of active lines and their active segments. © 1994 IEEE

For each case we indicate with an arrow the crossing direction that increases by one the visual number, and the new family of free lines obtained. Observe that the edges lying on lines not intersecting SP change VN from 0 to 1.

Concluding, for selecting the zones belonging to **VH** we need only to visit the dual graph of the partition, computing the VN of each zone. We can start from a zone internal to SP, whose VN is 0. After visiting the dual graph, we can merge the regions whose VN is 0. Step c) can be performed in time O(m), and the whole algorithm takes O(n^3+$m$log$m$) time.

An example, showing the initial set of polygons, the partition Π and the VN of each region, is shown in Figure 18. Observe that there part of **VH** is not connected.

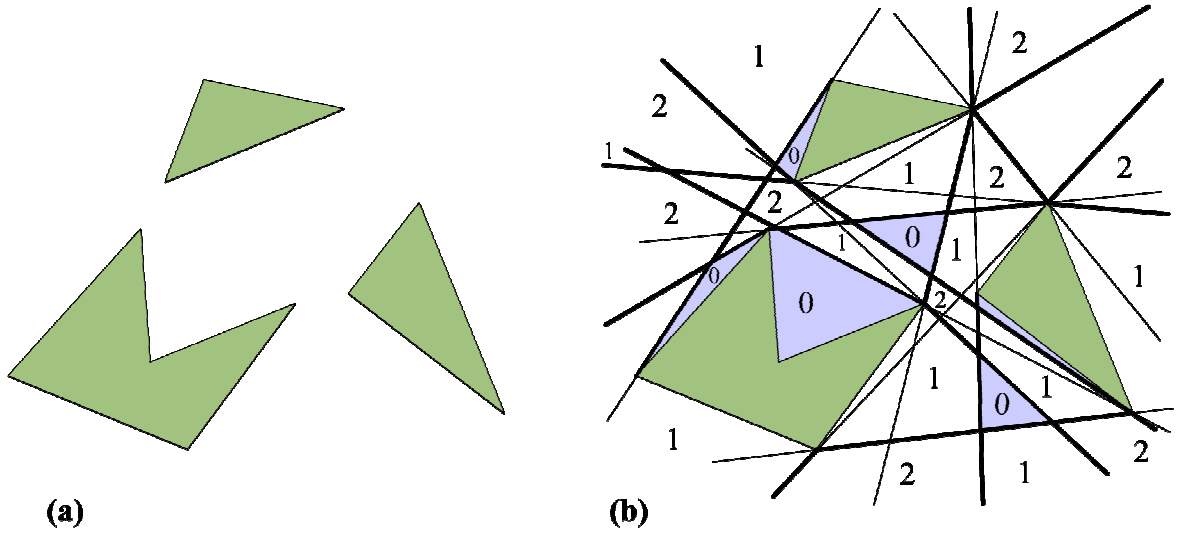

Figure 18: An example that shows the partition and the VNs. © 1994 IEEE

b) Computing IVH(SP)

Here we outline an algorithm for computing **IVH**, similar to the one previously described. The differences are the following. The active lines are those shown in Figure 19. Crossing these lines also adds or subtracts one to the *internal visual number IVN*, defined as the number of families of free half lines starting at a point (see Figure 20).

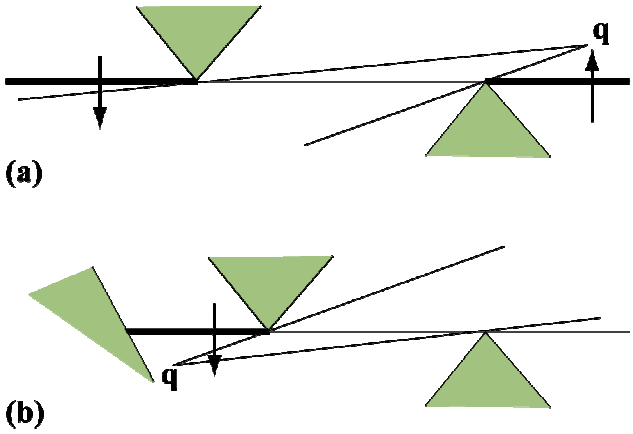

Figure 19: Active lines and active segments for IVH. © 1994 IEEE

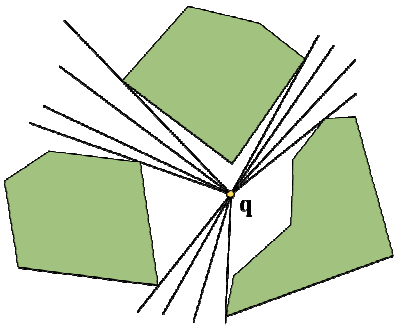

Figure 20: the internal visual number of q is three. © 1994 IEEE

The edges of SP must be also be used for forming the partition, but cannot be crossed visiting the partition, since the IVN can assume any value immediately outside SP. More details can be found in Laurentini (1994). An example is shown in Figure 21 where we show several objects, their **VH** and **IVH**:

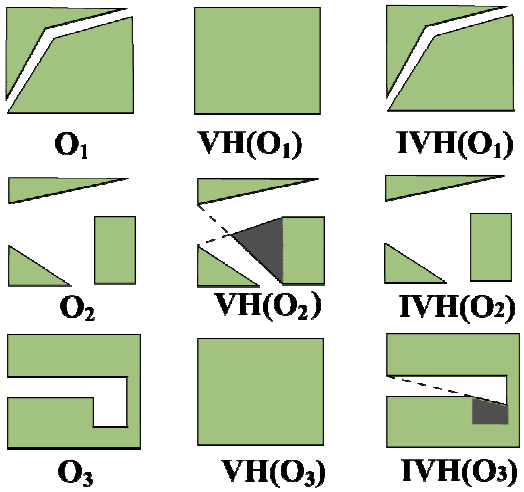

Figure 21: three 2D objects, their visual hull and their internal visual hull

c) Computing the visual hull of curved 2D objects.

The previously described algorithms for **VH** and **IVH** also hold for a set of objects bounded by curved lines, the only difference being that the active lines are those bitangent to the objects.

4. Polyhedral objects

An algorithm for computing the visual hull of polyhedral objects is described in Laurentini (1994). This algorithm uses a brute force approach and takes $O(n^{12})$ time.

A more efficient algorithm has been presented by Petitjean (1998). The algorithm uses a new visibility structure, *the visibility complex* (Durand and Puech, 1995), which is a partition of the free lines according to their visibility with respect to **O**. The visually active surfaces can be computed from the visibility complex, and also the surfaces that intersect the scene transversally can be pruned by traversing the complex. Defining the 3D visual number of a point as the number of different families of free lines passing through **p** (two lines being in the same family if one can be transformed into another by a rotation around **p** without intersecting the object), traversing a visually active surface means incrementing or decrementing the visual number by one. It then suffices to traverse the partition of the viewing space generated by the active surfaces, starting from

some initial position, to identify the cells belonging to the visual hull. The algorithm takes $O((n^4 + m^3) \log n)$ time, where m, the number of active surfaces, is $O(n^3)$ in the worst case.

Here we will propose a new efficient algorithm. The algorithm is able to determine only the parts of **VH** that are connected to the object. However, concave 3D objects able to produce visual hulls with parts not connected are neither likely to be found in practice, nor easy to construct. An example of such an object is shown in Figure 22. The object is multiply connected. With some more ingenuity, the reader can also construct examples where the object is simply connected. Concluding, the algorithm that we will describe in the following is perfectly adequate to most practical cases. Their main feature is that several pruning rules are proposed, able to discard a substantial amount of surfaces candidate to bound the visual hull. After intersecting these surfaces, a very simple rule allows to find cells belonging to **VH**.

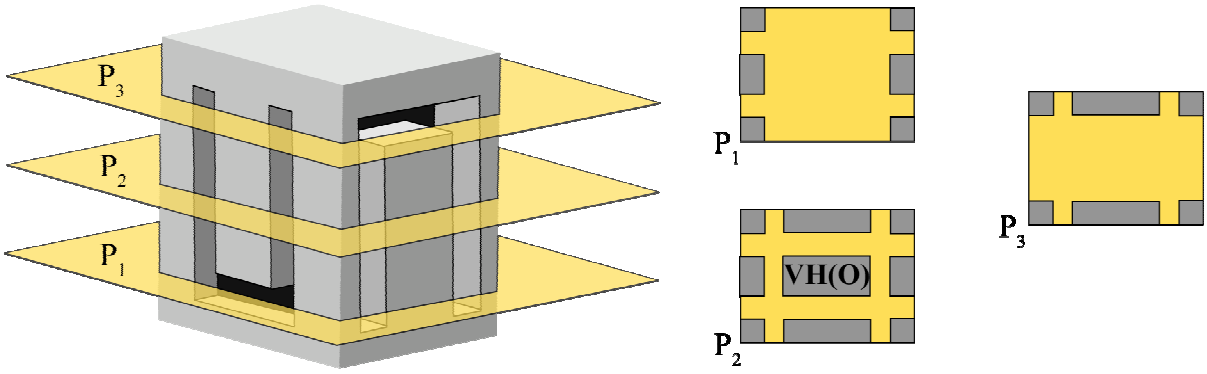

Figure 22: an example where there is a part of VH (O) not connected to O. The sections with planes P_1, P_2 and P_3 are shown on the right

In agreement with the general approach outlined in this work, we consider as candidates to form $S"_{VH}$ points of straight lines making two or three contacts with the object. For understanding if a point of a such line can actually be a point of surface $S"_{VH}$, we will investigate, as already done in 2D, if there are free lines passing through this point and compatible with the surface of **O** at the contact points.

a) Lines making two contacts.

As already observed, lines making two contacts with *two edges* of a polyhedron fill a volume. Let us consider a line L touching **O** at two points $\mathbf{p}_1$ and $\mathbf{p}_2$ of the edges E_1 and E_2 (Figure 23). Let also consider the two planes P and P', containing L, as shown in Figure 23(a), which is a projection of the edges from a viewpoint lying on L. The points $\mathbf{p}_1$ and $\mathbf{p}_2$ divide L into two exterior half-open infinite segments and one open interior segment. Let us consider the intersection of P and **O** (Figure 23(b)). It is clear that the exterior segments cannot contain points belonging to S''_{VH}, since in P there are free line as L', obtained with infinitesimal rotations of L, passing through any point of both segments. Consider now the interior segment. In P there are free lines passing above the segment at an arbitrarily small distance, but no free lines passes through points of the segment. The situation is different in a plane as P', whose intersection with S is shown in Figure 23(c). Through any point of the interior segment pass free lines as L' compatible with the local geometry. Concluding, no point of L belongs to $S''_{VH,}$ or, in other words, lines making contact at two generic edges are *inactive.*

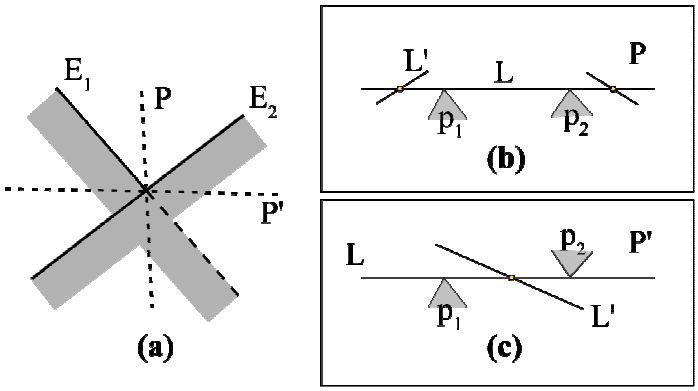

Figure 23: lines touching O at two generic edges are *inactive.*

Now let us consider lines making contact at *a vertex* V *and at a point* **p** *of an edge* E. We call these lines *VE lines*. Let us analyze the projection of V and E from a point lying on L onto a plane normal to L, and suppose first that the projection along L of the edges converging at V is convex. We also admit that E and one of the edges meeting at V are coplanar, since this case is important in practice. Five cases, shown in Figure 24(a), (b), (c), (d), (e) can take place. All these cases are *inactive.*

Actually, for each case, it is shown a plane P that contains L, whose intersection with **O** for all cases is like Figure 24(f), showing that, through any point of L pass free lines compatible with the vertex and the edge.

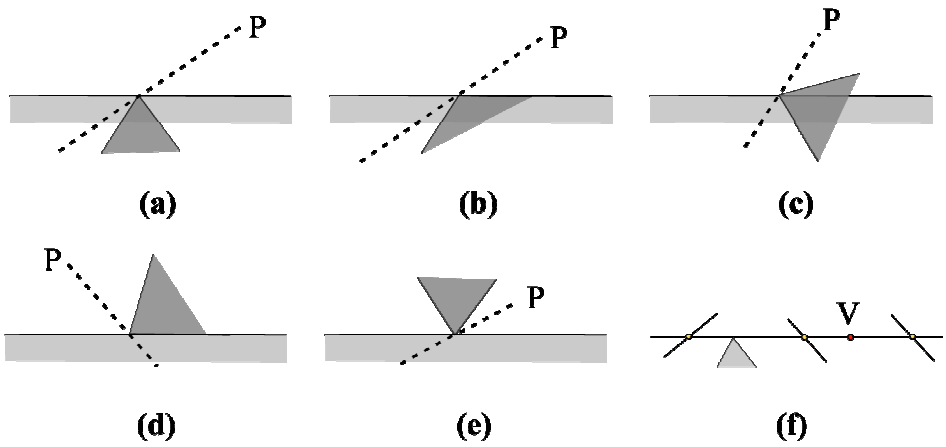

Figure 24: VE lines making contact at a convex vertex are not active

Consider now VE lines making contact at a vertex such that the projection along L of the edges converging at V is concave. Taking into account also coplanar edges, the five cases shown in Figure 25 can occur. For each case, we show two or three planes P, P' and P", representative of the families of planes containing L, and their intersections with the object at V and E. Let us consider case (a). It is clear that for any plane containing L the intersection is either as that of P' or of P". In planes as P' there are free lines passing through the exterior segments. In planes as P" no free line is possible. Combining these results, we find that the interior segment is active.

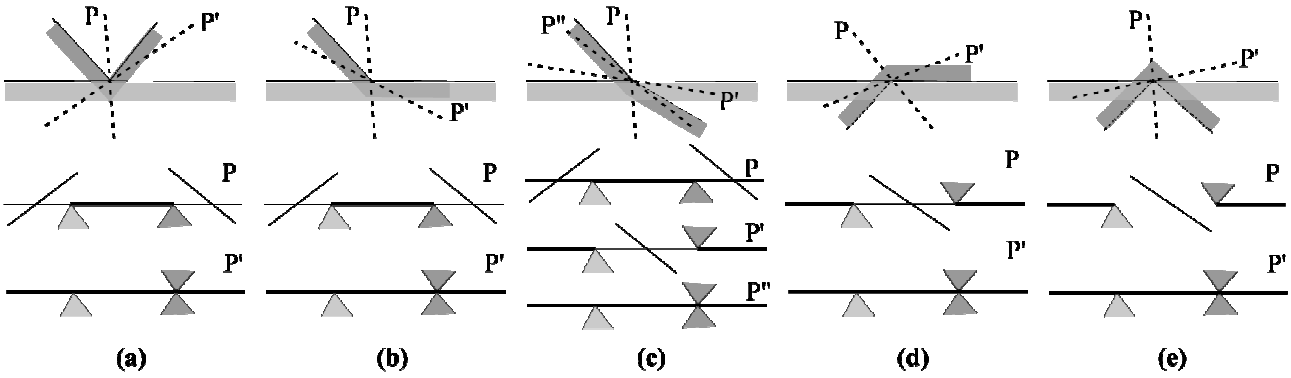

Figure 25: VE lines making contact at a concave vertex. Cases (a), (b) generate an interior active segment. Cases (d), (e) produce two exterior active segments. Case (c) is inactive

Similar arguments show that also in case (b) the interior segment is active, case (c) is inactive; the exterior segments are active in cases (d) and (e). Observe that in case (c) there are three families of planes generating three different kinds of intersection.

The two types of *active VE surface patches* generated by the active exterior or interior segments are shown, respectively, in Figure 26(a) and (b). We recall that the lines making contact at V and E must not intersect **O**, and then can be active only those parts of these surfaces where the generating line does not intersect **O** elsewhere.

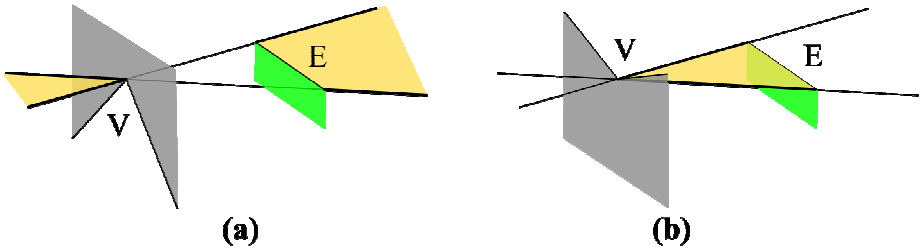

Figure 26: two types of active VE surfaces patches

We also consider another limit VE case, common in practice, where E and the edges converging at V are coplanar. It is easy to see that two cases are possible, generating or an interior active segment, or two active exterior segments (Figure 27). Also observe that cases of multiple contacts at coplanar edges can be decomposed into several cases of two contacts.

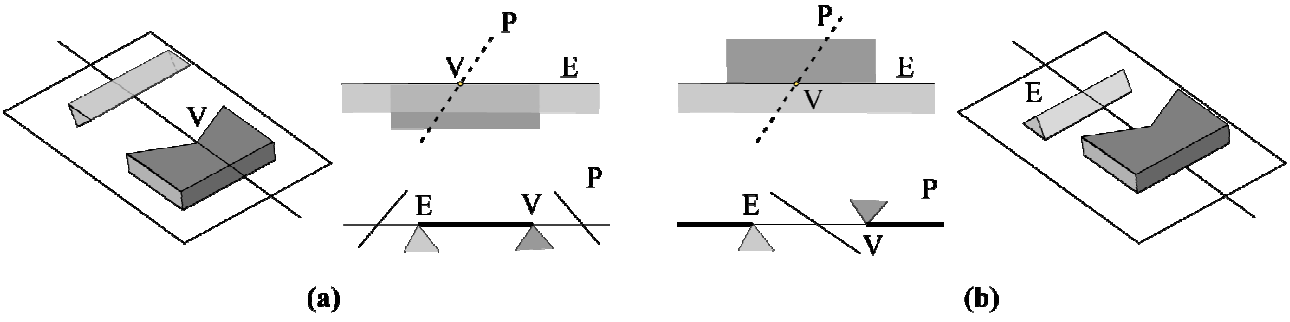

Figure 27: The VE case with three coplanar edges.

Finally, observe that the VE active patches are parts of the VE surfaces considered in the theory of the aspect graphs of polyhedra (Gigus et al. 1991).

b) Lines making three contacts.

We will call EEE this case and the line and surfaces involved. From basic analytic geometry it results that lines passing through three 3D lines skew to each other form quadric ruled surfaces. More precisely, they form a hyperboloid of one sheet or a hyperbolic paraboloid. The second case takes place when the three lines are all parallel to a plane (Salmon, 1874).

Also in the EEE case we analyze the four segments in which the line is divided by the contact points, for finding if there are active segments, which produce active patches. Let E_1, E_2 and E_3 be the three edges and $\mathbf{p}_1$, $\mathbf{p}_2$ and $\mathbf{p}_3$ the three contact points of line L. Obviously, preliminary to this analysis is checking that L shares with **O** the contact points only.

It is easy to verify that three different spatial arrangements of the edges can take place. The cases are shown in Figure 28(a), (b) and (c). Cases (d) and (e) are cases (b) and (c) observed from the opposite side. In the figure we show a view from viewpoints on L, as well as a 3D origami-like structure, intended to clarify the 3D relative position of the edges. The arrows in (b), (c), and (e) indicate the direction of the inner side of the surface at the hidden edge E_3.

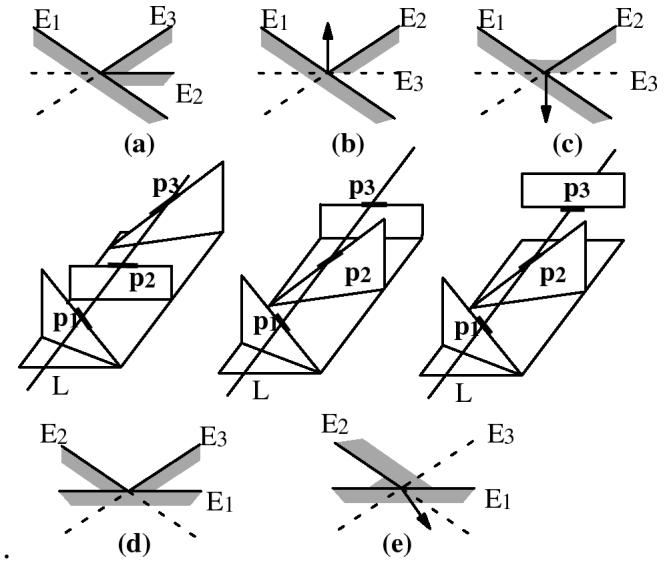

Figure 28: the possible 3D arrangements of the edges for the EEE case. © 2004 IEEE

Each case can be analyzed by studying the intersection with the edges at the contact points with a plane rotating about L. An example of this analysis for case (a) is shown in Figure 28 for case (a).

Three possible position of the plane P_R are possible, marked with **1, 2** and **3** in Figure 28(a). The three intersections, marked ($\mathbf{a_1}$), ($\mathbf{a_2}$) and ($\mathbf{a_3}$), show that the case is not active.

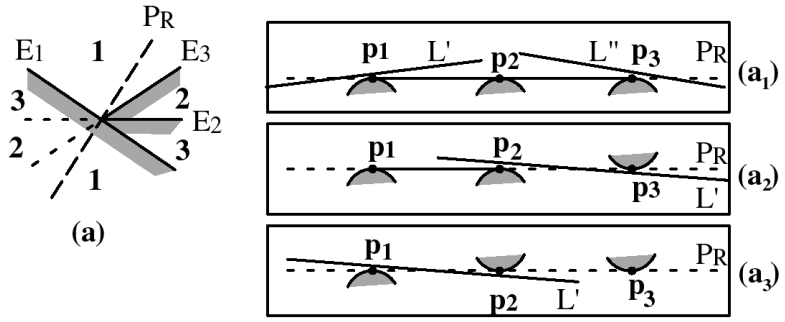

Figure 29: the analysis of case (a) with a rotating plane. © 2004 IEEE

The results obtained are summarized in Figure 30, where the solid lines mark the active segments.

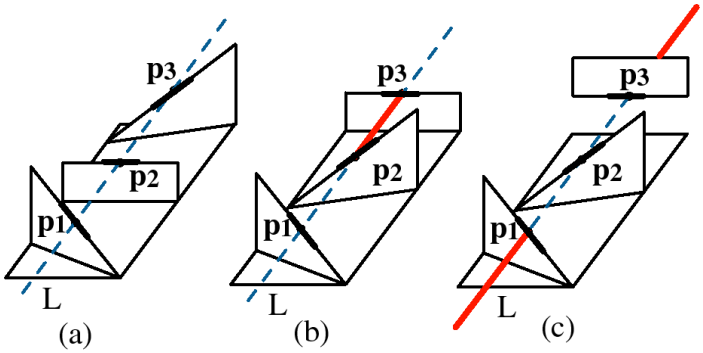

Figure 30: the active segments of tri-tangent lines. © 2004 IEEE

c) The algorithm for computing VH

The outline of the algorithm for computing **VH** is as follows.

a) Compute the surface S'_{VH}, which is the surface of O coincident with the surface of VH(O). If it covers completely **O**, it is **O**=**VH**(**O**), and we stop. Otherwise:

b) Compute the active VE and EEE patches

c) Compute **VH** using a volumetric approach, which intersects the active surfaces and selects the zones of the partition belonging to **VH**(**O**).

In the following we will discuss in detail the various steps, using as a running example the object in Figure 31.

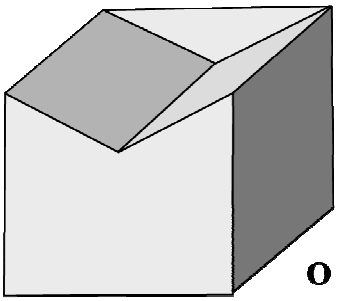

Figure 31: the running example

Step a). For computing S'_{VH} we can use the 2D visual hull algorithm applied in the plane of each face F_i to the set of polygons SP_i obtained intersecting this plane and **O** and excluding F_i itself. It is easy to see that a point of the face belongs to S'_{VH} iff it does not belongs to $\mathbf{VH}(SP_i)$ (for a full discussion see Laurentini, 1994). For each face the complexity of the algorithm is $O(n^3+m\log m)$, where m is the size of the partition (which is $O(n^4)$) and then the whole step is $O(n^5 \log n)$. However, much pruning can be performed. The convex hull of the polyhedron can be computed in $O(n \log n)$ time, and the faces, or their parts, belonging to the convex hull can be immediately selected as members of S'_{VH}. Other pruning operations are described in Laurentini (1994). The result of this step for our toy object is shown in Figure 32.

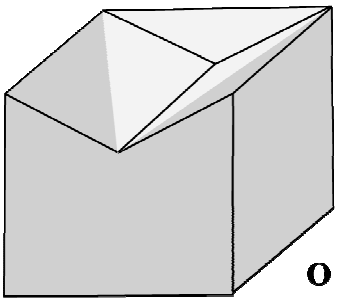

Figure 32: the surface S'_{VH} (light gray)

Step b). In principle, computing *the* active surfaces requires:

1. considering O(n^2) VE pairs and O(n^3) EEE triplets;
2. verifying in constant time if there could be active surfaces compatible with the contact points;
3. intersecting each candidate surface with **O**

In all, it takes O(n^4) time for finding O(n^3) active patches.

Actually, also in this case much pruning can be done, reducing greatly the computational burden. For both VE and EEE surfaces, a large number of cases can be ruled out in constant time. In the following we provide the main pruning rules, without detailing the rather trivial proofs. The rules also consider exceptional alignment cases.

- ***Rule 1.*** Concave edges must not be considered neither for VE nor for EEE surfaces
- ***Rule 2.*** Concave vertices must not be considered for VE surfaces. By definition, a concave vertex V is such that the intersection of any face sharing V as vertex with **O** form a concave angle at V (see Figure 33)

Figure 33: the intersection with O of a face sharing a convex vertex.

- ***Rule 3***. Edges shared by faces lying on CH, at least near the edge, must not be considered neither for VE nor for EEE surfaces, as well as vertices shared by these edges

Applying rules 1 and 2 to the running example deletes the edges and the vertex shown in Fig. (a). Applying rule 3 also deletes the edges and vertices shown in Fig. (b). The edges and vertices left, which are candidate to form VE and EEE active surfaces, are shown in Fig. (c), and are the rim of the concavity of the object.

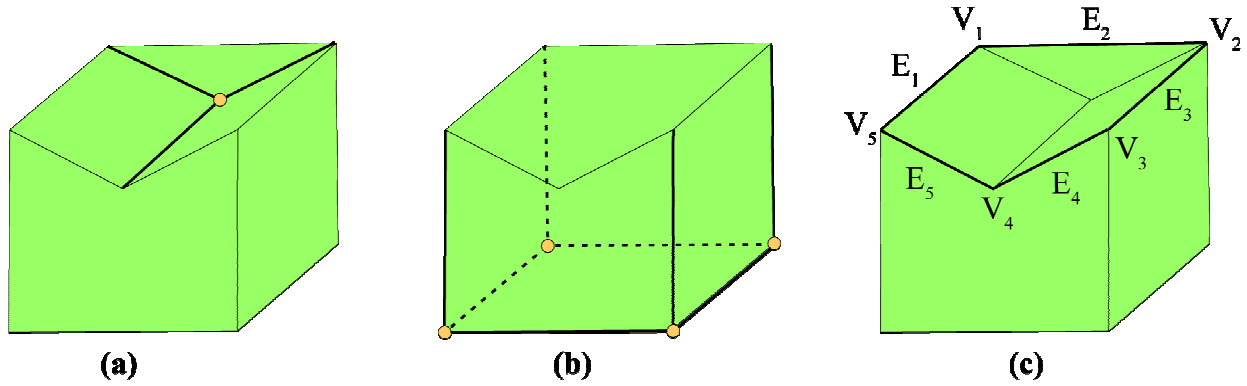

Figure 34: rules 1 and 2 discard edges and vertices highlighted in (a). Edges and vertices discarded according to rule 3 are shown in (b). In (c) edges and vertices candidate to give VE and EEE active patches are shown.

The analysis of the various VE cases generated shows that there is only one active VE surface. The surface $V_3\,E_5$ (Figure 35(a)), as well as the symmetric $V_5\,E_4$, is not active, since they both correspond to case (b) of Figure 24. For the same reason, also $V_3\,E_1$, as well as the symmetric $V_3\,E_2$ (Figure 35(b)), is not active. Observe that, since E_1, E_2 and E_3 are coplanar, (Figure 35(b)), the rectangle V_1, V_2,V_3 and V_5 is covered by other VE patches, as $V_2\,E_1$, that are active. However, the rectangle is inactive since it is covered completely by the two inactive patches $V_3\,E_1$ and $V_3\,E_2$. The patch $V_1\,E_4$, as $V_2\,E_5$, (Figure 35 (c)) is also inactive since it corresponds to case (c) of Figure 24. Concluding, only $V_4\,E_2$ (Figure 35 (d)), corresponding to case (a) of Figure 25, is potentially active. This is actually the case, since the whole VE surface does not intersects O elsewhere. Observe that, given this object with 16 edges and 10 vertices, the pruning operations, which are all performed in constant time, produce only one active VE surface.

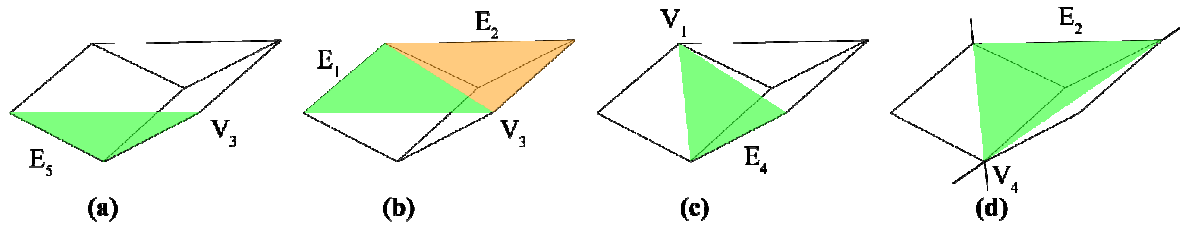

Figure 35: Inactive VE surfaces are shown in (a), (b), (c). The only active VE case is shown in (d).

As for EEE surfaces, only the edges candidate to form VE patches can form them. We can consider all pairs of such edges, prune all coplanar pairs, and, for each remaining pair $E_i E_j$ we must check if one of the remaining edges (or a part of it) is included into the volume formed by all lines passing through E_i and E_j (Figure 36). This takes constant time for each remaining edge. In the running example, the candidate pairs are $E_4 E_1$, $E_4 E_2$, $E_5 E_3$ and $E_5 E_2$, and, for each pair, no other candidate edge is enclosed into the corresponding volume, so that there are no active EEE surfaces.

In general, if one edge or part of edge is enclosed, we must verify if the case corresponds to one of the active cases of Figure 30, and in this case intersect the whole EEE surface with **O.**

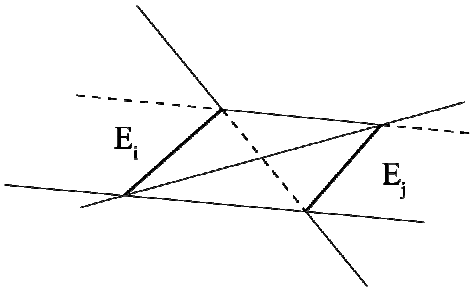

Figure 36

Step c). Let S'(**O**)= S(**O**) - S'_{VH}, that is the part of the unreconstructable surface of **O**. After finding all active VE and EEE patches, we construct a 3D partition Π using these patches and the faces of the surface S'(**O**). For *k* active patches and faces, the partition consists of O (k^3) cells. Note that this result holds for infinite algebraic surfaces: in the current case, with small finite patches (also when the active surface is an exterior segment, we are interested only in the part inside **VH**), the number of cells is much smaller.

Each cell of this partition belongs entirely or does not belong at all to **VH**. If we suppose that the visual hull is connected to **O**, which, as already shown, happens for most objects, selecting the cells of the partition belonging to **VH** can be done in constant time for each cell. In fact:

Proposition 9: if **VH**(**O**) is connected to **O**, only the cells of partition Π that are bounded by one or more faces of S'(**O**) belong to **VH**(**O**).

This can be easily proofed by contradiction. Observe that, for each active patch there is a side, that we will call *positive*, where free lines pass near to the surface. Suppose that there is a cell not bounded by S'(**O**). Then its boundary faces are only patches of active surfaces, whose positive side must lie on the exterior of the cell, otherwise there would exist free lines inside the cell. But this means that on the exterior side of each face pass free lines, and that the cell of **VH**(**O**) is an insulated one**,** against the hypothesis.

In the running example, the partition consists of one cell only that satisfies the condition of *Proposition 9.* The visual hull of the object is shown in Figure 37.

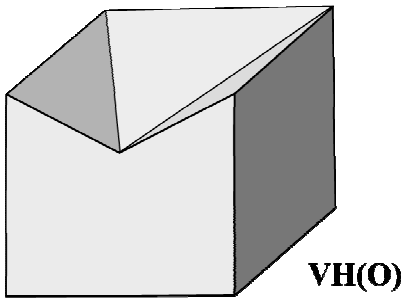

Figure 37: the visual hull of the example.

Let us consider another example. The object is shown in Figure 38(a). Its surface S'_{VH} is shown in Figure 38 (b). After applying the various pruning rules we find the candidate edges and vertices shown in Figure 38 (c). Various overlapping VE patches result. Consider for instance the rectangle V_1, V_2, V_4 and V_5. It includes two active patches (V_1E_5 and V_5E_2) and two inactive patches (V_2E_5 and V_4E_2). Superimposing the various patches, the active patch shown in (d) results. Two other similar VE active patches can be obtained. No EEE active surface results. The partition produced is shown in Fig (e). The only cell produced belongs to **VH** since it satisfies *Proposition 9*.

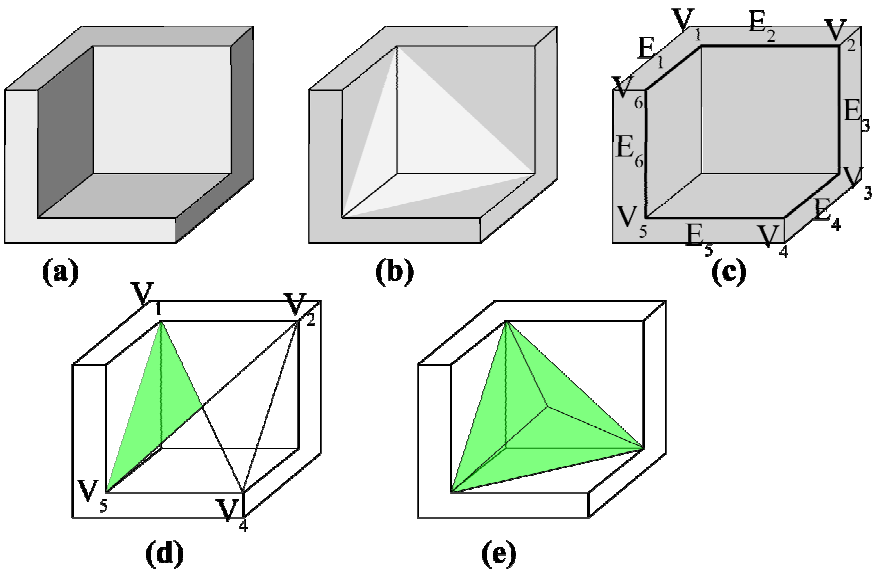

Figure 38: The object (a), the surface S'_{VH} (b), the edges and vertices after the pruning (c), an active VE patch (d), the only cell of the partition belongs to the visual hull (e).

Also this case shows the effectiveness of the pruning rules. The sample object has 21 edges and 14 vertices, but produces only 6 active VE patches, no EEE patches and a single partition with one 6 faces cell.

A more complex object, also able to produce EEE active surfaces, is shown in Figure 39 (a). In (b) we show the surface S'_{VH}, in (c) the vertices and edges surviving the pruning. Several VE surfaces are generated, some of them overlapping, which we will not enumerate. As an example, we show the patches originated by V_1. The patches V_1E_9, V_1E_8 and V_1E_4 are active, V_1E_5 corresponds to case (c) of Figure 25 and then is inactive. However, V_1E_4 is inactive since it is coincident with the

inactive patch V_3E_3 (case (b) of Figure 24). Also for V_1E_9 there are other patches lying in the same plane, but those superimposed are active. Concluding, V_1 actually generates two active patches: V_1E_9 and V_1E_8. The most interesting feature of the object is an EEE patch, adjacent to V_1E_8, formed by the lines making contact at E_3, $E_8$and E_5 (Figure 39 (e)). Using Figure 30 it results that the active part is between E_3 and E_8. Also in this case the partition includes only one cell belonging to **VH** (Figure 39 (f)).

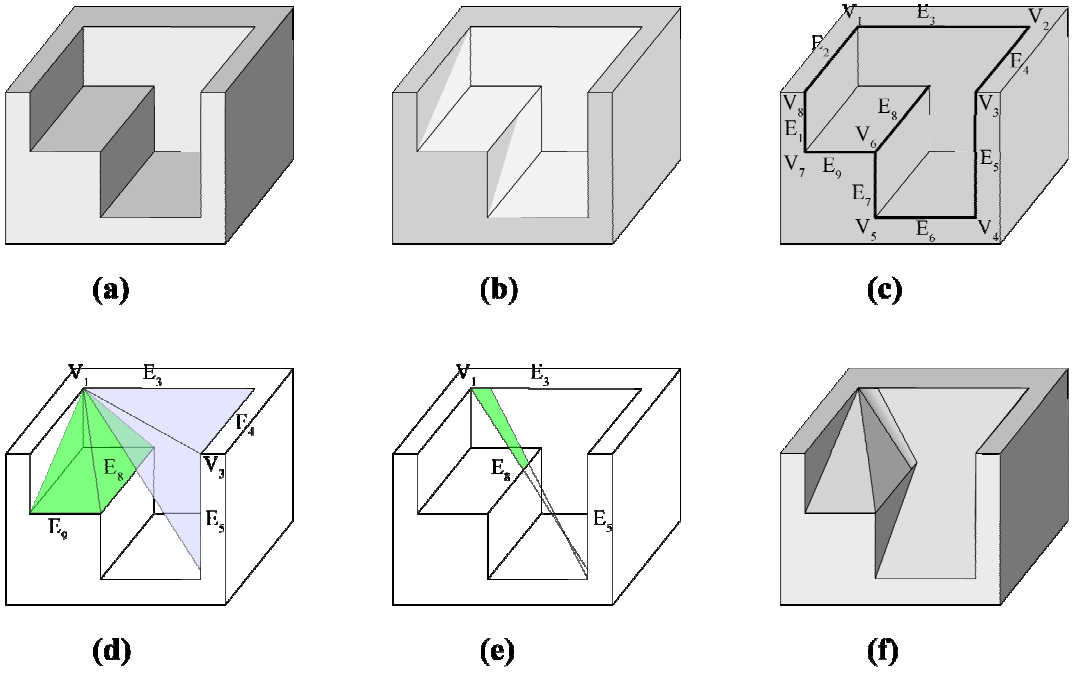

Figure 39: The object (a), the surface S'VH (b), the edges and vertices after the pruning (c), active VE patches (d), (e), the visual hull of the object (f).

5. Solids of revolution

The problem of computing the visual hull of solids of revolution can be transformed in a 2D problem, since also their visual hulls are solids of revolution.

Consider a solid **O** generated by revolution about an axis *X* of a series of planar areas lying in the same half plane. To construct **VH**(**O**), the algorithm for computing the visual hull of 2D objects of section II.B.3 can be extended, in order to find the intersection of the visual hull on an axial plane P. The intersection of **O** and P, $\mathbf{O}_P$, gives the symmetric 2D objects of Figure 40(a).

We recall that the 3D visual number of a point **p** relative to **O** is the number of families of free lines passing through **p**. A point belongs to **VH**(**O**) if and only if its visual number is zero. Therefore, computing the 2D visual hull on the axial plane P requires finding the areas of P whose 3D visual number is zero.

The boundaries of areas having different visual numbers are line and curve segments, called *active*, that belong to *visual curves*. A visual curve is the intersection with the axial plane of the surface obtained by rotating a free line about the axis of revolution *X*. Three kinds of visual curves can be obtained: hyperbolae (visual curves of type I), two straight lines symmetric with respect to *X*, when the hyperbolae degenerates (visual curves of type II), two half lines starting at *X* and perpendicular to the axis of revolution, when the free line lies on a plane normal to *X* (visual curves of type III). For each class of curves, we can define their active part. Curves of type I present active segments only if they make three contacts in P with $\mathbf{O}_P$. Active lines of type II are only those making contact at two points in P with $\mathbf{O}_P$. Active lines of type III are those normal to *X* and making contact with $\mathbf{O}_P$ at one point; the active segments are those which connect this point to *X* or to the first point where the line intersects $\mathbf{O}_P$. Examples of the three kind of active segments can be seen, respectively, in Figure 40(b), (c) and (d). Each active segment has a positive direction. Crossing an active segment in the positive (negative) direction increases (decreases) the 3D visual number by one. The complete catalogue of active segments is reported in Laurentini (1999), to which we refer the interested reader.

The algorithm for finding the intersection of **VH**(**O**) and P is the following:

1. find the active segments of the visual curves of type I, II and III
2. construct the partition of P given by the boundaries of $\mathbf{O}_P$ and the active segments
3. find the visual number of each region of the partition, merging those region whose visual number is zero; this can be done evaluating the visual number in one region and visiting the region graph from this one, adding (subtracting) one from the visual number of the starting region when crossing a boundary in the positive (negative) direction

In Figure 40(e) the partition of P for our example is shown, and the corresponding visual hull is shown in Figure 40(f).

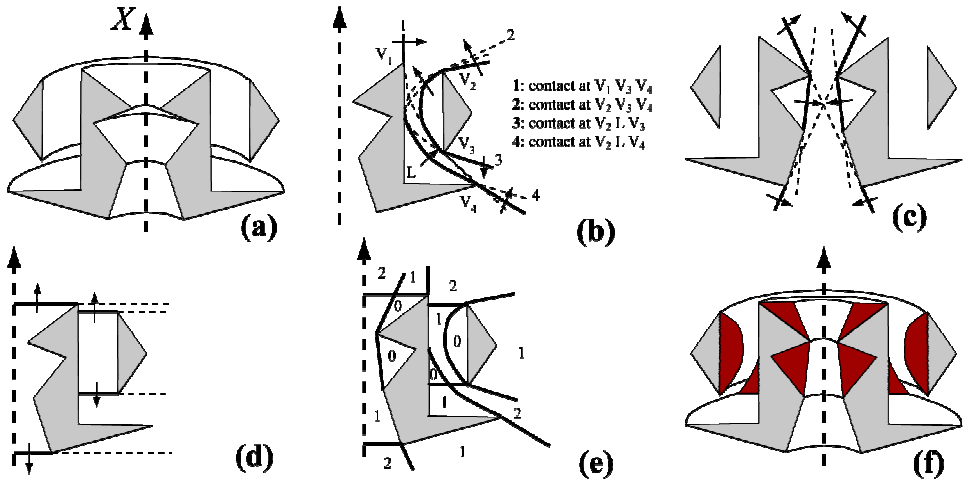

Figure 40: cross cut of the solid of revolution (a); active segments for visual curves of type I (b), II (c), III (d); the partition of (half) P, where each region is marked with its visual number (e); cross cut of the visual hull (f). © 1992 IEEE

It is worth noting that the surfaces we have used to compute **VH**(**O**) are due to *multilocal* visual events (Petitjean et al. 1992).

6. Smooth curved objects

An algorithm for computing the visual hull of smooth curved objects is described in Bottino and Laurentini (2004). It is shown that S''_{VH} consists of patches of the boundary surfaces of the aspect graph corresponding to the visual events *tangent crossing* and *triple point* of smooth curved objects. Tangent crossing occurs when the projections of two limbs meet at a point and share a common tangent (or, which is the same, the same normal) forming a tacnode (Figure 41). Triple point occurs when the free line is tri-tangent to S (Figure 42).

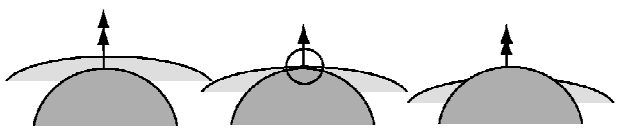

Figure 41: the visual event tangent crossing. © 2004 IEEE

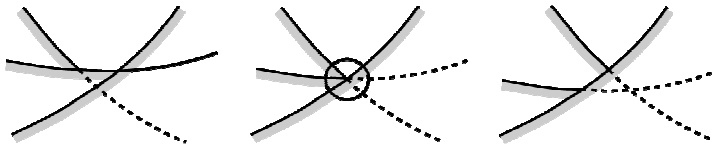

Figure 42: the visual event triple point

The equations that determine these surfaces can be found in Bottino and Laurentini (2004).

Much pruning of these surfaces can be done for finding the *active patches*. As for polyhedra, active patches are defined as the parts of these surfaces which are formed by tangent lines not intersecting elsewhere the object, and such that no free lines compatible with the surface near the tangency points pass through points of the patches.

The analysis of the surfaces corresponding to the visual event triple point is identical to that previously reported for EEE surfaces of polyhedra. Then, we only need to analyze the surfaces related to the visual event tangent crossing.

Let us consider a line tangent at two points of the surface. These points divide this line in two exterior segments and one interior segment (Figure 43). Let us consider the intersection of the surface and a plane containing the surface normals $\mathbf{n_1}$ and $\mathbf{n_2}$ at the tangency points. If the normals have the same directions, Figure 43(a), the exterior segments cannot contain points belonging to S''_{VH}, since through any point of the exterior segments can pass free lines compatible with the surface near the tangency points. The situation is different when the normals have opposite directions, as in Figure 43(b), showing that through any point of the interior segment can pass a free line.

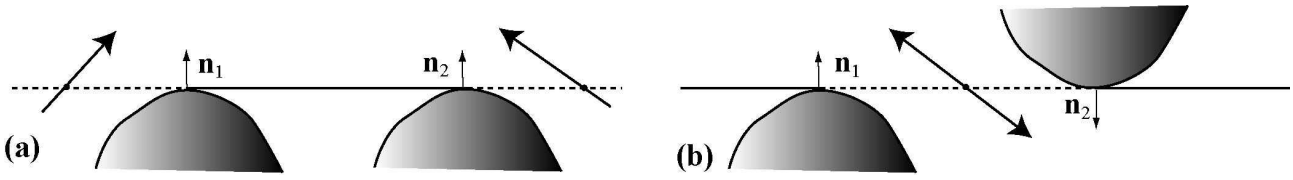

Figure 43: Two cases of bi-tangent line: potentially active segments

Each of the cases shown in Figure 43 can be analyzed in more detail considering if the shape of the surface at the tangency points is compatible with free lines trough the potentially active segments.

For performing this analysis, we recall that near any point a smooth surface can be approximated by a quadric. According to the Gaussian curvature ***k*** at a point, it is classified as elliptic (***k***>0), parabolic (***k***=0), and hyperbolic (***k***<0), and the surface near the point can be approximated by an ellipsoid, a cylinder or a saddle-shaped hyperboloid (Figure 44, see O'Neill, 1996).

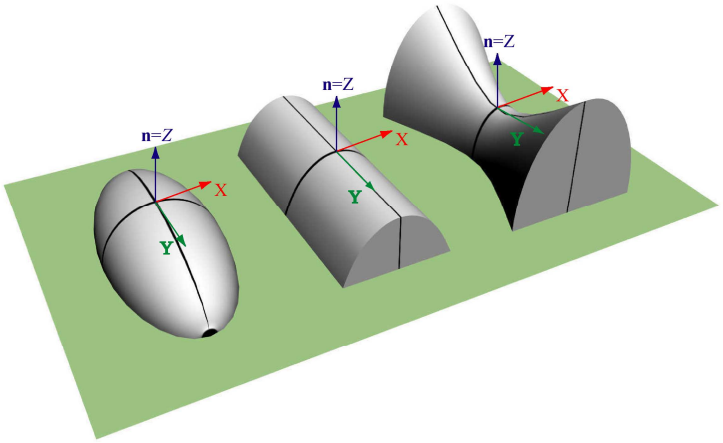

Figure 44: surface approximations: elliptic, parabolic and hyperbolic points. © 2004 IEEE

The results of this analysis are summarized in Table 1, where the state of exterior and interior segments of the two main cases, that is normals at the tangency point having or not the same direction, are itemized per kind of surface approximation at the tangency points. The details of this analysis can be found in Bottino and Laurentini (2004), to which the reader is also referred for further details on the complex case hyperbolic-elliptic.

	$\mathbf{n_1 = n_2}$		$\mathbf{n_1 = -n_2}$	
	interior	exterior	interior	exterior
Elliptic-elliptic	**inactive**	**inactive**	**inactive**	**inactive**
Parabolic-elliptic	**inactive**	**inactive**	**inactive**	**inactive**
Parabolic-parabolic	Only insolated tangent lines of this kind exist for generic surfaces	**inactive**	**inactive**	Only insolated tangent lines of this kind exist for generic surfaces
Hyperbolic-parabolic	**active**	**inactive**	**inactive**	**active**
Hyperbolic-hyperbolic	**active**	**inactive**	**inactive**	**active**

Hyperbolic-elliptic	The internal patch is divided, according to the local curvatures at the tangency points, into an inactive surface, near the hyperbolic point, and an active surface, near the parabolic point	**inactive**	**inactive**	The external segments near the parabolic and hyperbolic points are partially or totally active according to the local curvatures at the tangency points

Table 1: active and inactive bi-tangent surface patches

The algorithm for computing the visual hull is similar to that described for polyhedra, the only difference being the different computation of active surfaces.

An example is shown in Figure 45. The object, which is symmetric, has been modeled as a NURBS surface, and the rim of the concavity has a high curvature; (a) and (b) shows two views of the object. Three active patches, shown in (c), (d) and (e), are candidate to cover the concavity. Patch P_1 is generated by the lines touching the object along the contour generators C_1 and C_2, and is an instance of the active case hyperbolic-hyperbolic. P_2 is generated by lines tangent at C_3 and C_4 (which is very close to C_2). Since the object is symmetric, all these line pass through a point and form a conical surface. This is an instance of the case parabolic-hyperbolic, and only the patch near the parabolic rim is active. P_3 is symmetric to P_1. Lines L_1 and L_2, which are the boundaries between P_1 and P_2 and P_3 and P_2 respectively, correspond to a flex point of the projection of the far rim. Intersecting the three patches (P_1 and P_3 intersect along a curve lying in the symmetry plane of the object), only one closed cell results, which belongs to **VH**. The visual hull surface, formed by P_1 and parts of P_2 and P_3, is shown in (f).

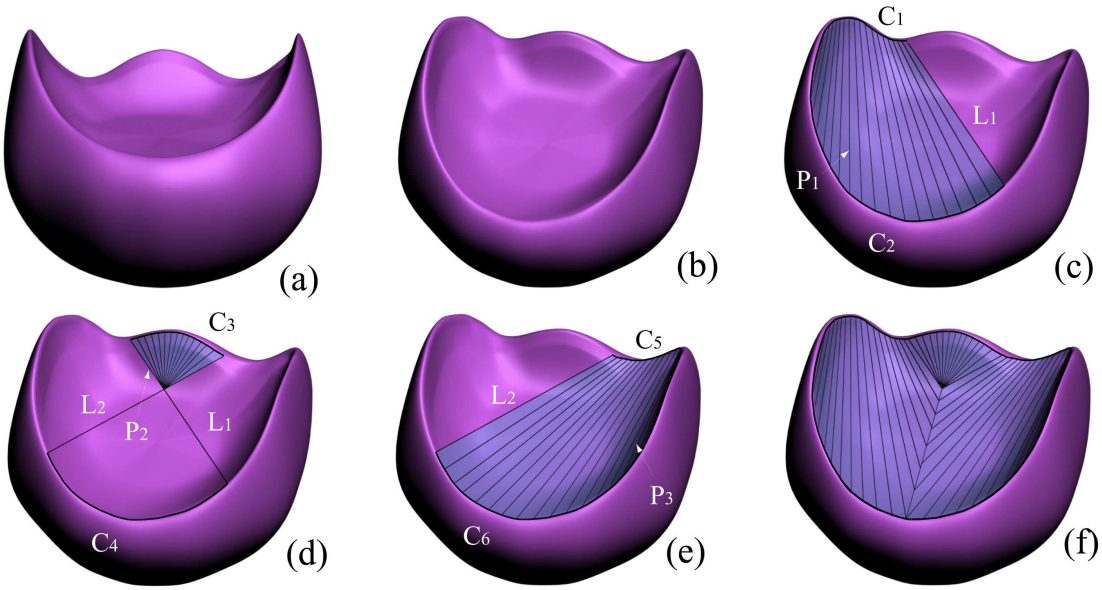

Figure 45: (a), (b): original object; (c), (d), (e): active surfaces; (f). © 2004 IEEE

C. Understanding 3D shapes from silhouettes: hard and soft points

Volume intersection summarizes the information provided by a set of silhouettes, and constructs a boundary volume **R** which approximates more or less closely an unknown object. Only if the object **O** is coincident with its visual hull we are able, at least in principle, to reconstruct the object exactly.

In this section we will deal with the problem of inferring the shape of the real object **O** from the object reconstructed **R**, or, which is the same, from a set of its silhouettes. We will face first the *optimal* case, where we have been able to construct the best possible approximation, the visual hull. Then we will consider the case which is more likely to happen in practice, where we do not know if the reconstructed object **R** is the visual hull or not. The problem has been discussed first in Laurentini (1995).

The intuitive analysis of a simple example will provide some insight into the problem. Let us consider the problem of inferring the shape of an object from the visual hull. In Figure 46, various objects sharing the same visual hull, the cube of Figure 46(a), are shown. They are part of an

infinite family of objects ranging from the cube itself to zero-volume objects like the one shown in Figure 46(e), provided that no free line intersect the bounding cube without intersecting the object itself. The objects must lie within the bounding cube: what else can we infer about their shape? It is easy to realize that all these objects share the 12 edges of the cube with the surface of the visual hull. In addition, the surfaces of these objects can take any shape, without the need to be connected, as long as no line passes through the object without intersecting these surfaces.

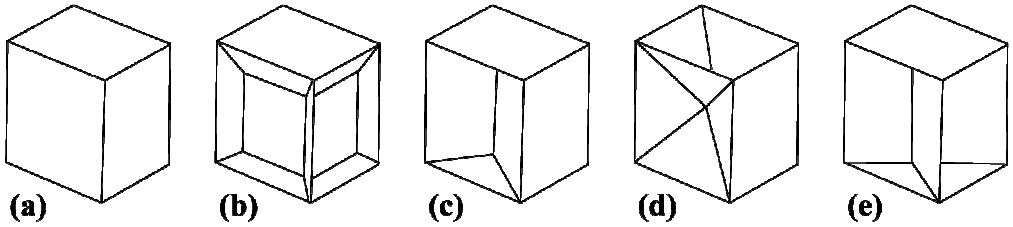

Figure 46: all objects have the same visual hull, coincident with the object (a). © 1995 IEEE

This example shows that in general the best reconstruction achievable is not sufficient to infer the real shape of the unknown object. However, we have also seen that some points of the surface of the visual hull are shared by any possible object **O** originating on the visual hull, while the other points may or may not belong to **O**. This leads to the following definition:

Definition 4: let **p** be a point on the surface of a visual hull **VH**. **p** will be specified as*:*

- a *hard* point if it also belongs to the surface of any possible object originating **VH**
- a *soft* point otherwise

In other words, a soft point may or may not belong to the surface of **O**. This definition raises a problem: how can we find hard and soft points on the surface of a visual hull?

Recalling *Proposition 5*, the following statement can be derived (Laurentini, 1995):

Proposition 10: a necessary and sufficient condition for a point **p** belonging to the surface of a visual hull **VH**(**O**) to be hard is that at least one line L passes through **p** without intersecting **VH**(**O**) at any other point.

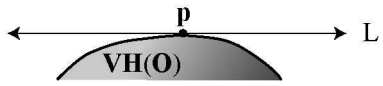

Figure 47: a necessary and sufficient condition for p to belong to VH(O). © 1995 IEEE

Proposition 10 allow us to divide the surface of the visual hull into a part belonging to the originating object, the *hard* part, and a part which is an outer bound of **O**, the *soft* part. However, even if the surface of the unknown object **O** can assume different shapes within the boundary of the soft surfaces, every visual ray intersecting the soft surfaces will also intersect **O**. We will call this the *opaqueness property*. It is important to note that the opaqueness property does not hold for a generic reconstructed object **R** when it is not coincident with the visual hull.

Finally observe that, in spite of its name, volume intersection supplies precise information only on the surface of an unknown object.

Proposition 10 is a point condition. Finding the hard surfaces for various types of visual hulls requires further work. Let us consider polyhedral visual hull. We have seen that polyhedral visual hulls can be originated by polyhedra, but not every polyhedron has a polyhedral visual hull. It can be shown that the interior points of faces of a polyhedral visual hull are always soft, and only the convex edges can be hard (Laurentini, 1995). It is also easy to see that a sufficient condition for a convex edge to be hard is to lie on the convex hull of the visual hull. This immediately confirms the intuitive analysis of the hard points of the object of Figure 46.

A general algorithm for finding the hard edges of polyhedral visual hull in $O(n^6)$ time can be found in Laurentini (1995). Only curved visual hulls can have hard surfaces. For instance, all the surface of a spherical visual hull is hard. However, constructing algorithms for finding the hard points of non-polyhedral objects, as smooth or piecewise smooth object is currently an open problem.

The results presented so far in this section (inferring the real shape from the visual hull) are rather theoretical. In practice, when in addition to silhouettes no other information on the object is available, we do not know whether the object **R** reconstructed by volume intersection is or not the

visual hull. In the following we will face the practical problem of finding hard points on the surface of a generic reconstructed object.

A simple example will provide some insight into the problem. It will also show that in general a generic reconstructed object **R** supplies less information on **O**, i.e. less hard points, than the same object considered as a visual hull. Figure 48 shows a cube reconstructed from three square silhouettes obtained with viewing directions parallel to the axes. It is clear that infinite objects enclosed by the cube supply the same silhouettes. Figure 49 shows some of these objects, three polyhedra and three origami-like objects. In particular, inspecting the origami-like objects it is easy to realize that in this case no hard point exists on the surface of **R**.

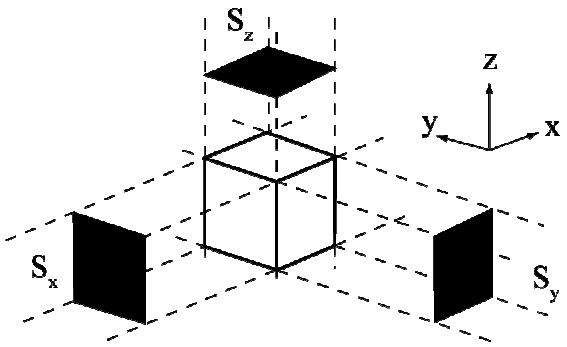

Figure 48: the volume R obtained from three square silhouettes. © 1995 IEEE

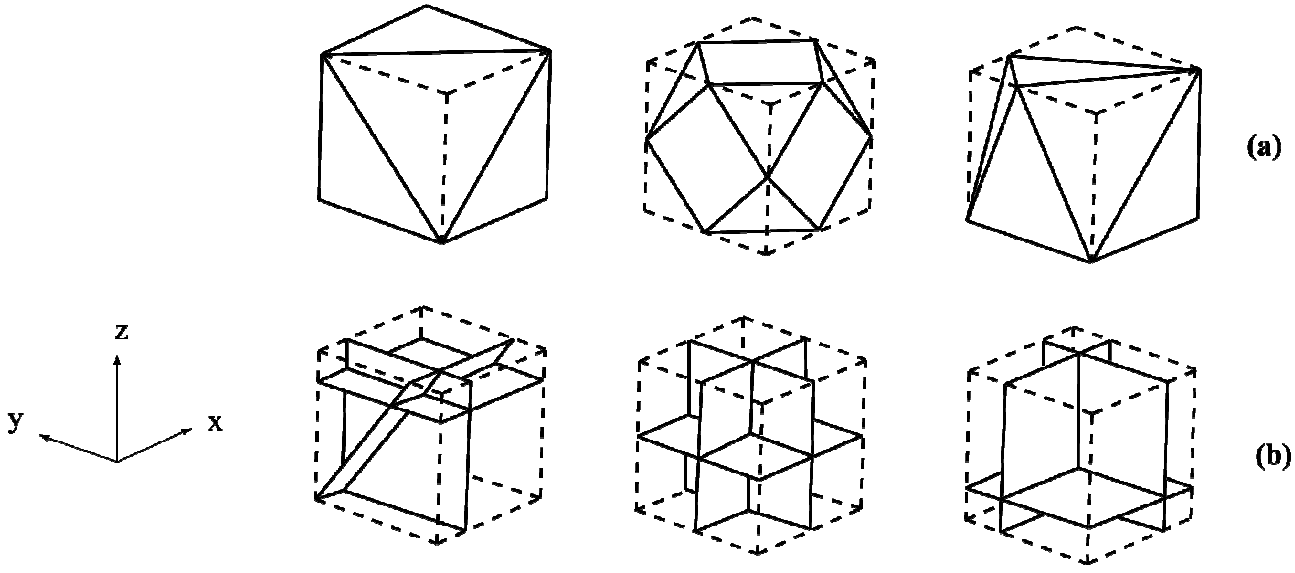

Figure 49: three polyhedra (a) and three zero-volume, planar face objects (b) which may give the three square silhouettes of Figure 48. © 1995 IEEE

In order to understand how to find the hard points, let us consider one of the silhouettes involved in the process of VI, the corresponding viewpoint **V** and the viewing cone circumscribed to the object **O** and starting at **V** (Figure 50). The viewing cone is tangent to **O** along a curve $\mathbf{C_V}$ that is shared by **O** and the surface of the cone. In general this curve can have discontinuities, and, for exceptional alignments, a half line of the circumscribed cone may share multiple points or segments with **O**. This curve belongs to an annular surface homeomorphic to a closed curve, a strip **ST(V)** of variable width (measured along a line of the cylinder), which is what is left of the original circumscribed cone after the various intersections. During the reconstruction process, this annular strip cannot be interrupted (i.e. become homeomorphic to an open curve), since no point of **O** can be removed by VI; at most it can reduce to a curve with zero width. The following theorem holds:

Proposition 11: let **p** be a point on the surface of the reconstructed object **R**. A necessary and sufficient condition for **p** to be hard is that it belongs to at least one strip **ST(V)** of width zero at **p**.

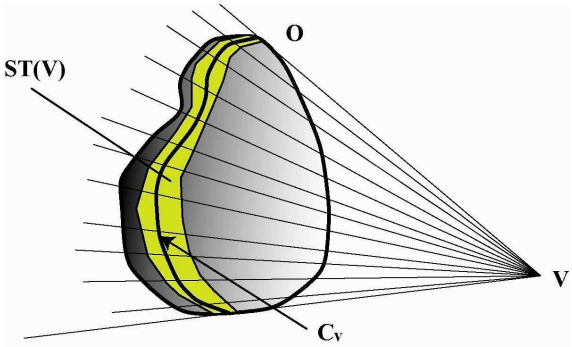

Figure 50: VI reduces the surface of the circumscribed cone to ST(V) by, containing the curve CV belonging to O. © 1995 IEEE

Let us summarize the results obtained. The surface of any reconstructed object consists of a number of strips $\mathbf{ST(V_i)}$, one for each viewpoint $\mathbf{V_i}$. On each of these strips there is a curve $\mathbf{C_V}$ of points belonging to **O,** but usually we are not able to locate the curve. The only case when we are able to locate points of this curve, and therefore hard points, is when the strip has zero width.

An important consequence of the proposition is that we are not able to find hard surfaces when considering an object **R** reconstructed with a finite number of intersections, but only hard points or,

at most, hard lines. As a matter of facts, the visual hull is able to supply hard surfaces, but they require an infinite number of zero-width strips for their full description.

It is not difficult to understand that hard points results when two strips intersect (Figure 51). Hard line results in situations like the one depicted in Figure 52, where the two viewpoints $\mathbf{V_i}$ and $\mathbf{V_j}$ produces the corresponding strips $\mathbf{ST(V_i)}$ and $\mathbf{ST(V_j)}$. Viewpoint $\mathbf{V_k}$ produces a third strip $\mathbf{ST(V_k)}$ of width zero and therefore coincident with the hard curve $\mathbf{C_{Vk}}$. From the figure, it is also easy to see that hard lines can appear where there is a *crease*, that is a discontinuity of the surface normal.

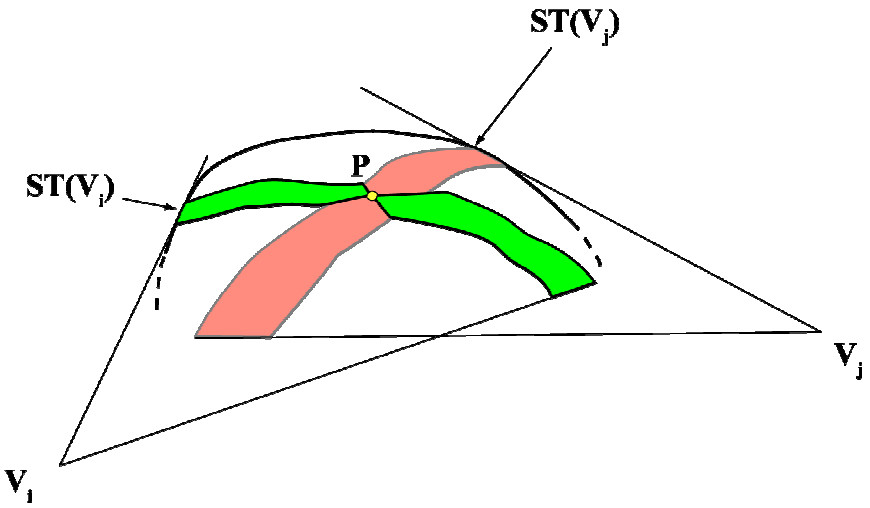

Figure 51: How to generate a hard point P on a smooth surface. © 1995 IEEE

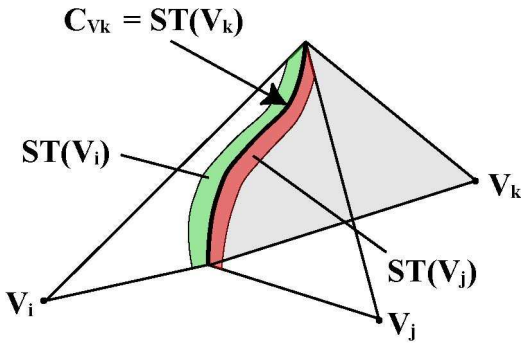

Figure 52: how a hard line is generated. © 1995 IEEE

An important practical consequence of this discussion is that hard points and hard lines can be simply obtained as a by-product of the VI algorithm.

A simple example is shown in Figure 53 that shows an L-shaped polyhedron observed from ideal coplanar viewpoints. Two ideal coplanar viewing directions, $\mathbf{V}_1$ and $\mathbf{V}_2$, shown together with the corresponding strips, do not produce hard points, since both strips have everywhere non-zero width (Figure 53 (a)). One additional co-planar viewing direction $\mathbf{V}_3$ supplies four R-hard segments (Figure 53 (b)), highlighted with thick lines.

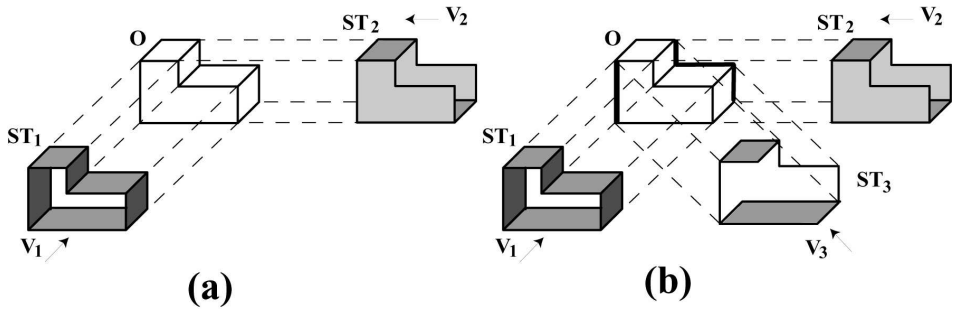

Figure 53: examples of hard edges

For further detail and examples the reader is referred to Laurentini (1995).

D. How many silhouettes does the best reconstruction require?

In theory, the visual hull is the best reconstruction that we can obtain from the silhouettes of an object. In practice, also the number of silhouettes required by this optimal reconstruction is important. In this section we will briefly deal with the problem of finding the minimum theoretical number of silhouettes required (Laurentini, 1997). Let us first define the volumetric accuracy A_V of the VI reconstruction as the ratio between the volumes of the real object **O** and that of the object reconstructed **R**:

$A_V = \text{Vol}(\mathbf{O})/\text{Vol}(\mathbf{R})$

A_V can be expressed as the product of two accuracies:

$A_V = A_{VH}\, A_I$ where $A_{VH} = \text{Vol}(\mathbf{VH}(\mathbf{O}))/\text{Vol}(\mathbf{R})$ and $A_I = \text{Vol}(\mathbf{O})/\text{Vol}(\mathbf{VH}(\mathbf{O}))$

This allows to separate the accuracy A_I due to the intrinsic properties of the object from A_{VH}, due to the number and positions of the viewpoints. Let us define *e*-reco an object coincident with its visual hull, such that $A_I = 1$.

Also define *f*-reco objects whose visual hull can be reconstructed with a finite number of silhouettes. It is clear that the volumes obtained by back-projecting the silhouettes have conical surfaces, for perspective projections, formed by lines passing trough a point, or cylindrical surfaces formed by lines parallel to a direction for parallel projections. Therefore, the surface of any object obtained with a finite number of VI operations consists of a finite number of patches of conical or cylindrical surfaces.

It follows that objects with curved surfaces are *f*-reco only in exceptional cases. Even a simple sphere is not *f*-reco. Also a cylinder is not *f*-reco using perspective silhouettes. In these cases, since these objects are *e*-reco, we can speak of reconstruction with arbitrarily high accuracy A_{VH}.

Polyhedra are a more promising category of objects. A convex polyhedron, which is *e*-reco as all convex objects, is also *f*-reco. In fact, if the planes supporting the faces are in general position, we can choose $\lceil n/3 \rceil$ viewpoints at the intersection of disjoint triplets of these planes. More viewpoints are required, in any case less then *n*, if some planes are parallel each other.

Let us consider a general polyhedron. If it is not *e*-reco, the surface of the visual hull could contain curved quadric patches. Although ruled, they are not patches of a cone or a cylinder, and then these polyhedra are not *f*-reco.

Let us restrict ourselves to polyhedra with polyhedral visual hulls. It seems not unreasonable to expect that the minimum number of silhouettes required is bounded by some function of *n*. Actually, in general this is not the case. For some classes of polyhedra we have the rather counterintuitive result that, for a given *n*, the number of silhouettes required can assume any value. An example of such a polyhedral visual hull with 14 faces is shown in Figure 54(a). Reconstructing face F, and in particular the part F' highlighted in the figure requires viewpoints lying in the plane of the face and outside the convex hull, and then outside the face. Figure 54(b) shows that the part

of F' reconstructed by each viewpoint can be made arbitrarily small by reducing the distance between the vertical wedges, without changing the number of faces.

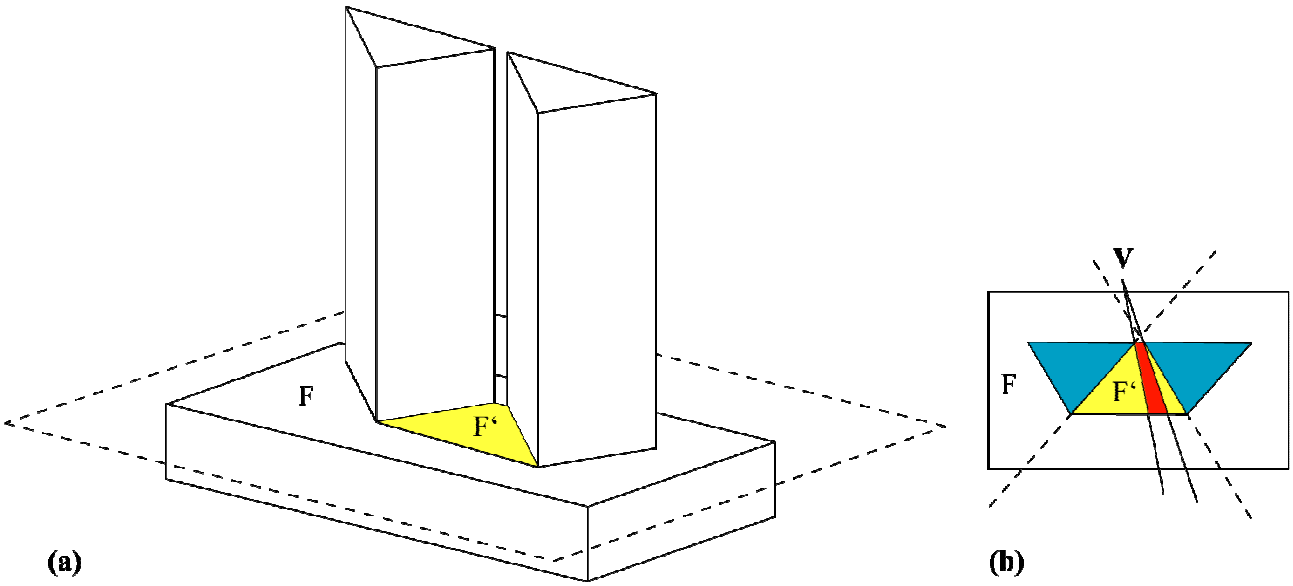

Figure 54: a visual hull whose reconstruction may take any number of silhouettes

The situation is different if we want to reconstruct an internal visual hull, and then viewpoints inside the convex hull are allowed. In this case, it can be shown that a polyhedral internal visual hull is *f*-reco. In Laurentini (1997) an algorithm is described, which computes polyhedral internal visual hull using $O(n^5)$ viewpoints.

E. Interactive reconstruction

In the previous section we have discussed the theoretical minimum number of silhouettes required for the optimal reconstruction. Silhouette-based reconstruction algorithms must face another important problem. If no *a priori* information about the 3-D shape is available, we have no idea of the accuracy of the reconstruction obtained, and consequently we do not know whether halting or not the reconstruction process. It is also clear that, for a given object, each new VI operation can refine the reconstruction to different degrees, depending on the viewpoint chosen. Then another problem is where to locate new viewpoints. If we were given the shape of the object, in principle we could construct some object specific algorithm for finding the next best viewpoint, but the shape is

the very information we are looking for. So in general we are reduced to a simple “the more silhouettes the better” strategy.

Object specific heuristics for finding the next viewpoint have been suggested, for instance in Shanmukh and Pujari (1991) and Lavakusha et al. (1989). Some results have been also obtained in the area known as *geometric probing* (see Skiena, 1992, for a survey).

In Bottino et al (2001) a *quantitative* approach is presented, able to solve, at least in part, this apparently under-constrained problem. The approach is based on a *necessary condition* for the reconstruction obtained to be optimal. The condition can be verified considering the reconstructed object **R** only, without any *a priori* knowledge about **O**. If the necessary condition is not satisfied, it is possible to derive suggestions for locating new viewpoints.

We have seen in section II.C that the points of the *surface* of **R** can be divided into *hard points*, which belong to any possible object originating **R**, and *soft points*, which could belong or not to **O**. The computation of the hard points of **R** can be performed under two hypotheses: 1) **R** is the visual hull; 2) **R** is a generic object reconstructed by VI. Let us call *VH-hard* the hard points in the first case, and *R-hard* a hard point in the second case.

.

1. The interactive reconstruction approach

Consider an object **R** reconstructed by VI. Let VH(**R**) be the set of VH-hard points of **R**, and R(**R**) be the set of R-hard points. The following proposition holds.

Proposition 13: VH(**R**) $\supseteq$ R(**R**)

This follows from the fact that each new VI operation cannot delete R-hard points already obtained, but only add new R-hard points, and the visual hull is the reconstruction made with all the possible silhouettes. If the best possible reconstruction has been obtained, that is **R**=**VH**(**O**), no more hard points can be found. It follows:

Proposition 14: a necessary condition for the reconstruction to be optimal is*:* VH(**R**)=R(**R**)

Let us define *compatible* a reconstruction that satisfies this condition, that is, such that all the points of **R** which *could be* hard (the VH-hard points) have been found to be *actually* hard (R-hard points). The condition in general could be not sufficient for the reconstruction to be optimal, and in some cases small details of the object could remain undetected.

However, the necessary condition is fruitful. If it is satisfied, we can stop the VI process, although the optimal reconstruction is not absolutely guaranteed. If not, we are sure that the optimal reconstruction has not yet been obtained, but the R-hard and VH-hard points suggest the location of the next viewpoint.

a) A general approach to interactive reconstruction

The general idea of the interactive reconstruction approach is attempting to obtain a compatible reconstruction, and, in any case, to improve at each step the accuracy, which is not guaranteed if the new viewpoints are selected at random.

At each step:

- Compute VH(**R**), R(**R**)
- If VH(**R**)=R(**R**), stop
- Otherwise, refine the reconstruction by checking if the potentially R-hard points VH(**R**)-R(**R**) are actually R-hard or not. This can be done by selecting a viewpoint such that some potentially hard points are projected on the boundary of the silhouette of **R**.

Whatever the new silhouette of **O** turns out to be, it improves the reconstruction. If the new viewpoint produces a circumscribed cone containing half-lines tangent at potentially hard points, these points are verified as R-hard. If not, the VI process deletes a part of the reconstructed object **R**. Then either we find new R-hard points, or we increase the volumetric reconstruction accuracy A_V. In any case, our knowledge of **O** improves. The outlined approach can be applied to any class of objects (i.e., convex polyhedra, concave polyhedra, curved objects…) provided that an algorithm

for evaluating VH-hard points is available for that class. As an example this approach has been applied to convex polyhedra.

b) An interactive algorithm for convex polyhedra

Interactively reconstructing unknown convex polyhedra from silhouettes has been studied by Dobkin et al. (1986). Their algorithm, restricted to orthographic projections, uses a strategy that determines the polyhedron with a bounded number of silhouettes N_s:

$$V/2 \leq N_s \leq V+5F$$

where V and F are the numbers of vertices and faces respectively. The algorithm has not been implemented.

The interactive approach described is able to reconstruct any convex polyhedron from a bounded set of parallel or perspective silhouettes and has been implemented for virtual polyhedra (Bottino et al., 2001). It consists of three parts:

1. ALGVI, which performs the volume intersection;
2. ALGR, the algorithm for computing the R-hard points R(**R**);
3. NEXT, which compares VH(**R**) and R(**R**) and, if the sets are not equal, computes the next viewpoint;

ALGVI implements VI and works for any kind of polyhedra (actually also concave) and parallel and perspective projections. ALGR is a straightforward consequence of VI. In this case the algorithm in Laurentini (1995) for computing the VH-hard points is not necessary, since any reconstructed object **R** is convex, and all edges are VH-hard. Therefore, NEXT computes new viewpoints until all the edges of **R** are R-hard. An edge is R-hard if it belongs to a strip with zero width. Then to verify an R-hard edge E requires three planar surfaces of three cones $\mathbf{C}_i$ making contact with **R** at E. To verify a face requires a viewpoint in the plane of the face.

(1) The strategy of NEXT

The general approach described in section II.E.1.a) guarantees that each new viewpoint improves the reconstruction, but not that the reconstruction is performed with a bounded number of silhouettes. To this purpose, NEXT uses a set of rules that guarantees that at each step a face at least is verified, or a contact with at least one undetected R-hard edge is obtained. For the exact reconstruction, NEXT requires

$$F/3 + 2 \leq N_s \leq F + 3E$$

silhouettes (see Bottino et al., 2001).

NEXT works as follows. Start with two random viewpoints (after the first VI, all edges are potentially R-hard). For each following step, select a potentially R-hard edge E and locate the viewpoint according the following rules, related to the characteristics of the faces F_1 and F_2 of **R** which meet at E. Three cases are possible.

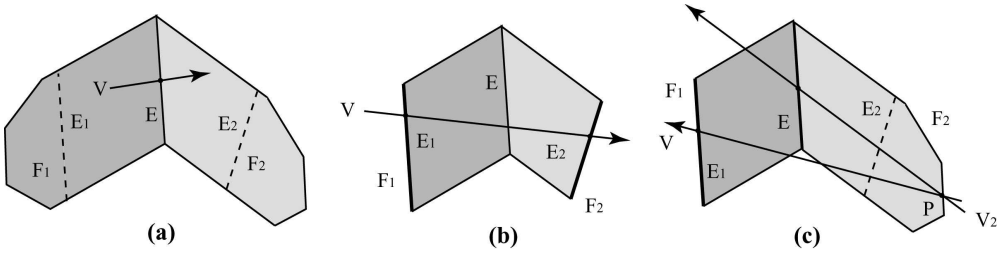

Figure 55: three cases for the candidate edge E

Case 1 – The edges of **O** that produce F_1 and F_2 have not yet been detected (see Figure 55(a), where the undetected edges E_1 and E_2 are shown as dotted lines). In this case, any viewpoint projecting E on the boundary of the silhouette is convenient. In fact, it will either verify E as a R-hard edge, or produce at least a new face of **R** making contact with one or more R-hard edges yet undetected inside the region bounded by F_1 and F_2. Then the viewpoint can be located anywhere in two of the four regions into which the space is divided by the planes of F_1 and F_2.

Case 2 - There is one R-hard edge on both F_1 and F_2 (see Figure 55(b), where the hard edges are thick). A viewpoint lying on a line passing through one point of both the hard edges will either verify a possible undetected face (for co-planar hard edges), or produce one or more faces making

contact with undetected R-hard edges. Also in this case the viewpoint can be located anywhere in an infinite space region, which is planar for coplanar R-hard edges.

Case 3 - There is one R-hard edge on one face; let it be F_1 (see Figure 55(c)). Consider a line starting at $\mathbf{V}_2$ (the viewpoint which produces F_2) and lying on F_2. The intersection of this line with F_2 is bounded by two points. Let P be the one closer to $\mathbf{V}_2$. Chose a new viewpoint on a line passing trough P and one point of the R-hard edge E_1 on F_1. It is clear from the figure that it will produce one or more new faces making contact with undetected R-edges (possibly E_2). Also in this case the viewpoint is restricted to lie in an infinite space region.

In some cases the above conditions can be satisfied for more than one potentially R-hard edge. Actually, two additional rules have been used for speeding up the algorithm. Since the polyhedron is known to be convex, if two R-hard vertices are joined by a potentially R-hard, the edge can be immediately classified as R-hard, even if it does not belongs to a strip of zero width. This happens frequently in our examples.

The second rule is the following heuristic. When possible, a viewpoint is located on a line containing one R-hard edge, if at least one of the faces sharing the edge has not yet been verified. This verifies one or two faces, usually deletes several non R-hard edges and adds one or two faces tangent at several R-hard edges not yet detected.

An example of the steps performed by the algorithm is shown in Figure 56, where the arrows indicate a line containing the viewpoint and the R-hard edges are marked thick. The original object is shown in (a) and, at the end of the reconstruction, when all the edges are R-hard, in (h). In (b) the first cone is shown, truncated by one fore and one back plane. The first rule and the heuristic are used in several cases, (d) to (g), while (c) refers to case 1.

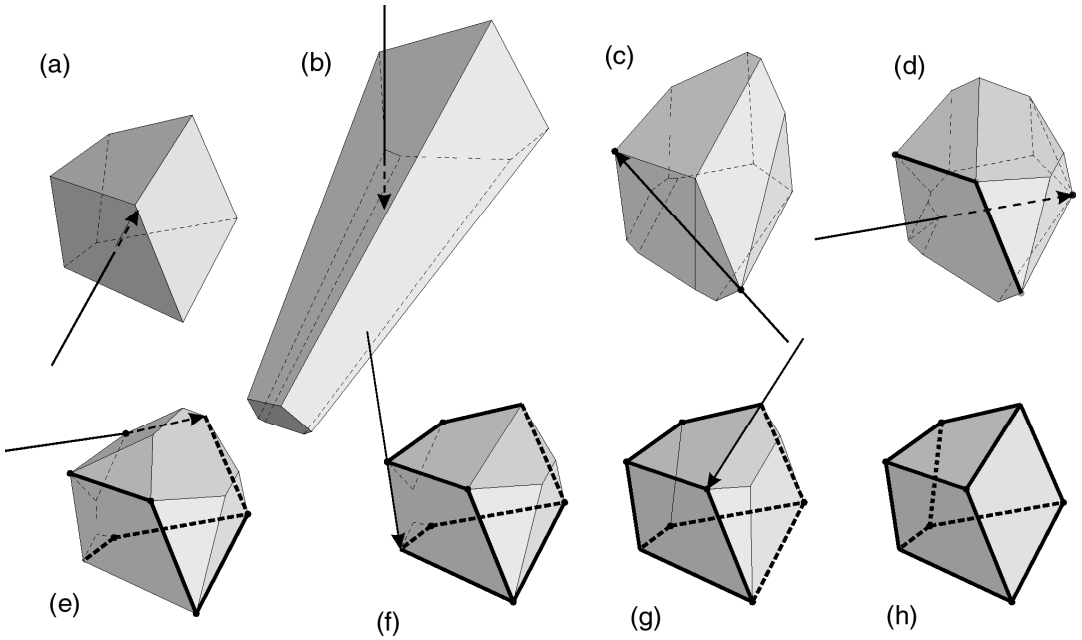

Figure 56: an example of reconstruction of a convex polyhedron

We have experimentally evaluated the performance of the algorithm with respect to our upper and lower bounds, and to the bounds of the algorithm of Dobkin et al. For this purpose, the algorithm has been applied to fifty randomly generated convex polyhedra. The average number of vertices, edges and faces of the fifty polyhedra are 17.8, 26.7 and 10.9 respectively. We have found that the average number of silhouettes required for the reconstruction is 13. This number should be compared with 91, our upper bound averaged over all the fifty polyhedra, with 99.9, the average upper bound of Dobkin et al., and with 5.6 and 8.9, the two corresponding lower bounds. These comparisons show that the average behavior of our algorithm is much closer to the lower then to the upper bounds.

F. Reconstruction with unknown viewpoints

So far, we have discussed the problem of reconstructing 3D shapes from 2D silhouettes by back-projecting them from the corresponding viewpoints and intersecting the resulting solid cones. This requires knowing the position of the viewpoints with respect to the object. However, in several practical situations, as observing a vehicle, a plane or an asteroid, this information is not available.

In these cases we have a set of silhouettes together with the position of the corresponding viewpoint with respect to each silhouette, or, in other words, a set of circumscribed cones, without knowing their relative position. This raises the following questions

1. does an object exist able to produce them?
2. if a set of silhouettes can be generated by the same object, how can we find one or more objects able to produce the same silhouettes?

In the following, we will call *compatible* a set of silhouettes if the same object can generate them. An object able to produce a compatible set of silhouettes will be said to be compatible with the set. As an example, are the three orthographic silhouettes in Figure 57 compatible?

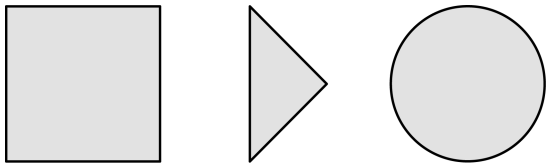

Figure 57: Can these orthographic silhouettes be generated by the same object? © 2003 IEEE

A preliminary discussion of the problem has been reported in Bottino and Laurentini (2003), where the case of planar orthographic silhouettes has been thoroughly investigated. This case idealizes some practical situations, as observing a vehicle on a planar surface, or a ship on a calm see.

First, we must define the compatibility of two silhouettes. Given S, a 2D orthographic silhouette of a 3D object, and $L(S,\alpha)$, the length of the 1D silhouette of S obtained projecting orthographically S along a direction in the plane of S making an angle α with the x axis of that plane (see Figure 58), two silhouettes S_1 and S_2 are compatible if two angles α_1 and α_2 exist such that $L(S_1,\alpha_1) = L(S_2,\alpha_2)$.

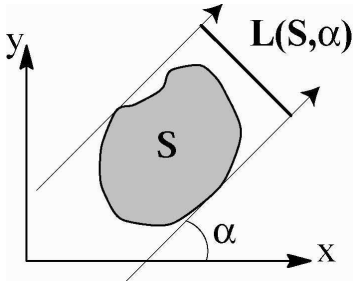

Figure 58: 1D silhouette of S. © 2003 IEEE

When two silhouettes are compatible, it is easy to select the parameters required for intersecting in 3D the cylinders obtained by back-projecting S_1 and S_2. First, we must select the angle β between the viewing directions, or, which is the same, the angle between the planes of the silhouettes. Let I be the intersection of these planes. The two cylinders must be positioned as follows:

- the 2D viewing directions that produce $L(S_1,\alpha_1)$ and $L(S_2,\alpha_2)$ must be both orthogonal to I
- both cylinders must be supported by the same plane P orthogonal to I

It is clear from the figure that the object obtained by VI is compatible with both silhouettes. This reconstruction is possible for any angle β between the viewing directions.

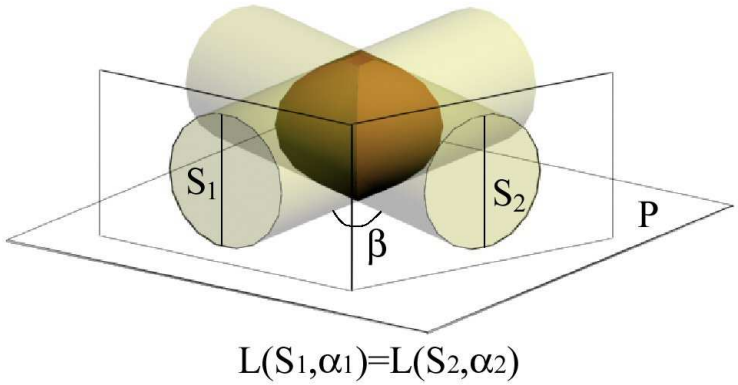

Figure 59: reconstructing an object compatible with S_1 and S_2. © 2003 IEEE

Clearly, when we have more than two silhouettes, we must ensure the pairwise compatibility between silhouettes. However, this condition is not sufficient for the full compatibility of the set, as shown in Bottino and Laurentini (2003). For instance, the three silhouettes of Figure 57 are pairwise compatible, but it can be shown that no objects exist able to produce them all.

A necessary and sufficient condition can be found referring to the annular strip **ST(V)**, that is, to what is left after various VI operations of the circumscribed cone (or cylinder, in this case) relative to viewpoint **V**.

Proposition 12: A necessary and sufficient condition for a set of silhouettes to be compatible is that it be possible to find viewpoints such that no annular strip of the reconstructed object is interrupted.

Proposition 12 allows to write sets of inequalities that determine feasible VI parameters. Before introducing them, let's give some notations (Figure 60). Each planar silhouette S_i is defined, for $0 \le y \le y_{max}$ by two curves $S_{il}(y)$ and $S_{ir}(y)$. For simplicity, let us consider mono-valued functions. Also let $S_i(y)=S_{ir}(y)-S_{il}(y)$.

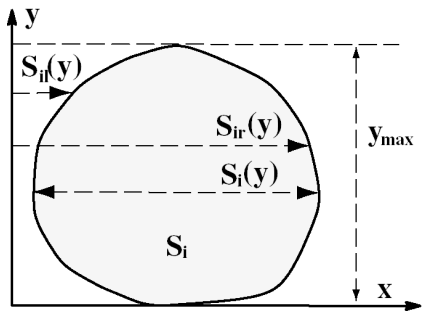

Figure 60: notations used for a silhouette. © 2003 IEEE

1. Writing the sets of inequalities

The general case is when $S_{ir}(0) \neq S_{il}(0)$ and $S_{ir}(y_{max}) \neq S_{il}(y_{max})$ for all the silhouettes considered. Special cases, not respecting these conditions, simply require considering additional constraints and are discussed in details in Bottino and Laurentini (2003).

a) Three silhouettes

Let's start considering a generic horizontal plane P between 0 and y_{max}, three viewpoints $\mathbf{V}_1,\mathbf{V}_2,\mathbf{V}_3$ and the intersections of the corresponding back-projection cylinders and the plane P are shown in Figure 61(a). Other arrangements of the viewpoints satisfying the condition of *Proposition 12* will be discussed at the end of this section. In this case, it is not difficult to see that the condition requires, for all possible values of y, that the two lines projecting the endpoints $S_{3l}(y)$ and $S_{3r}(y)$ of $S_3(y)$ along $\mathbf{V}_3$ falls inside the curves defining the projections along $\mathbf{V}_3$ of vertices **3-4** and **1-2**, respectively (see Figure 61(b)).

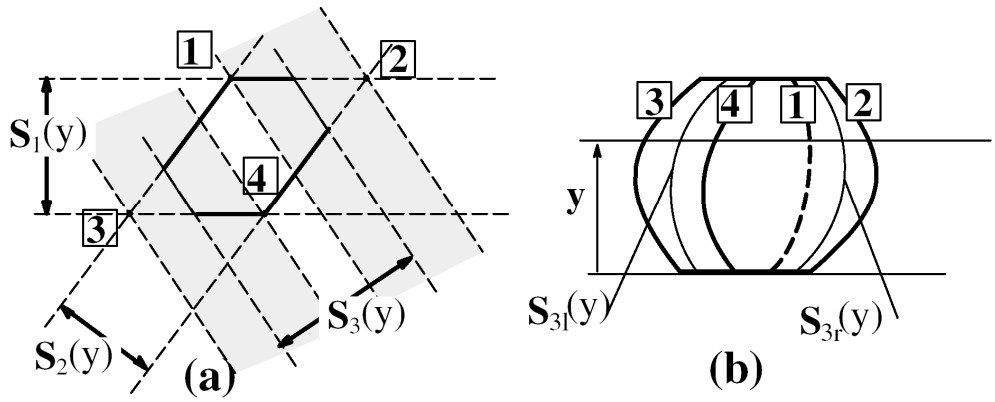

Figure 61: (a) A case where S3(y) is compatible with S1(y) and S2(y) in a horizontal plane. (b) The condition for the compatibility of the whole silhouettes. © 2003 IEEE

Inspecting in details the intersections in P, we can derive set of inequalities characterizing this case (Figure 62). Let O_1, O_2, and O_3 be the intersections of the axes y of the coordinate system of each silhouette with this plane. The unknown are three: the angle α_1 between $\mathbf{V}_1$ and $\mathbf{V}_2$, the angle α_2 between $\mathbf{V}_1$ and $\mathbf{V}_3$ and the distance *d* between the intersections on the line projecting O_1 along the direction $\mathbf{V}_1$ of the line projecting O_2 along $\mathbf{V}_2$, and line projecting O_3 along $\mathbf{V}_3$. Thus, to find feasible solutions we must search the 3-dimensional space [α_1, α_2, *d*].

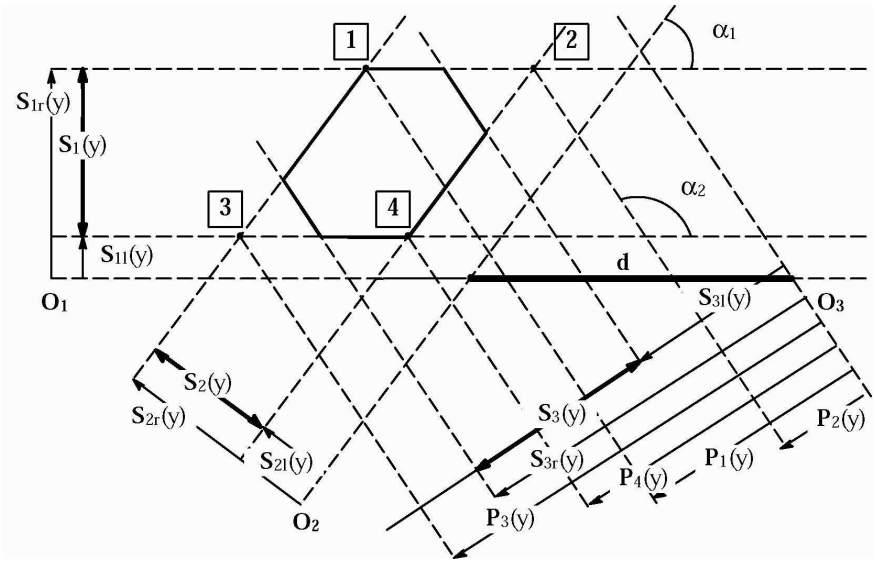

Figure 62: the intersections in a horizontal plane

As shown in Figure 61(b), the compatibility condition can be expressed in terms of $P_1(y)$, $P_2(y)$, $P_3(y)$ and $P_4(y)$, the distances from O_3 of the orthographic projections of the vertices of the parallelogram onto the line supporting $S_3(y)$. The resulting inequality set is the following:

$S_{3r}(y) \geq P_4(y), \quad S_{3r}(y) \leq P_3(y), \quad S_{3l}(y) \geq P_2(y)$
$S_{3l}(y) \leq P_1(y), \quad P_4(y) \geq P_1(y)$

The fifth inequality characterizes the case in analysis, let it be Case 1. The other feasible cases are determined by the direction of V_3 with respect to V_1, V_2, and by the directions of the diagonals V_{14} and V_{32} of the parallelogram. Each of these cases, depicted in Figure 63, produces different sets of inequalities. For each case, a possible orthographic projection onto the plane of S_3 of the edges of the object produced by the first intersection is shown with thick lines. The boundaries of S_3 are the thin lines.

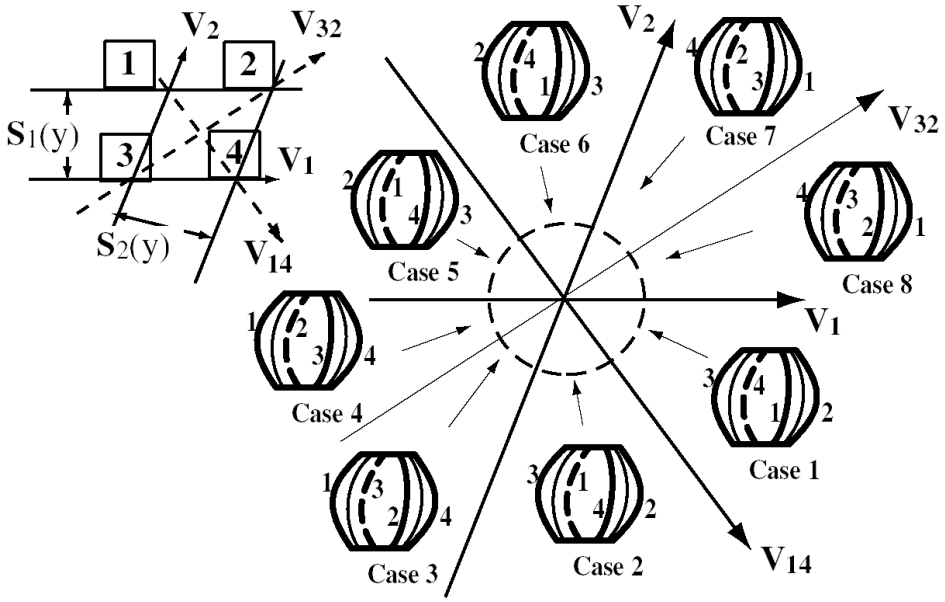

Figure 63: the eight intersection cases. © 2003 IEEE

b) Four silhouettes

Adding a fourth silhouette S_4 introduces two new parameters: the angle α_3 between $\mathbf{V}_1$ and $\mathbf{V}_4$ and the distance d_1, measured, as d, along the line that projects O_1 from $\mathbf{V}_1$. The condition of *Proposition 12* requires cutting away two opposite vertices of the polygon produced by three silhouettes, without eliminating completely the edges that meet at these vertices (Figure 64). By orthographically projecting the six vertices onto the plane of S_4 we obtain six curves. For the new intersection to be feasible, the boundaries $S_{4l}(y)$ and $S_{4r}(y)$ of S_4 must lie in the areas bounded by the two leftmost and the two rightmost curves respectively.

Various sets of inequalities result, depending on the direction of V_4. For each couple of opposite vertices, there are two main cases (case (**a**) and (**b**) in the left of Figure 64) related to the direction that determines the leftmost and rightmost vertices (**5** and **7** for case (**a**) and **7** and **5** for (**b**) in Figure 64). Each case can be subdivided into four sub-cases for the leftmost and rightmost strips where S_{4l} and S_{4r} must lie (see right part of Figure 64). The inequalities corresponding to each sub-case are easily written. For instance, for the sub-case $\mathbf{a_1}$ in Figure 64, it is:

$$P_5(y) \le S_{3l}(y) \qquad S_{3l}(y) \le P_4(y)$$

$$P_1(y) \le S_{3r}(y) \qquad S_{3r}(y) \le P_7(y)$$

$$P_4(y) \le P_6(y) \qquad P_8(y) \le P_1(y)$$

where $P_i(y)$ are the projections of the points $\mathbf{i}(y)$ onto the plane of S_4. As before, the last two inequalities guarantee that the inner boundaries of these areas are actually P_6 and P_8.

Summarizing, each set of inequalities for four silhouettes contains 11 inequalities (the five inequalities related to the first three silhouettes and six new inequalities also referring to S_4). As for the number of sets of inequalities, we have 8 cases for three silhouettes, 3 pairs of opposite vertices and 8 cases for each pair, and thus 192 sets each containing 11 inequalities.

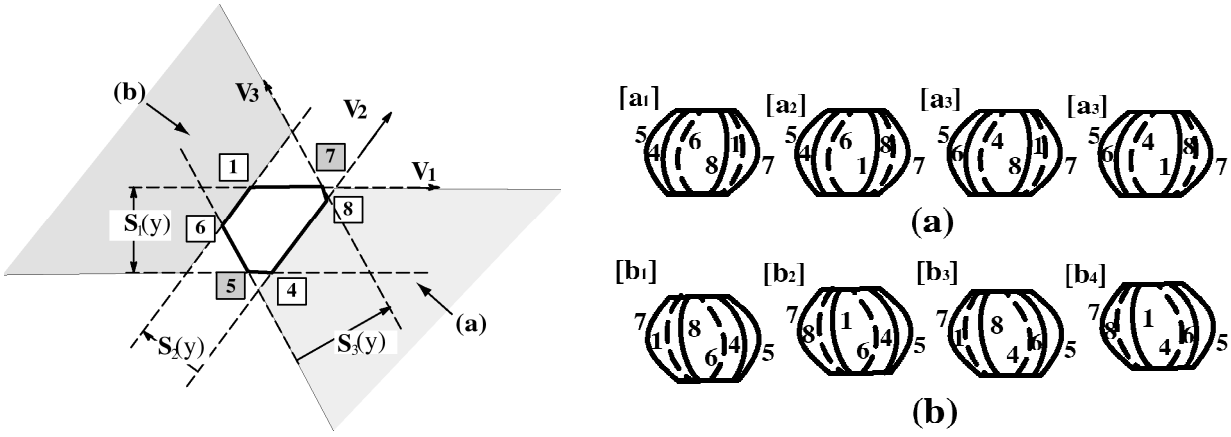

Figure 64: cases (a) and (b) and the 8 sub-cases. © 2003 IEEE

c) Five or more silhouettes

Each new silhouette adds two new parameters and, to provide compatibility, it must always cut a pair of opposite vertices without deleting completely the edges converging at these edges; this

corresponds to seven additional inequalities for each sub-case. Thus, for n silhouettes, the number of parameters is $2n$-3, and the number of inequalities $6(n\text{-}3)+5$ ($n \geq 3$). Each new silhouette adds 8 sub-cases for each pair of opposite vertices. For the n-th silhouette, the pairs of vertices are n-1. Let $N_c(n)$ be the number of sets of inequalities for n silhouettes. For $n>3$ it is: $N_c(n)=8(n\text{-}1)N_c(n\text{-}1)$.

2. Writing and solving the sets of inequalities

The inequalities discussed in the previous section allow to answer, in a particular case, both question raised: finding objects compatible with a set of compatible silhouettes, and understanding if an (artificial) set of silhouettes is compatible. We have developed an algorithm for automatically writing the sets of inequalities, detailed in Bottino et al. (2003b). A set inversion technique (Jaulin and Walter 1996) has been applied for finding the feasible solution set **S** of the set of non-linear inequalities that characterizes each sub-case. This technique performs a paving the parameter space with boxes. If the current box [**b**] is proved to be inside **S**, then the box is kept as part of the solution space. If it is proved to have an empty intersection with **S** then it is discarded. Otherwise, [**b**] is bisected except if its width is smaller than a defined threshold. The dimensionality of the initial box is equal to the number of variables involved in the set of inequalities. To prove that a given box [**b**] is inside **S**, interval computation (Moore 1979) has been used.

In Figure 65 three silhouettes S_1, S_2 and S_3 of a parallelepiped are shown. The boxes defining the paving of the solution set of each of the eight sub-cases obtained, are depicted in Figure 66, where the axis of the reference system are α_1 and α_2 on the plane and d as vertical axis. Several different compatible objects, each one reconstructed from a different feasible sets, can be seen in Figure 67.

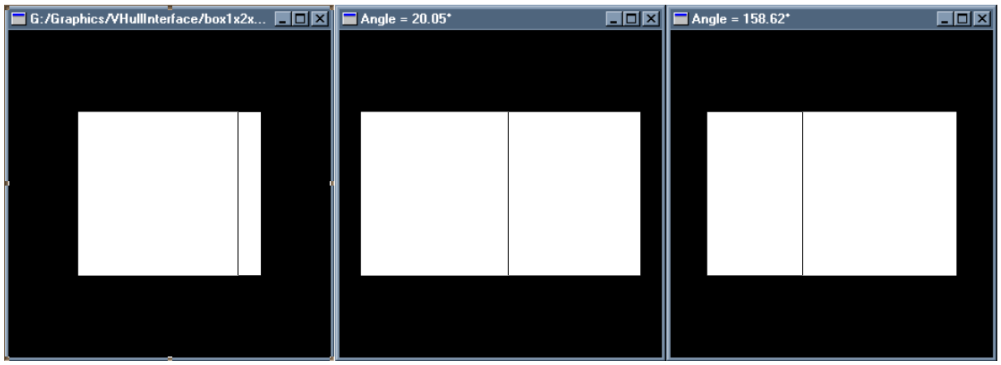

Figure 65: the silhouettes S1, S2 and S3

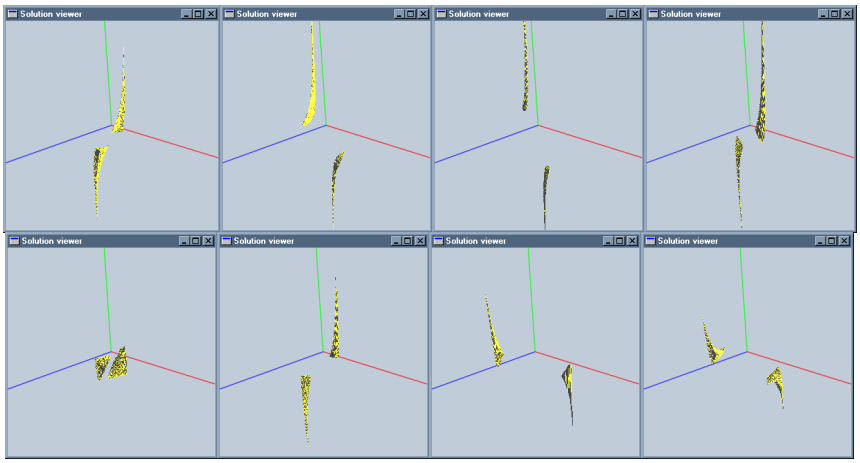

Figure 66: the eight solution spaces

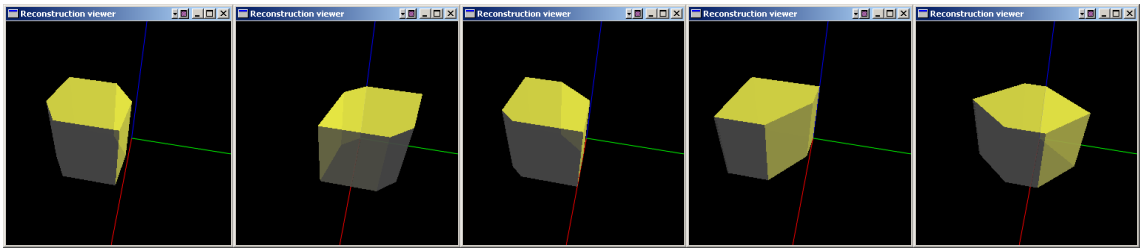

Figure 67: objects compatible with the silhouettes of Figure 65

III.Practical object reconstruction

In the previous sections we have presented a theoretical discussion on limits and capabilities of the shape from silhouette approach. In practical situations we must be content with the object **R** reconstructed using a finite set of viewpoints, which, as already discussed, in general is not the optimal reconstruction. To comply with the current parlance in the field, we will use the term visual hull, or **VH**(**O**), to indicate the *inferred visual hull* computed from a finite set of images.

Practical shape from silhouette algorithms require facing several significant problems.

First of all, in order to perform VI, the position and orientation of the image planes relative to each other must be known, in order to relate the respective back-projection cones. In other words, the cameras need to be calibrated. A copious literature is devoted to this problem, and a complete survey is outside the scope of this work. The most common approaches (Tsai 1987, Zhang 2000) require to evaluate camera parameters a grid of 3D reference points, together with their corresponding 2D projections on the image planes, such as in Figure 68. Camera parameters are divided into intrinsic parameters (specifying the camera characteristics, like focal distance, etc…) and extrinsic parameters (describing the spatial relationship between the image plane and a common and fixed 3D reference system). Efficient implementations of these techniques, such as the Tsai Camera Calibration Code (TsaiCC), the Camera Calibration Toolbox (Bouguet) and the Open Computer Vision Library (OpenCV) can be easily found over the Internet.

Figure 68: reference points on a calibration object, and their position on the image plane

Given a set of calibrated cameras, the projective geometry between two different views is completely described by the *epipolar geometry*. Referring to Figure 69, given two viewpoints $\mathbf{V_1}$ and $\mathbf{V_2}$, the corresponding image planes and a point **P** in 3D, the three points $\mathbf{V_1}$, $\mathbf{V_2}$ and **P** are coplanar, and form the so-called *epipolar plane*. The line connecting the two optical centers $\mathbf{V_1}$ and $\mathbf{V_2}$ is called *baseline* and its intersection with the image planes gives the points $\mathbf{e_1}$ and $\mathbf{e_2}$, the *epipoles*. The intersection of the epipolar plane with the two image planes gives the *epipolar lines*, passing through the corresponding epipole. The epipolar geometry can be expressed in algebraic form by means of the *fundamental matrix*. Assuming that **P** projects into the image planes in pixel space as $\mathbf{p_1}$ and $\mathbf{p_2}$, the main result of the epipolar geometry is that the linear relationship $\mathbf{p_1} \cdot \boldsymbol{F} \cdot \mathbf{p_2} = 0$ can be written, where $\boldsymbol{F}$ is the fundamental matrix. $\boldsymbol{F}$ determines the relation between a point in one image and the corresponding epipolar line on the second image. $\boldsymbol{F}$ can be computed from the calibration parameters or, for instance, with the so-called "eight point algorithm" (Hartley 1995).

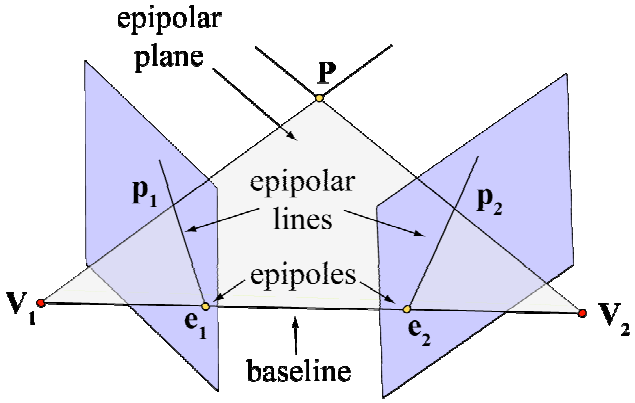

Figure 69: epipolar geometry

Another problem to face with real images is the silhouette extraction. Since usually stationary cameras are used, the silhouette extraction component processes a relatively static scene. Therefore, a naïve approach is to detect the silhouette using the difference between the current frame and an image of the background scene. Defining a likelihood parameter, each pixel of the image can be compared with the corresponding pixel on the scene model to decide if the pixel belongs to the silhouette or not. However this simple approach has a series of drawbacks. First of all there is likely

to be noise in the video signal and thus a certain neighborhood of the target color must be classified as background color. Then, classification involves the measurement of a certain distance function between the pixel color and the target region; a distance lower than a threshold indicates background, otherwise foreground. Second, in a real situation the background is unlikely to be uniform due to illumination changes, subtle camera motion, or changes in the background geometry. A possible solution is to model the scene on the background as a texture surface: each point on this surface is associated with a mean color value and a distribution about that mean. The basic idea is that, as we stated before, the pixels of the image acquired by the camera sensors contain noise, whose effect can be drastically reduced if we observe the same static background and evaluate the mean color of each pixel. This is the approach used in Wren (1997) and Yamada et al. (1998). The static model can be initialized shooting the empty scene for few seconds and it is continuously updated in an adaptive way when a new frame has been classified in background and foreground pixels. This allows compensating for changes in the lighting conditions of the scene. As an example, in Figure 70(a) is depicted the scene model built for the camera, while in Figure 70(b) is shown one of the frames where a person is present in the image. In Figure 70(c)-(d) is shown the result of the silhouette extraction process: the darker pixels represent the silhouette and the lighter pixels the cast shadows that are removed from the silhouettes. The false recognition due to the lighting changes will be slowly removed in the following frames using the adaptative filtering.
In order to cope with multimodal background distributions, more complex methods have been developed using either mixture of gaussians, Stauffer and Grimson (1999), Kernel Density Estimators, Elgammal et al. (2000), “eigenvectors” exploiting Principal Component Analysis, Oliver et al (2000), or mean-shift based estimation, Han et al. (2004).

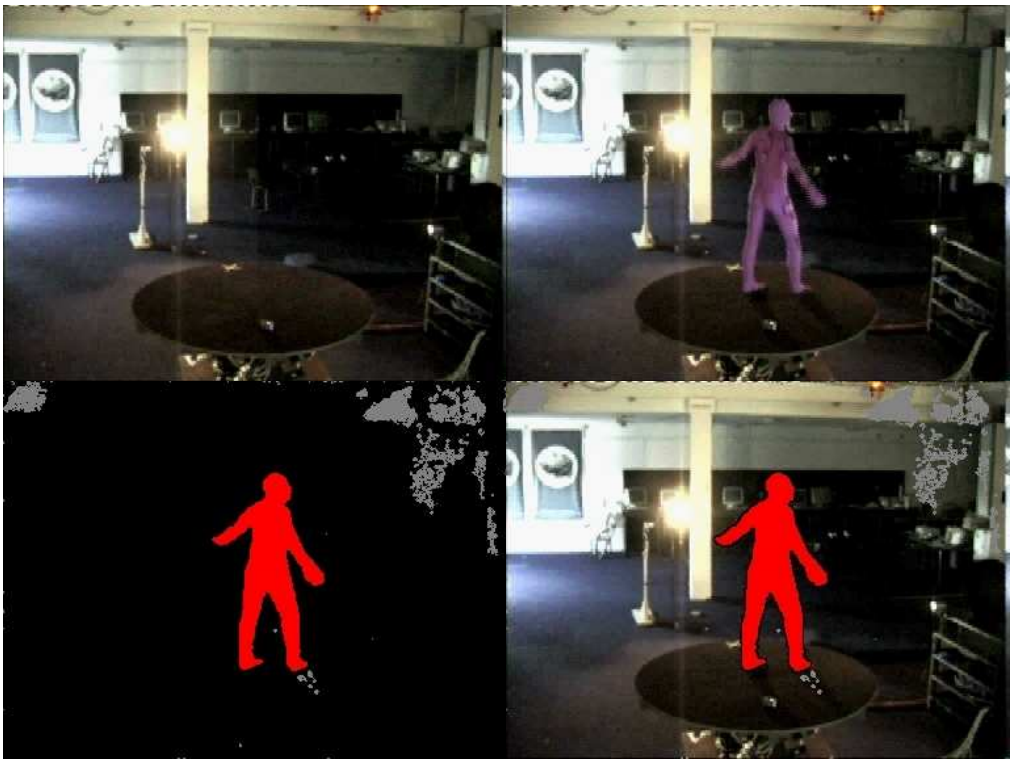

Figure 70: (a-d): the scene model, a frame of the sequence and the extracted silhouette

Finally, we must discuss the types of representation used for the reconstruction. As far as shape from silhouette has been stated, a number of modeling approaches have been proposed. They can be broadly divided in two main categories. The first includes surface based approaches, where the boundary surface of **R** is reconstructed, usually exploiting polyhedral or spline representation. The second category comprises volume-based methods, representing the entire volume as a set of finite samples of different characteristics (octrees, voxels). This is also the main taxonomy we will use in the rest of the section to describe the approaches in literature to practical shape from silhouette. This short survey is not meant to be exhaustive and will be only devoted to describe the latest contribution to the state of art.

A. Surface reconstruction

Surface modeling provides a geometric representation of the surface of the reconstructed objects. These methods compute the bounding surface of **R** from the envelope of rays defined by the optical

center and the contour of the silhouettes in the image planes. Before entering the discussion, we recall the meaning of some terms.

For each camera, the *contour generator* is the set of points of **O** where the viewlines are tangent to the object. The projection of a contour generator on the corresponding image plane is called *apparent contour*. The *visual cone* is defined as the set of viewlines passing through the optical center of the camera and all the points of the apparent contour. The intersection of all the visual cones is the inferred visual hull. Contour generators can intersect at isolated points, called *frontier points*. Frontier points are related to epipolar geometry, since for epipolar planes tangent to **O**, the tangency point is a frontier point. On the image planes, the apparent contours are tangent at epipolar tangent lines in points $\mathbf{f_1}$ and $\mathbf{f_2}$, projections on the image planes of the frontier point **F** (Figure 71). A *triple point* is a point in 3D where the viewlines passing from three points on different apparent contours intersect.

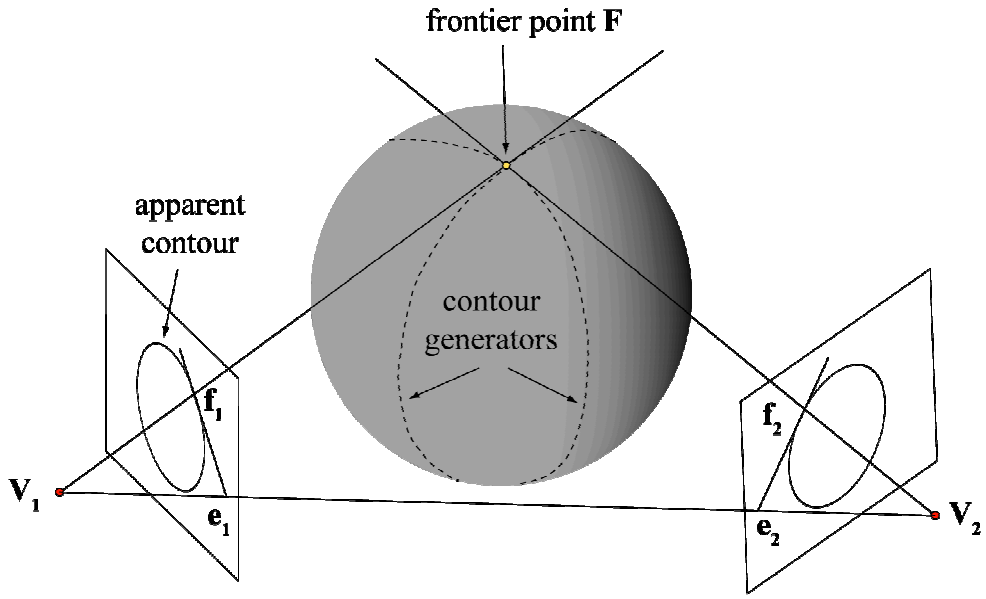

Figure 71: contour generators, apparent contours and frontier points

Several authors tackled the problem of surface visual hull reconstruction.

Spline reconstruction has been used in Sullivan and Ponce (1998). The advantage of using splines is that they guarantee certain smoothness conditions. In particular their approach uses G^1 *triangular splines*, which are triangular Bezièr patches sharing the same tangent at their boundaries. First, they

reconstruct a polygonal approximation of the volume occupied by the visual hull, intersecting in 3D the visual cones obtained back-projecting from each viewpoint a polygonal approximation of the corresponding apparent contour. The result is converted into a triangular mesh that is subsequently simplified and converted into a spline surface made up by triangular spline patches. The final step is to deform these spines to ensure that they are tangent to the viewing cones defined by the silhouettes. Surface fitting is obtained minimizing a suitable distance function between the spline surface and the viewing cones.

Lazebnik et al (2001) compute the visual hull of objects with smooth surfaces and without planar patches exploiting two data structures: the rim mesh, describing the connections of the contour generators on the surface of **O**, and the hull mesh, describing the structure of the surface of the visual hull, whose patches are the strips **ST(V)** of Figure 50. The rim mesh is a graph whose vertices are frontier points, edges are contour generator segments connecting frontier points, and faces are regions bounded by edges. The visual hull mesh is a graph whose vertices are frontier points and triple points, edges are intersection curve segments between consecutive vertices, and faces are patches of the strips **ST(V)**. Frontier points and triple points are evaluated exploiting epipolar geometry. The advantage of building the rim mesh is that its topology can be used to reconstruct the adjacency relationship between vertices and edges of the visual hull. However, the approach is limited by the fact that only zero genus surfaces can be reconstructed. Otherwise, the graph built is not necessarily complete. Calibration errors and noise on the input data are also taken into account.

Matusik et al. (2001) compute a polyhedral textured representation of the visual hull. The apparent contours are represented with their polygonal approximation. The face of the cone corresponding to each edge of an apparent contour is intersected with all other viewing cones. The intersection is performed in 2D, projecting the viewing cones on the plane of the face, and then back-projecting in 3D the result. The obtained polygons maintain as texture a reference to the images that contain it, entirely or partially. The final polygon texture is evaluated at rendering time by means of a series of

blending weights that are used to merge the original images associated to the polygon. The weights are based on the angles at the rendered vertex between the virtual camera and the corresponding image.

The approach of Franco and Boyer (2003) can deal with general surfaces, producing a polyhedral representation of the visual hull based on frontier points and triple points. Also in this case the apparent contours are approximated by polygonal contours. For each vertex of these polygons, the corresponding viewline is cast in 3D, and its intersection with all the viewing cones is computed. The result is a set of intervals along this line, called the *viewing edges*, which are the contribution of the viewing line to the visual hull. However, since the surface of the object can be general, viewing edges alone may not be sufficient to reconstruct a closed surface. Therefore missing vertices and connections needed to reconstruct the final polyhedron are computed using the geometry and orientation of the apparent contours.

An interesting and original contribution is the one proposed by Kang et al. (2001). They exploit a dual space based on differential geometry that relates a point of an occluding contour to the corresponding local tangent plane on the 3D surface of the object. The position of this plane is deduced by fitting an implicit polynomial surface of degree *n*, which estimates the local surface of the object, in the dual space. The surface of the reconstructed object is obtained as union of the set of locally estimated polynomial patches. The approach has been generalized in Brand et al. (2004) in order to reconstruct an entire free-form surface. They still make use of the dual space, but the tangent plane position is derived from continuity principles, given the position of nearby points on the surface and information on the local surface curvature. The output is a set of tangent planes and corresponding surface contact points (lying on the surface of the visual hull), that can be transformed into a 3D surface by locally intersecting the tangent planes.

There are also works exploiting multiple image cues for surface reconstruction. In the work of Isidoro and Scarloff (2003) an initial polygonal representation of the **VH**, constructed via CSG, is simplified and subdivided in order to provide a more uniform distribution of vertices on the surface.

Then photometric consistency with the incoming images is used to remesh the visual hull, until the error between the silhouettes and the reprojection of the textured visual hull falls below a predefined threshold. Hernandez Esteban and Schmitt (2004) use a snake-based deformable model which is adapted to the **VH** by means of two external forces based on the images, driven by the shape of the silhouettes and by the model texture.

B. Volumetric reconstruction

Algorithms for intersecting visual cones of general objects in 3D are complex and generally prone to rounding errors. Therefore, effective approaches to represent the volume occupied by **R** are needed. The most common is a regular tessellation of cubic primitives, called voxels, and the reconstruction is defined as an occupancy classification problem, the voxel belonging to the volume if at least one of its vertices project inside all the silhouettes in the corresponding image planes. When the model is used for synthesizing new views, several rendering algorithms can be used, including ray casting, splatting and shear-warp. Other representations use octrees, allowing a more efficient subdivision of the scene space (Potemesil 1987, Noborio et al. 1988, Ahuja and Veenstra 1989). From such models, it is possible to reconstruct a polygonized representation of the surface of the visual hull using the Marching Cubes, Lorensen and Cline (1987), or the Marching Tetrahedron algorithms, Carneiro et al. (1996).

A 2D example of voxel-based reconstruction is shown in Figure 72. Image (a) shows the inferred visual hull of an object computed from four views, while in (b) its approximated voxelized reconstruction is shown. As can be seen, the latter is significantly bigger.

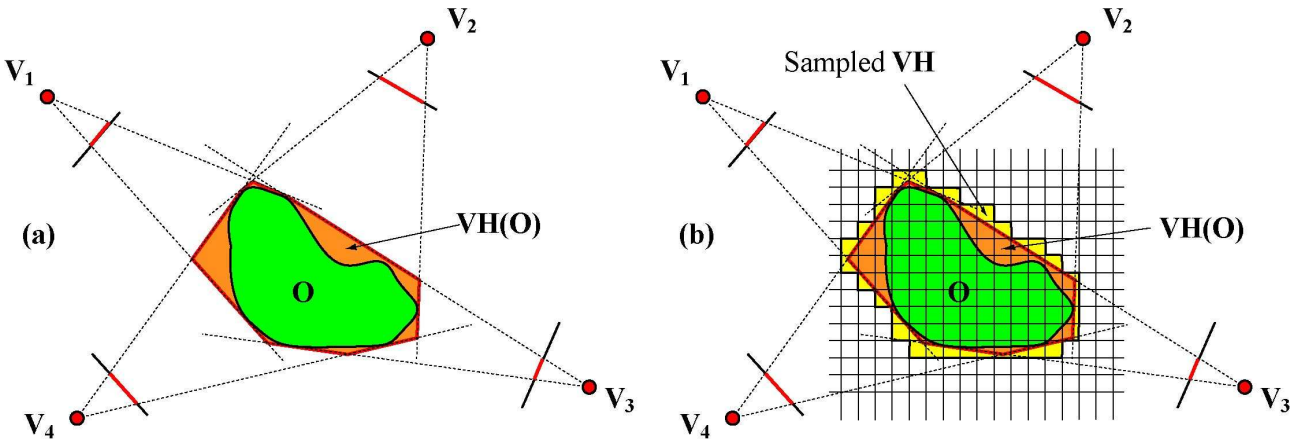

Figure 72: (a) the object O and its inferred visual hull; (b) the approximated reconstruction of VH (O)

A different representation, using the so-called Bounding Edges (BE), has been introduced by Chen et al. (2003). A BE is defined as follows. A point *u* on an image plane corresponds to a 3D line in world space. The BE is the part of this line that projects inside all the silhouettes used for reconstruction. It can be immediately seen that a BE can be composed by several separate segments along the originating line. The visual hull is then reconstructed in terms of the BE corresponding to points sampled on the contour of all the silhouettes used. It should be noted that each BE is part of the strip **ST(V)** shown in Figure 50, that is to what is left of the visual cone relative to **V** after the various volume intersection operations, and therefore it belongs to **VH**(**O**). Hence, this representation, although incomplete, is an effective description of the surface of the visual hull.

All these approaches, however, provide a coarse reconstruction of the visual hull. The results can be improved when multiple image cues, and not only the binary silhouettes, are used. The first attempt has been presented in Fromherz and Bichsel (1995), where volume intersection is combined with a consistency test based on the luminance of the pixels in the image planes. This basic idea has been extended by Seitz and Dyer (1997) with their *voxel coloring* algorithm. Initially the volume containing the scene is reconstructed by means of a traditional voxel based algorithm. Then voxels are checked for color consistency with the silhouettes in the image planes, and inconsistent voxels are removed from the initial volume. Finally consistent voxels are assigned the colors they project in the image planes, enhancing the quality and realism of the object when rendered from a different

view. Consistency checking requires knowing voxel visibility in the image planes. Since this is a complex problem, some simplifications have been introduced. If it is possible to establish a topological ordering of the voxels, that is if it is possible to sort the voxels in order of visibility with respect to a set of cameras, voxels can be traversed in this order, guaranteeing that when a voxel is visited, all other possible occluders have been already checked. An example is when we have a single camera, Fuchs at el. (1980), or when all the cameras lye on the same side of a plane, Langer and Zucker (1994). In their work, Seitz and Dyer (1997) demonstrated that a multi-view visibility ordering exists for particular camera configurations, that is when **O** lies outside the convex hull of the optical centers of the cameras.

Those pioneering works focused the attention of the scientific community on the so-called shape from photo-consistency problem, whose main recent contributions are detailed in the following sections.

1. Space Carving

Space carving has been proposed in its original formulation by Kutulakos and Seitz (2000). This method addresses the problem of computing an unknown shape from a multiple set of colored images taken at arbitrary viewpoints. It takes into account several image and object cues, like parallax, occlusion, scene radiance, shape contours and image correspondences.

First, the authors prove the existence of a *photo-hull*, which is a shape subsuming all the class of objects having appropriate reflectance properties that, seen from any of the specified viewpoints, produce the same input image. This can be done constraining the reconstructable objects to belong to the particular subclass of objects whose radiance function is locally computable (or, in other words, when global effects of illumination, like diffuse to diffuse, diffuse to specular and specular to diffuse interactions can be neglected, as well as shadows and transparencies) and also accordant to a known radiance function. The existence of the photo-hull is also in accordance with the fact that 3D shape reconstruction from a set of images is an under constrained problem, since multiple

shapes can be consistent with the set of input images. This problem has been discussed thoroughly in chapter II for silhouettes.

Second, the authors give a set of consistency rules, to define when a point is consistent with a single image, when a set of objects is consistent with a single photo, and finally when a set of objects is consistent with the full set of input images. Point consistency is given when the point does not project on the image plane on a background pixel and its computed radiance is equivalent to the color of the image pixel. Objects consistency with respect to a single photo is given when all not background pixels correspond to consistent points. Objects consistency with the full set of input photos is given when the objects-photo consistency is guaranteed for every input image. These rules are used to characterize the photo-hull, and are used in the photo-hull reconstruction.

The process of photo-hull reconstruction is performed by effectively carving an initial volume and repeatedly scraping off small portion of volume until the final shape is reached. The initial volume is supposed to consist of a set of voxels. When a voxel is checked to be not consistent, it is removed from the volume, until no more inconsistent voxels are found. The contribution of this work is how visibility is dealt with. All the cameras are considered, and no particular configuration of the position of their optical centers is required. Since in the general case there is no topological ordering of the voxels with respect to the viewpoints, voxels are traversed in a visibility compatible order by means of a multi plane-sweep algorithm, along the positive and negative directions on the three coordinate axis, and consistency is checked using only the cameras lying on the positive side of the plane. A different approach has been used in Culbertson et al. (2000). Their algorithm iterates through bounding voxels, until no more inconsistent voxels are found. Visibility is determined using all the cameras at the same time exploiting a particular data structure, *layered depth images* (Shade et al. 1998) to update visibility constraints when inconsistent voxels are carved away.

A similar approach, presented by Carceroni and Kutulakos (2001), uses *surfels*, or surface elements, instead of voxels to reconstruct the surface of **R**.

As can be seen from this description, the final result is not an exact reconstruction of the photo-hull, but only its approximated description in terms of its composing voxels. Therefore, the quality of the approximation is related to the size of the voxels, and no exact geometric description of the final shape is given. Another problem is given by the photo-consistency measure used to decide when the voxel should be eliminated. In the work of Kutulakos and Seitz (2000) this measure is based on the color variance of the projections of the voxel on all the image planes, which is thresholded to decide when to retain a voxel. The drawback of this method is that it is very sensitive to the global threshold selected, and that a single threshold is unlikely to give optimal results for complex scenes. Therefore, other researcher have used different approaches, like in Slabaugh et al. (2000), where the authors introduce a post-processing iterative method, where voxels are added or removed from the initial result until the squared sum of the differences between the reconstructed volume and its reprojections on the input images are minimized. Other approaches, Agrawal et al. (2001), Broadhurst et al (2001), Yezzi et al (2002), Montenegro et al. (2004), use probabilistic approaches, where each voxel is assigned a probability based on an appropriate likelihood, and the reconstructed surface is minimal with respect to a metric defined by these probabilities.

Another problem of this approach is the model of surface reflectivity, and the difficulty of dealing with specular highlights or textureless surfaces. The work of Yang (2003) tries to overcome this difficulty by developing a novel photo consistency measure, valid for both lambertian and specular surfaces, conjugated with a progressive space carving scheme that, starting from some reliable voxels, incrementally add voxels using photo consistency, uniqueness and local smoothness constraints. The basic idea is to defer any decision about uncertain voxels until a sufficient amount of evidence has been achieved. The new photo-consistency measure is based on the assumption that the surfaces are such that the reflected light is only modulated by the incident light, like plastic and glass. In this case, the reflected colors observed by different viewpoint are collinear in color space, that is they form a segment starting from the diffuse color to the color of the incident light. This *color signature* in color space is used for evaluating photo consistency.

2. Image Based Visual Hull

Image based Visual Hull (IBVH), Matusik et al. (2001), is a method for producing an image-based rendering of the **VH** from silhouettes. The method renders the **VH** from a generic view without reconstructing an auxiliary representation. Given the image to create, for each pixel, a ray starting from the optical centre of the virtual camera is cast in 3D. The ray is projected on each reference image, its intersection with the corresponding silhouette is computed and back-projected on the 3D ray, and the intersection of all these back-projections is computed. All these operations are optimized exploiting the epipolar geometry. The point of the resulting intersection on the 3D ray closer to the virtual viewpoint, which is a point belonging to the surface of the visual hull, is then assigned the color of the projection of the point on the image that is more similar to the virtual view. The similarity measure is given by the angle between the visual hull point and the desired camera centre. An example can be seen in Figure 73, where a virtual image is created from two input images.

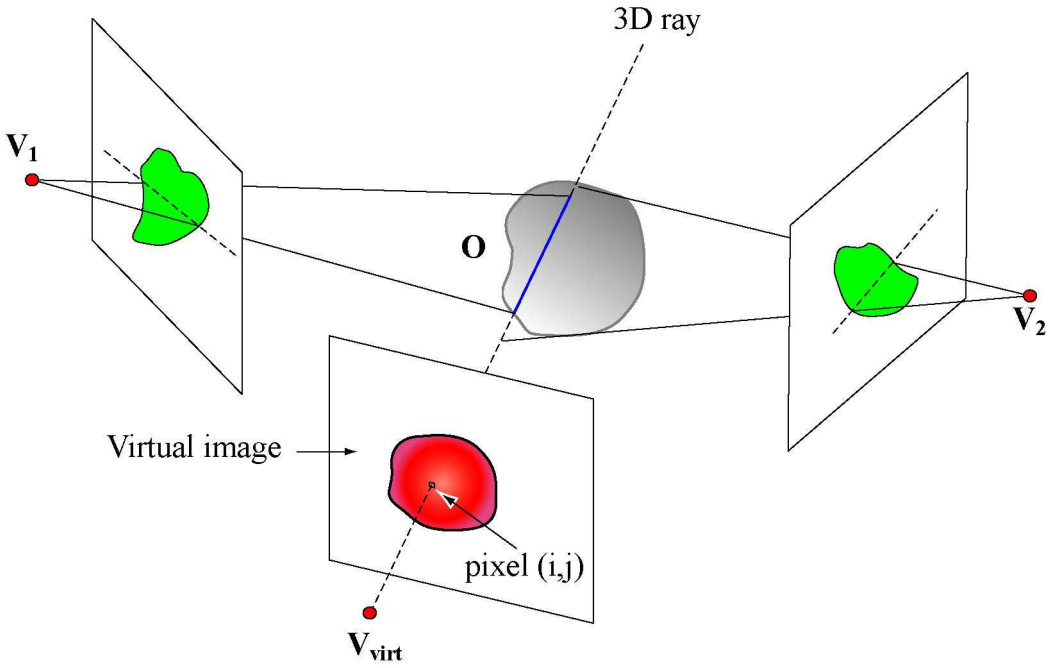

Figure 73: the ray from pixel (i,j) is projected from the desired view, then it is projected on the input images with the corresponding silhouettes; finally their back-projections in 3D are intersected, and the closest point to the virtual view is rendered on the virtual image

The Image Based Photo Hull (IBPH), by Slabaugh et al. (2003), combines the IBVH and the photo hulls approaches. The drawback of IBVH is that it is not very accurate, since it reconstructs only the visual hull. On the contrary, the photo hull is a tighter bound to the object **O**. Therefore the IBPH combines the efficiency of the IBVH and the improved reconstruction capabilities of the photo hull. The idea of the algorithm is simple. First, the IBVH is reconstructed. Then, for each ray, the photo consistency of the closest point on the visual hull is evaluated. If the point is inconsistent, the point is shifted away from the virtual camera center of a small portion, until a consistent point is found or the entire ray is discarded as being totally inconsistent. The advantage of this method is also that only the portion of the photo hull visible from the virtual camera is effectively reconstructed. The disadvantage is the loss of performances. On similar test data, as shown in Slabaugh et al. (2003), the IBVH runs at 24 frames/sec, while the IBPH runs between 6 and 7.5 frames/sec.

IV. References

M. Agrawal and L. Davis. "A probabilistic framework for surface reconstruction from multiple images." IEEE Conference on Computer Vision and Pattern Recognition(CVPR), 2001.

N. Ahuja and J. Veenstra: Generating octrees from object silhouettes in orthographic views, IEEE Trans. on PAMI, Vol.11, pp.137-149,1989

J. Aloimonos," Visual shape computation," *IEEE Proc.*Vol.76, no.8, pp.899-916, August.1988

V.I. Arnold, *Catastrophe Theory*, Springer-Verlag, Heidelberg,1984

B. G. Baumgart. "Geometric modeling for computer vision". Technical Report Artificial Intelligence Laboratory Memo AIM-249, Stanford University, 1974.

A. Bottino, L. Cavallero and A.Laurentini, "Interactive reconstruction of 3D objects from silhouettes," Proc. WSCG'2001, Vol.II, pp. 230-236, Plzen, February 5-9, 2001

A. Bottino, A. Laurentini, "Introducing a New Problem: Shape-from-Silhouette when the Relative Positions of the Viewpoints is Unknown". IEEE Transactions on Pattern Analysis and Machine Intelligence, Vol. 25, No. 11 , Nov. 2003, pp. 1484 -1493

A. Bottino, L. Jaulin, A. Laurentini, "Reconstructing 3D objects from silhouettes when the Relative Positions of the Viewpoints is Unknown: the case of planar orthographic views". Proceedings of CIARP 2003, L'Havana (Cuba), 26-29 November, 2003

A. Bottino and A.Laurentini, "The Visual Hull of smooth curved objects", IEEE Transactions on Pattern Analysis and Machine Intelligence, Vol. 26, No. 12 , Nov. 2004, pp. 1622-1632

Bouguet, Camera Calibration Toolbox, http://www.vision.caltech.edu/bouguetj/calib_doc/

K.W. Bowyer and C.R. Dyer, "Aspect Graphs: An Introduction and Survey of Recent Results," *Int'l J. Imaging Systems and Technology*, vol.2, pp. 315-328, 1990

M. Brand, K. Kang, D.B. Cooper, "Algebraic solution for the visual hull", Proc. CVPR'04, pp. 1063-1069, 2004

A. Broadhurst, T. Drummond and R. Cipolla, "A Probabilistic Framework for Space Carving" - IEEE International Conference of Computer Vision (ICCV), Vancouver, Canada, pages 388-393, July 2001

J. Callahan and R. Weiss, ”A Model for Describing Surface Shape,” in *Proc. IEEE Conf. on Comp. Vision and Pattern Recognition*, pp. 240-245, 1985

R. L. Carceroni and K. Kutulakos. Multi-view scene capture by surfel sampling: From video streams to non-rigid 3d motion, shape, and reflectance. In Proc. Int. Conf. Computer Vision, June 2001

B.P. Carneiro, C. Silva, A.E. Kaufman, Tetra-cubes: an algorithm to generate 3d isosurfaces based upon tetrahedra, Anais do IX SIBGRAPI (1996) 205-210.

C. H. Chian and J. K. Aggarwal: Model reconstruction and shape recognition from occluding contours", IEEE Trans.on PAMI, Vol.11, pp.372-389, 1989

G.K.M. Cheung, S. Baker and T. Kanade. Visual Hull Alignment and Refinement Across Time: A 3D Reconstruction Algorithm Combining Shape-From-Silhouette with Stereo, *in Proceedings of IEEE Conference on Computer Vision and Pattern Recognition 2003 (CVPR'03)*, Vol. 2, pages 375-382

W. B. Culbertson, T. Malzbender, and G. Slabaugh. “Generalized voxel coloring”. Vision Algorithms: Theory and Practice (Proc. Int. Workshop on Vision Algorithms),volume 1883 of Lecture Notes in Computer Science, pages 100–115. Springer-Verlag, 2000.

C.R. Dyer. Volumetric Scene Reconstruction from Multiple Views. In Foundations of Image Understanding, pages 469–489. Kluwer, Boston, 2001

D. Dobkin, H. Edelsbrunner, C. K. Yap, Probing Convex Polytopes, Proc. 18th ACM Symp. Theory of Computing, 1986, pp. 424-432

F. Durand, C. Puech, "The visibility complex made visibly simple: an introduction to 2D structures of visibility", Proceedings of the eleventh annual symposium on Computational geometry, Vancouver, British Columbia, Canada, 1995

D. Eggert and K. Bowyer," Computing the Perspective Aspect Graph of Solids of evolution,"*IEEE Trans. Pattern Analysis and Machine Intelligence,* vol.15, no.2, pp.109-128, 1993

Elgammal, A., Harwood, D., and Davis, L.S., "Non-parametric Model for Background Subtraction", Proc. of ICCV '99 FRAME-RATE Workshop, 1999.

D. A. Forsyth, "Recognizing algebraic surfaces from their outlines," Int. J. of Computer Vision, vol. 18, no.1, pp.21-40, 1996

J. Franco, E. Boyer, "Exact Polyhedral Visual Hull", Proceedings of BMVC '03

T. Fromherz and M. Bichsel, "Shape from Multiple Cues: Integrating Local Brightness Information," Fourth International Conference for Young Computer Scientist, ICYCS 95, Beijing, P. R. China, 1995.

H. Fuchs, Z. Kedem, B. Naylor, "On visible generation by a priori tree structure", Proc. SIGGRAPH '80, 1980, pp. 39-48

J.J. Gibson, "What is a form?" Psychol. Rev., vol.58, pp. 403-412, 1951

Z. Gigus, J. Canny, and R. Seidel, "Efficiently computing and representing aspect graphs of polyhedral objects", *IEEE Trans. Pattern Analysis and Machine Intelligence,* vol.13, no.6, pp. 542-551, 1991

B. Han, D. Comaniciu, and L. Davis, "Sequential kernel density approximation through mode propagation: applications to background modeling," Proc. ACCV -Asian Conf. on Computer Vision, 2004.

R. Hartley, "In Defence of the 8-Point Algorithm," Proc. Fifth Int'l Conf. Computer Vision, pp. 1064-1070, June 1995

C. Hernández Esteban, F. Schmitt, "Silhouette and Stereo Fusion for 3D Object Modeling", Computer Vision and Image Understanding, vol. 96, no. 3, pp. 367-392, December 2004.

J. Isidoro, S. Sclaroff, "Stochastic Refinement of the Visual Hull to Satisfy Photometric and Silhouette Consistency Constraints", Proceedings of the International Conference on Computer Vision (ICCV), pp 1335--1342, 2003

L. Jaulin and E. Walter: Guaranteed nonlinear set extimation via interval analysis, Bounding Approaches to System Identification, edited by M.Milanese et al, Plenum Press, New York, pp.363-381, 1996

K. Kang, J.-P. Tarel, R. Fishman, and D. B. Cooper, “A linear dual-space approach to 3D surface reconstruction from occluding contours using algebraic surface,” in International Conference on Computer Vision, vol. 1, (Vancouver, Canada), pp. 198 – 204, 2001.

Y.K. Kergosien,”La Famille des Projections Orthogonales d’Une Surface et Ses Singularités,” *C.R. Acad.Sc.Paris*, 292, pp.929-932, 1981

Y. Kim and J. Aggarwal. Rectangular parallelepiped coding: A volumetric representation of three dimensional objects. IEEE Journal of Robotics and Automation, RA-2:127.134, 1986.

J. J. Koenderink and A. J. van Doorn," The Singularities of the Visual Mapping," *Biol. Cybern*. vol.24, pp.51-59, 1976

J. J. Koenderink and A. J. van Doorn," The Internal Representation of Solid Shapes with Respect to Vision," Biol. Cybern. vol.32, pp.211-216, 1979

D.J. Kriegman and J.Ponce, “On recognizing and positioning curved 3D objects from image contours,” IEEE Trans. Pattern Analysis and Machine Intelligence, vol.12, no.12, pp.1127-1137, 1990

K. Kutulakos, S. Seitz, “A theory of shape by space carving”, International Journal of Computer Vision, Vol. 38 (3), 2000, pp. 199-218

M. Langer, W. Zucker, “Shape-from-shading on a cloudy day”, Journal Opt. Soc. Am, vol 11(2), pp. 467-478

A. Laurentini," The Visual Hull Concept for Silhouette-based Image Understanding," IEEE Trans. Pattern Analysis and Machine Intelligence,vol.16, pp.150-162, 1994

A. Laurentini ," How Far 3-D Shapes Can Be Understood from 2-D Silhouettes," IEEE Trans. Pattern Analysis and Machine Intelligence, vol. 17, pp.188-195, 1995

A. Laurentini," How Many 2D Silhouettes it Takes to Reconstruct a 3D Object," Comput. Vision and Image Understanding, vol.67, pp. 81-87, 1997

A. Laurentini, "Computing the Visual Hull of Solids of Revolution," Pattern Recognition, vol. 32, pp.377-388, 1999

Lavakusha et al.: VI algorithms with changing directions of views, *Proc. MIV-89*, Tokyo, pp. 309-314

E. Lazebnik, E. Boyer, J. Ponce, „On how to compute exact visual hulls of object bounded by smooth surfaces“, Proceedings CVPR ,01, vol 1, pp. 156-161

W.E. Lorensen, H.E. Cline, Marching cubes: A high resolution 3D surface construction algorithm, Proceedings of the 14th annual conference on Computer graphics and interactive techniques, 1987, Pages: 163 - 169

W. Martin and J. Aggarwal. Volumetric descriptions of objects from multiple views. IEEE Transactions on Pattern Analysis and Machine Intelligence, 5(2):150.174, March 1983.

W. Matusik, C. Buehler, R. Raskar, S. J. Gortler, and L. McMillan. Imagebased visual hulls. In Proc. SIGGRAPH 2000, pages 369–374, 2000.

W. Matusik, C. Buehler, L. McMillian, „Polyhedral visual hulls for real-time rendering“, Eurographics Workshop on Rendering, 2001

P. Mendonca, K. Wong and R. Cipolla, ”Epipolar geometry from profiles under circular motion,” IEEE Trans. Pattern Analysis and Machine Intelligence, vol. 23, pp.604-616, 2001

A. Montenegro, P. Carvalho, M. Gattass, L. Velho, “Adaptive Space Carving”, Proceedings of 2nd 3dPVT, 2004

R.E. Moore, Methods and Applications of Interval Analysis, Proc. SIAM, 1979

H. Noborio, S. Fukuda, S. Arimoto, “Construction of the octree approximating three-dimensional objects by using multiple views”, *IEEE Trans. on PAMI*, Vol. 10, pp. 769-782, 1988

B. O’Neill, *Elementary differential geometry*, San Diego, Academic Press, 1996

N. M. Oliver, B. Rosario, and A. P. Pentland, “A Bayesian Computer Vision System for Modeling Human Interactions,” IEEE Trans. on Patt. Anal. and Machine Intell., vol. 22, no. 8, pp. 831-843, 2000

OpenCV, Open Computer Vision Library, http://sourceforge.net/projects/opencvlibrary/

S.Petitjean, J.Ponce and D.J.Kriegman, "Computing exact aspect graphs of curved objects: algebraic surfaces," *Int. J. of Computer Vision*, vol. 9, no.3, pp.231-255, 1992

S. Petitjean,” The enumerative geometry of projective algebraic surfaces and the complexity of aspect graphs,” *Int. J. of Computer Vision*, vol. 19, pp.1-29, 1996

S. Petitjean, “A Computational Geometric Approach to Visual Hull”, Int. J. of Comput. Geometry and Appl., vol. 8, no.4, pp. 407-436, 1998

O.A. Platonova, ”Singularities of projections of smooth surfaces,” *Russian Math. Surveys,* . 39, pp.177-178, 1984

M. Potemesil, “Generating octree models of 3D objects from their silhouettes in a sequence of images”, *Comput.Vision Graphics Image Process*., Vol. 40, pp. 1-29, 1987

F. P. Preparata and M. I. Shamos, Computational Geometry: An Introduction, Spinger-Verlag, New York, 1985

J. Rieger, “On the classification of views of piecewise smooth objects,” Image Vis. Comput. Vol.5, pp. 91-97, 1987

J. Rieger, “The geometry of view space of opaque objects bounded by smooth surfaces,” Artificial Intelligence, Vol. 44, pp. 1-40, 1990

J. Rieger,” On the complexity and computation of view graphs of piecewise smooth algebraic surfaces,” Philos. Trans. Roy. Soc. London Ser.A, pp.-1899-1940, 1996

J. Rieger,” Notes on the complexity of exact view graph algorithms for piecewise smooth algebraic surfaces,” Discrete & Comput. Geometry, Vol.20, pp. 205-229, 1998

G. Salmon, “Analytic geometry of three dimensions”, Cambridge England, Metcalfe, 1874

S. Seitz and C. Dyer, “Photorealistic Scene Reconstruction by Voxel Coloring,” Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition, June 1997, pp. 1067- 1073.

J. Shade, S. Gortler, L. He, R. Szeliski, “Layered Depth Images”, Proceedings of SIGGRAPH 98, pp. 231-242.

K. Shanmukh and A.K.Pujari: Volume intersection with optimal set of directions, *Pattern Recognition Letters*, Vol.12, No.3, pp. 165-170, 1991

S. Skiena: Interactive reconstruction via geometric probing, *IEEE Proc.*,Vol.80, No.9, pp1364-1383, 1992

G. Slabaugh, B. Culbertson, T. Malzbender, and R. Schafer. “Improved voxel coloring via volumetric optimization”. Technical Report 3, Center for Signal and Image Processing, Georgia Institute of Technology, 2000.

G. Slabaugh, R. Schafer, M Hans, “Image-based photo hulls for fast and photo-realistic new view-synthesis”, Real Time Imaging, vol. 9, 2003, pp. 346-359

T. Sripradisvarakul and R. Jain,” Generating aspect graphs for curved objects,” *Proc. IEEE. Workshop on Interpretation of 3D Scenes*, pp. 109-115,1989

C. Stauffer, W.E.L. Grimson, “Adaptive background mixture modelsfor real-time tracking”, Proc. of CVPR 1999, pp. 246-252.

R. Szeliski. Rapid octree construction from image sequences. Computer Vision, Graphics and Image Processing: Image Understanding, 58(1):23.32, July 1993.

S. Sullivan, J. Ponce, “Automatic model construction abd pose estimation from photographs using triangular splines”. IEEE Trans. PAMI, 20 (10), pp. 1091-1096, 1998

Wren C, Pentland A (1998) Dynamic models of human motion. Proc. of third IEEE International conference on Automatic Face and Gesture Recognition, April 1998, Nara, Japan: 22-27

E. Trucco and A. Verri, *Introductory techniques for 3D computer vision*, Prentice-Hall, 1998

R. Tsai, “A Versatile Camera Calibration Technique for High-Accuracy 3D Machine Vision Metrology Using Off-the-Shelf TV Cameras and Lenses,” *IEEE Transactions on Robotics and Automation*, Vol. 3, No. 4, Aug. 1987, pp. 323-344.

TsaiCC, Tsai Camera Calibration Codehttp://www-2.cs.cmu.edu/~rgw/TsaiCode.html

M. Yamada, E. Kazuyuki, J. Ohya, A new robust real-time method for extracting human silhouettes from color images, Proceedings of 3rd IEEE International Conference on Face & Gesture Recognition, April 14-16, 1998, Nara, Japan, 528-533

R. Yang, M. Pollefeysy, G Welch, “Dealing with Textureless Regions and Specular Highlights.A Progressive Space Carving Scheme Using a Novel Photo-consistency Measure”, proceedings ICCV 2003

A. J. Yezzi, G. Slabaugh, A. Broadhurst, R. Cipolla, R.W. Schafer, “A Surface Evolution Approach to Probabilistic Space Carving”. Proceedings of 3DPVT 2002, pp. 618-621

Z. Zhang. A Flexible New Technique for Camera Calibration. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 22(11):1330-1334, 2000.

R. Zhang, P-S. Tsai, J.E. Creyerand M. Shah,” Shape from shading: a survey,” *IEEE Transactions on Pattern Analysis and Machine Intelligence*, Vol. 21, no.8, pp.690-706, Aug.1999

J.Y. Zheng: Acquiring 3D models from sequences of contours, IEEE Trans. on PAMI, Vol. 16, no.2, pp.163-178,1994