# A Hybrid Approach for Computing Visual Hulls of Complex Objects

Edmond Boyer and Jean-Sébastien Franco
GRAVIR–INRIA Rhône-Alpes
655, Avenue de l'Europe, 38330 Montbonnot, France
Edmond.Boyer@inrialpes.fr, Jean-Sebastien.Franco@inrialpes.fr

## Abstract

*This paper addresses the problem of computing visual hulls from image contours. We propose a new hybrid approach which overcomes the precision-complexity trade-off inherent to voxel based approaches by taking advantage of surface based approaches. To this aim, we introduce a space discretization which does not rely on a regular grid, where most cells are ineffective, but rather on an irregular grid where sample points lie on the surface of the visual hull. Such a grid is composed of tetrahedral cells obtained by applying a Delaunay triangulation on the sample points. These cells are carved afterward according to image silhouette information. The proposed approach keeps the robustness of volumetric approaches while drastically improving their precision and reducing their time and space complexities. It thus allows modeling of objects with complex geometry, and it also makes real time feasible for precise models. Preliminary results with synthetic and real data are presented.*

## 1. Introduction

Assume we are given several silhouettes of an object corresponding to different camera viewpoints. The visual hull is the maximal solid shape consistent with the object silhouettes. Such an approximation of the object captures all the geometric information available from the object silhouettes. The interest arises, therefore, in all modeling applications making use of silhouettes. In this paper we describe how to efficiently use the silhouette information to compute visual hulls. The motivation is to propose a new practical solution for computing precise models of complex objects, along with a reasonable complexity in time and space.

Visual hulls were first introduced by Laurentini [13] in the theoretical context where an infinite number of viewpoints, surrounding the object's convex hull, are considered. Before and after this work, visual hulls have also been, implicitly and explicitly, widely studied in the computer vision and computer graphics communities. In particular, it has been shown recently [14] that the visual hull of a smooth object is a topological polyhedron that can be determined using weak calibration only (oriented epipolar geometry). However, the solution given in this work does not apply to most real situations. There are many other algorithms for computing approximations of the visual hull in both communities: some consider the volume enclosed by the visual hull and are based on space discretizations; some others focus on the surface of the visual hull and consider individual points or polyhedral representations.

Volumetric approaches are based on space discretizations into elementary cells, the voxels, which are carved according to their image positions with respect to the silhouettes. An early approach was proposed by Martin and Aggarwal [15] who used parallelepipedic cells aligned with the coordinate axis. Later on, octrees were proposed [5] as adaptive data structures for representing visual hulls and efficient approaches [21, 18, 4] were presented to compute voxel-based representations. These approaches are purely geometric and do not consider photometric information. Recent methods [12] make use of such information and carve voxels according to the *color consistency* of their projections onto the different images. See [19] and [9] for reviews on volumetric approaches for modeling. All the aforementioned approaches are based on regular voxel grids and can handle objects with complex geometries. However, the 3D space discretizations used are computationally expensive and lack precision since most of the grid points do not belong to the visual hull surface under consideration.

Surface-based approaches use a different strategy. Visual hull boundary elements, points and faces, are estimated by intersecting the viewing cone surfaces associated with the occluding contours. Baumgart [2] made an early contribution using polygonal approximations of the occluding contours. [11, 6, 3] focused on individual points reconstructed using local second order surface approximations. More recently, approaches have been proposed to compute surface patches[20], or strips [16] of the visual hull. Surface-based approaches can be precise, especially compared to volumetric approaches, however the surface models produced are often incomplete or corrupted, in particular when considering complex objects. A reason for this is the fact that intersections of viewing cone boundaries are generally not well defined, and thus very sensitive to numerical instabilities.

Related to surface-based approaches, Matusik *et al.* [17] have shown that 2D calculations are sufficient when computing new images of an object using its visual hull. This interesting result follows the fact mentioned earlier that the visual hull is a projective structure [14], the approach, however, does not lead to geometric models as required in many applications.

Our approach takes advantages of both categories described above. It uses the robustness of volumetric approaches while keeping the precision of surface-based approaches. A space discretization into cells is still used, but unlike most volumetric algorithms, sample points are not regularly spaced but computed on the surface of the visual hull. Elementary cells are then tetrahedrons coming from the Delaunay triangulation applied on the sample points. The final polyhedral model is obtained by carving these cells according to their image projections. This approach presents two important contributions with respect to the methods mentioned previously: first, the points used to construct the model lie on the surface of the visual hull, thus enabling a high level of precision; second the surface representation is obtained by means of the Delaunay triangulation, hence ensuring robustness, and for which fast implementations exist.

The paper is organized as follows. Section 2 introduces definitions which are used in the paper. Section 3 describes how points on the visual hull are computed. Section 4 details the polyhedral representation algorithm. Experimental results are presented in section 5, before concluding with potential extensions of this work.

## 2. Definitions

**Contours** We assume that a scene, composed of several objects, is observed by a set of pinhole cameras. The objects' surfaces are supposed to be orientable closed surfaces, smooth or polyhedral. No assumption is made on their genus which may be non-zero. *Rims* are the locus of points, on the object surface, where viewing rays are tangent to the surface. Rims project onto image curves, called the *occluding contours* [15], which border the object silhouettes in the image plane. In what follows, subscripts will denote contour numbers and superscripts image numbers, thus $O_j^i$ denotes the $j$th occluding contour in image $i$. Occluding contours are oriented in the images. Their orientation is such that the object is on the left of the oriented contour. Hence, exterior contours are oriented counterclockwise and interior contours are oriented clockwise. We will call the *inside region* of an occluding contour the closed region of the image plane delimited by the contour and containing the silhouette, and we will call the *outside region* its complement in the image plane (see figure 1).

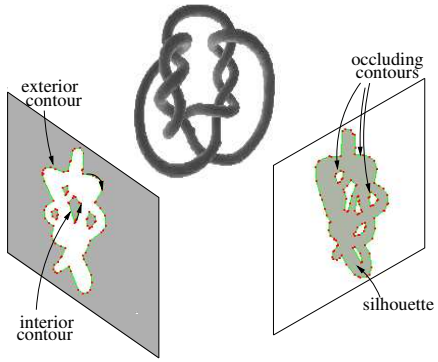



Figure 1: The occluding contours delimit the object silhouette in the image plane. The shaded region on the left image represents image points which are outside at least one contour. Its complement, shaded in the right image, represents image points which are inside all the contours, and thus belong to the silhouette.

**Viewing cones** Intuitively, a viewing cone is a generalized cone whose apex is the image center and whose base is the inside region of an occluding contour. More formally, the *viewing cone* $\mathcal{V}_j^i$ associated with the occluding contour $O_j^i$ is the closure of the set of rays passing through points inside $O_j^i$ and through the camera center of image $i$. $\mathcal{V}_j^i$ is thus tangent to the corresponding object surface along the rim that projects onto $O_j^i$. According to the orientation of $O_j^i$, exterior or interior, the viewing cone $\mathcal{V}$ is an acute or obtuse cone of $\mathbb{R}^3$ respectively. Viewing cone boundaries intersect along space curves which do not lie on the surface, except at *frontier points* where rims intersect. Note that in the case of polyhedral surfaces, frontier points are not necessarily isolated and can form frontier edges.

**Visual hulls** The visual hull is usually defined as the intersection of all the viewing cones available from the different viewpoints, it is thus the closed space region where points project inside all the occluding contours. Let $\mathcal{I}, \mathcal{C}$ be respectively the image set and the contour set under consideration, then:

$$\mathcal{VH}(\mathcal{I}, \mathcal{C}) = \bigcap_{i \in \mathcal{I}, j \in \mathcal{C}} \mathcal{V}_j^i,$$

where $\mathcal{V}_j^i$ is the viewing cone of rim $j$ in image $i$. When a finite set $\mathcal{I}$ of images is considered, the visual hull is a topological polyhedron composed of cone patches delimited by cone intersection curves [14]. In practical situations, occluding contours are approximated by 2D polygonal curves, thus viewing cones are polyhedral cones and visual hulls polyhedrons. The definition above holds if a single object is observed in every image of $\mathcal{I}$. But it can not correctly handle scenes composed of several unconnected objects, some of which may not appear in all images. To this aim, we could straightforwardly extend the definition to the union

of individual visual hulls, each associated to a unique real object. Let $\mathcal{K}$ be the real object set and $\mathcal{C}_k$ be the contour set for the object $k$, then:

$$\begin{aligned}\mathcal{VH}(\mathcal{I},\mathcal{K}) &= \bigcup_{k\in\mathcal{K}} \mathcal{VH}(\mathcal{I},\mathcal{C}_k),\\ &, = \bigcup_{k\in\mathcal{K}} ( \bigcap_{i\in\mathcal{I}_k, j\in\mathcal{C}_k} \mathcal{V}_j^i),\end{aligned} \tag{1}$$

where $\mathcal{I}_k$ is the image subset of $\mathcal{I}$ where object $k$ appears or: $\mathcal{I}_k = \{i \in \mathcal{I} : \mathcal{O}_j^i \neq \emptyset \text{ for some } j \in \mathcal{C}_k\}$. A direct application of this definition requires the sets $\mathcal{C}_k$ and $\mathcal{I}_k$ to be known. In other words, the occluding contours of objects need to be identified over the whole image set. This operation is not necessarily easy, in particular from one image to another. Furthermore, silhouettes might overlap in one image, making the object identification difficult.

Another solution is to define the visual hull as the set of points in $\mathbb{R}^3$ that project inside one silhouette in every image where the points are visible. A first step in that direction is to consider the following expression:

$$\mathcal{VH}(\mathcal{I},\mathcal{K}) = \bigcap_{i\in\mathcal{I}} ( \bigcup_{k\in\mathcal{S}^i} ( \bigcap_{j\in\mathcal{C}_k^i} \mathcal{V}_j^i)), \tag{2}$$

where $\mathcal{S}^i$ is the set of silhouettes in image $i$ and $\mathcal{C}_k^i$ is the set of contours associated to the silhouette $k$ in $i$. This expression is equivalent to (1) applied to a set $\mathcal{K}$ of virtual objects having their silhouettes either disjoint or entirely included in one another in every images. The interest is that objects'x contributions are, in that case, distinguished by their silhouettes. Since any silhouette includes exactly one exterior contour and possibly several interior contours, expression (2) can easily be applied using the exterior contours in the image set.

Nonetheless, expression (2) is not completely satisfying in its current form since it does not take into account the fact that virtual objects may not be seen in one or several images (i.e. $\bigcap_{j\in\mathcal{C}_k^i} \mathcal{V}_j^i = \emptyset$ for some $k$ and some $i$) . As a consequence, they will not be part of the visual hull because they do not appear in one image contribution (see figure 2-(b)). This is due to the fact that the intersection of the image contributions in (2) should be carried out over their common domains. A simpler approach is to consider the complement of the visual hull. It is the the open region $\mathcal{VH}^c$ of $\mathbb{R}^3$ defined by:

$$\mathcal{VH}^c(\mathcal{I},\mathcal{K}) = \bigcup_{i\in\mathcal{I}} ( \bigcap_{k\in\mathcal{S}^i} ( \bigcup_{j\in\mathcal{C}_k^i} \mathcal{D}^i \setminus \mathcal{V}_j^i)), \tag{3}$$

where $\mathcal{D}^i$ is the image $i$ visibility domain in $\mathbb{R}^3$ and $\mathcal{D}^i \setminus \mathcal{V}$ is the complement of $\mathcal{V}$ relative to this domain. Using (3), objects which do not appear in one image can still contribute to the visual hull since empty contributions do not affect image contributions in the above expression. Considering the visual hull or its complement is equivalent since the surface of interest borders both regions, and identifying the cells which belong to the visual hull or to its complement are dual operations. The above expression is in fact the definition that is implicitly used by volumetric approaches when carving voxels. It should be noted that expression (2) could also be modified to account for objects which are not always visible, however using the complement of the visual hull instead of the visual hull itself simplifies both expressions and algorithms.

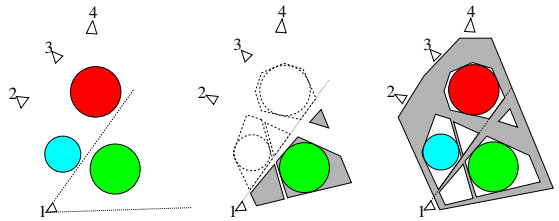



Figure 2: Cross sections of a 4-viewpoints situation: (a) the original scene where camera 1 sees only the green object; (b) expression (2) is used, the visual hull (shaded) does not contain any contribution relative to the blue and red objects; (c) expression (3) is used, the complement of the visual hull (shaded) is computed and includes contributions from the blue and red objects.

Both definitions (2) and (3) may add independent virtual objects that do not appear in the original scene (as shown in figure 2). Notice however that the second definition may add more virtual objects, in particular near or far from projection centers. This is a consequence of the visibility domain constraint which limits the domain of the visual hull complement. The number and sizes of these undesired objects are usually reduced by increasing the number of viewpoints. Another solution, as implicitely used by volumetric approaches, is to use a region of interest instead of $\mathbb{R}^3$.

# 3. Visual hull surface points

## 3.1 Algorithm outline

Assume that occluding contours are extracted in the image set and let us consider a polygonal contour $O_j^i$ in image $i$. Points on the associated viewing cone $\mathcal{V}_j^i$ contribute to the surface of the visual hull if: (i) they project onto $O_j^i$ in image $i$, (ii) they do not project inside the intersection of silhouette complements in any other images. An obvious way to compute these points is therefore to take points on $O_j^i$ and to look at the intersection of their viewing lines with the viewing cones originating from other viewpoints. These intersections define one or several intervals on the viewing line corresponding to the contribution of this viewing line to the surface of the visual hull.

Let $p_j^i$ be a point of $O_j^i$. The contribution intervals on the viewing line of $p_j^i$ are delimited by the intersections of the viewing line with the surfaces of the concerned viewing cones. These intervals can be determined directly in 3D by intersecting lines and cones, however, and as mentioned in [17], most of the calculations can be achieved in 2D. Indeed, points delimiting the intervals on the viewing line of $p_j^i$ are such that their projections belong to both the epipolar line and the concerned occluding contours (see figure 3). We use these principles in algorithm 1 to reconstruct points on the visual hull surface.

**Algorithm 1** Visual hull surface points

1: **for all** contours $O_j^i$ in all images: **do**
2: **for all** images $k$ such that $k \neq i$: **do**
3: **for all** points $p_j^i$ in $O_j^i$ : **do**
4: **compute** the epipolar line $l$ of $p_j^i$ in image $k$,
5: **for all** contours $O_l^k$ in image $k$: **do**
6: **compute** the intersections of $l$ with $O_l^k$,
7: **update** depth intervals along the viewing line of $p_j^i$,
8: **end for**
9: **end for**
10: **compute** the 3D points delimiting intervals along the viewing line of $p_j^i$.
11: **end for**
12: **end for**

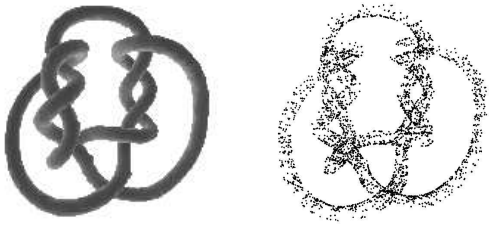
## 3.2 Updating depth intervals along the viewing line

As explained before, intersections of the epipolar line with occluding contours are first computed. From these intersections, we can easily compute the depths of points delimiting intervals along the viewing line, which points belong to the surface of the visual hull. The question is how to combine two lists of depths from different contours or images ?

We proceed in the following way: expression (3) is used to sum up intervals contributing to the visual hull complement, over the contour and image sets. To this aim, the object contributions, or equivalently the silhouettes, must be distinguished in each image. As explained before, this can be done by means of the exterior contours since every one of them identifies a single object. We could therefore group contours, in each image, according to the exterior contour they belong to. A simpler solution takes advantage of the fact that interior contour contributions entirely belong to the visual hull complement. Thus, only the contributions from exterior contours need to be intersected when computing the whole contribution of an image. The corresponding expression for definition (3) becomes:

$$\mathcal{VH}^c(\mathcal{I},\mathcal{K}) = \bigcup_{i\in\mathcal{I}} \left[ \left( \bigcap_{j\in \mathrm{Ext}^i} \mathcal{D}^i \setminus \mathcal{V}_j^i \right) \bigcup \left( \bigcup_{j\in \mathrm{Int}^i} \mathcal{D}^i \setminus \mathcal{V}_j^i \right) \right] \quad (4)$$

where $\mathrm{Ext}^i$ and $\mathrm{Int}^i$ are the sets of exterior contours and interior contours respectively in image $i$. The above expression is equivalent to (3) but it simplifies the function that updates depth intervals. Note here that when applying the above expression, the contribution of a viewing cone complement along the viewing line should be limited to the line interval visible from its image. Figure (4) displays the algorithm result for a synthetic object.



(a) (b)

Figure 4: (a) the *knots* taken from Hoppe's web site [10]; (b) its visual hull surface points for 40 viewpoints located on a circle around the object.

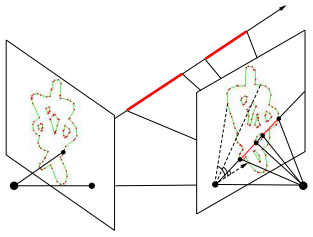


Figure 3: Contribution intervals (in red) to the visual hull surface along the viewing line. Epipolar line angles can be used to accelerate the search for the segments intersecting the epipolar line.

## 3.3 Complexity

Assume that $n$, $m$ and $q$ are the number of images, contours per image and points per contour respectively, then the above algorithm computes $\theta(nmq)$ 3D points in $O(n^2m^2qr)$ time, where $r$ is the upper bound complexity of the line-contour intersection function (step 6 in the algorithm) [1]. A naive implementation leads to $r = O(q)$ if we consider that occluding contours are polygons with $q$ vertices on average. The overall asymptotic complexity would then be $O(n^2m^2q^2)$, or $O(N^2)$ where $N$ is the number of 3D points computed.

[1] We suppose that the number of intersections between the epipolar line and an occluding contour is negligible compared to $n$, $m$ and $q$, we thus expect the function that updates depth intervals to use $O(1)$ time.

Interestingly, the complexity $r$ can be reduced to $O(1)$ by optimizing the intersection function. To this purpose, and between steps 2 and 3 of the algorithm, the image $k$ can be rectified so that epipolar lines become horizontal lines (i.e., the epipolar rectification). In that case, search for intersections between the epipolar line $l$ and the occluding contour $O_l^k$ is simplified by using image ordinates as lookup values. Only the contour segments for which the epipolar line ordinate falls within the vertices' ordinates are to be considered. Equivalently, angles of lines joining the epipole and the contour vertices can also be used as lookup values, with the advantage that image rectifications are not required (see figure 3). Both solutions lead to $r = O(1)$ but add $\theta(n^2mq)$ operations to either rectify image coordinates or compute angle values of contour vertices. Note that in [17, 16] line slopes are used for similar reasons, however slopes do not always partition the image plane in a consistent way. In particular, the slope function is not monotonic when two successive contour vertices lie on both sides of the vertical line incident to the epipole, and hence such a function can not be used for lookup. Using the optimized intersection function, the asymptotic complexity reduces to $O(n^2m^2q)$.

# 4. Visual hull surface

We have shown in the previous section how to compute points on the surface of the visual hull. The next step is concerned with the estimation of the visual hull shape. Classical volumetric approaches consist in carving a partition of the 3D space into regular cells: the voxels. In contrast to this, our space partition lies on the computed visual hull points, and is thus composed of non-regular cells: the Delaunay tetrahedrons. The major advantage is to allow precision at a reasonable cost in time and space complexities.

## 4.1 Point triangulation

The approach we propose is based on the Delaunay tetrahedrization of the visual hull surface points. Delaunay triangulations have been widely used to reconstruct surfaces from unorganized 3D points. Indeed, this problem has received a lot of attention over the last decade and most of the proposed methods consider that the surface solution is included into the Delaunay triangulation of the input points. There are two advantages to the Delaunay triangulation: first, it ensures a regular partition of space in which cells satisfy properties such as having empty circumscribed balls; second fast and robust implementations exist.

The problem we address is similar except that the input data includes, in addition to the 3D points, the 2D image information. Thus, our approach also searches for a subset of the Delaunay triangulation, but the criterion applied to carve, or sculpt, the tetrahedral cells takes this additional information into consideration. In terms of complexity, the Delaunay tetrahedrization is known to have a worst case running time in $O(n^2)$ where $n$ is the number of points. In our case, and as explained in the previous section, the number of 3D points is $\theta(nmq)$ where $n$, $m$ and $q$ are the number of images, contours per image and points per contour respectively. Thus the worst case complexity would be $O(n^2m^2q^2)$, which is more than the time required to compute the visual hull surface points. However, recent works (see [1] for instance) tend to show that the complexity of the Delaunay triangulation for points on a polyhedron is linear in the number of points. This is also confirmed by our experiments which show that most of the running time is spent in the previous step of the algorithm when computing visual surface points. Observe that the overall complexity is therefore not dominated by the Delaunay triangulation.

## 4.2 Surface extraction

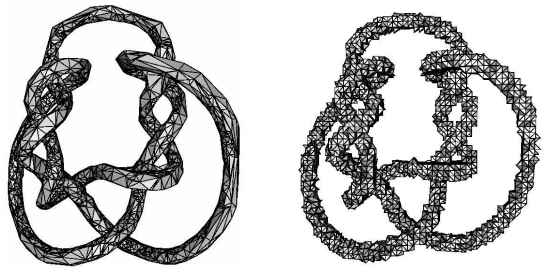
The Delaunay triangulation leads to a set of tetrahedrons which form the convex hull of the set of input points. From this set of tetrahedrons, those contributing to the complement of the visual hull need to be identified and eliminated. A straightforward approach consists in computing the projections of their centroids onto the images and to check whether they lie inside any silhouette. Such an approach is fast if binary images representing background and foreground information are available, which is often the case with silhouette-based applications. It also gives satisfactory results as shown in the next section. Notice that more complex operations involving surface or volume criteria could also be applied. We are currently conducting experiments in that direction. The set of tetrahedrons whose centroids project inside the silhouettes are therefore considered as the visual hull cells. This set is not necessarily bordered by a manifold surface since the elimination may leave isolated tetrahedrons. Such tetrahedrons are detected in a final step where the triangular facets delimiting the visual hull tetrahedrons are identified. The final surface is thus a manifold composed of triangular facets such that all the vertices project onto occluding contours.

Remark also that most of the 3D points computed are naturally grouped pairwise since they delimit intervals. Thus, in addition to the 3D points, the segments defined by these pairs of points also form elements of the visual hull surface and should, therefore, be included into the space partition. To take these new elements into consideration, we have experimented a conforming Delaunay triangulation algorithm [7] which ensures that the triangulation includes any predefined linear complex (edges and faces). However, this algorithm adds a possibly important number of points to the input set, in order to satisfy the edges or faces constraints. Moreover it appears to be very slow and cancels

the interest of a fast triangulation, which is a strong limitation especially in the case of real time applications. We also investigate alternative solutions to enforce these constraints.

# 5. Experimental results

We have experimented the described method on several input sets. A first experiment on the knot object compares our approach with a voxel-like approach. Figure 5 shows results obtained with the same silhouettes (40 images). The boundaries of the voxel grid were chosen close to the object, which is rarely the case in practical situations. Note that results are geometrically better with our approach for a fairly lower complexity. Indeed 3772 points are present in our model while $60^3$ voxels must be verified in each image, not mentioning any surface extraction step, with the voxel approach. The number of images, however, has a linear influence on the upper bound complexity of the volumetric approach while it has a quadratic influence on the upper bound complexity of our approach. This is because the number of images does not affect the space partition with volumetric approaches while it does with our approach. However adding images does not always improve the estimated shape as shown by the next experiments.



(a) (b)

Figure 5: The visual hull surface of the knots: (a) our algorithm result (3772 points reconstructed) (b) a voxel like reconstruction with a $60^3 = 216000$ voxel grid.

The second set of experiments show results on a synthetic torus with cameras randomly distributed on a spherical region surrounding it. Figure 6 displays different visual hulls of the torus obtained with different numbers of points on each contours and different numbers of cameras. Observe that the running time of the algorithm is $O(n^2m^2q)$ where $n$ is the number of images and $q$ the number of points on each contour. Thus adding points on contours has less effect on the running time. Note also that the accuracy of the visual hull decreases surprisingly when the number of images reaches a certain value. This is especially clear in the left column, from 16 to 32 images. Such a behavior is explained by the fact that when adding images, visual hull points closer to the surface are also added. Thus, some of the Delaunay tetrahedrons get closer to the surface and their centroids may project outside some silhouettes. To avoid this behavior, the number of contour points should also be increased when increasing the number of images. Interestingly, this suggests that there is an optimal ratio between the number of images and the number of points on the contours.

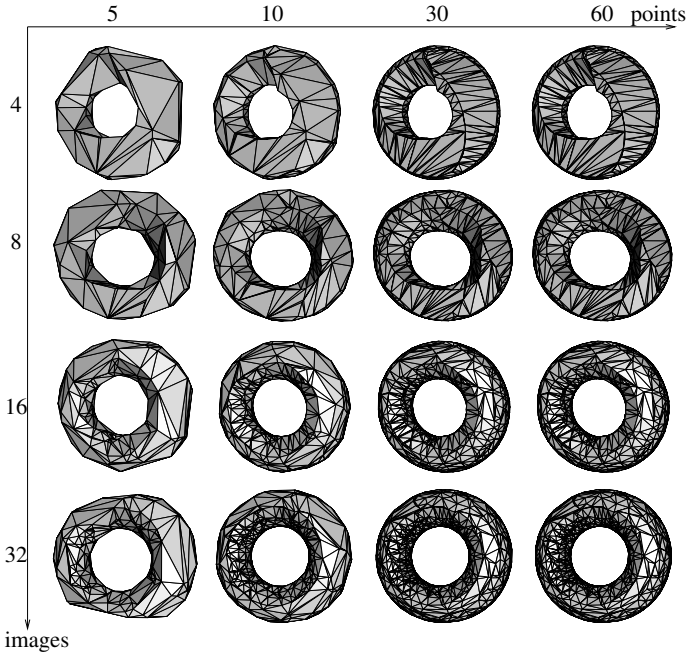



Figure 6: Visual hulls of a torus: the number of points per contour versus the number of images. Points are regularly sampled on the contours and images are randomly chosen on a sphere surrounding the torus.

Figure 7 shows results obtained with real objects. Contours are extracted in the images using the optimal algorithm described in [8]. The human example is interesting since it shows virtual objects as explained in section 2. We are also experimenting the same scenario with a cluster of PCs in real time situations, our preliminary results show that real time computations of fairly precise models can be expected.

# 6. Conclusion

In this paper, we have presented an approach for computing the visual hull of complex scenes when silhouettes are available. Our main contribution is to propose a hybrid algorithm which takes advantage of both volumetric approaches and surface-based approaches. The algorithm first computes points on the surface of the visual hull, and second extracts the visual hull surface from a Delaunay triangulation. The first step is achieved by computing points delimiting the visual hull complement. The second step is achieved by

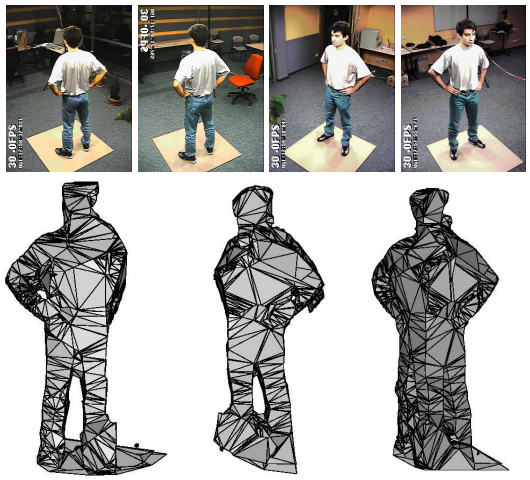


Figure 7: The visual hull of a person from 4 viewpoints. It contains all the geometric information available from the silhouettes. Nevertheless, note, in the right view, the virtual legs that are added to the person. This is due to the camera positions in this particular case and illustrates the point we made in section 2

taking the surface delimiting the polyhedrons that project inside the silhouettes. We have shown that our approach is equivalent to volumetric approaches for efficiency. They are both based on the same definition for visual hulls and they both use all the geometric information available from the silhouettes. However, we have also shown that our approach gives significantly better results in terms of precision, together with lower time and space complexities. The resulting reconstruction method is naturally aimed at real time modeling applications.

We are currently working on further improvements and applications of our method. First, tetrahedrons are classified according to the positions of their centroid projections in the images. This can be improved by applying other elimination schemes. Second, the Delaunay triangulation is applied on points only, however there are more information about the visual hull yet to be used, namely the contribution intervals along viewing lines. The final model could therefore be improved by including these intervals. Third, real time implementations of modeling methods are still a challenging issue, in particular when considering a cluster of PCs.

# References


[1] Dominique Attali and Jean-Daniel Boissonnat. A linear bound on the complexity of the delaunay triangulation of points on polyhedral surfaces. Rapport de recherche 4453, INRIA, 2002.

[2] B.G. Baumgart. A polyhedron representation for computer vision. In *AFIPS National Computer Conference*, 1975.

[3] E. Boyer and M.-O. Berger. 3D surface reconstruction using occluding contours. *IJCV*, 22(3):219–233, 1997.

[4] K.M. Cheung, T. Kanade, J.Y. Bouguet, and M. Holler. A real time system for robust 3d voxel reconstruction of human motions. In *CVPR'00*, volume 2, pages 714 – 720.

[5] C.H. Chien and J.K. Aggarwal. Volume/surface octress for the representation of three-dimensional objects. *CVGIP*, 36(1):100–113, 1986.

[6] R. Cipolla and A. Blake. Surface Shape from the Deformation of Apparent Contours. *IJCV*, 9:83–112, 1992.

[7] David Cohen-Steiner, Eric Colin de Verdière, and Mariette Yvinec. Conforming Delaunay triangulations in 3d. In *Proc. 18th Annu. ACM SoCG*, 2002.

[8] I. Debled-Rennesson and J.P. Reveillès. A linear algorithm for segmentation of digital curves. *IJPRAI*, 9(4):635–662, 1995.

[9] C.R. Dyer. Volumetric Scene Reconstruction from Multiple Views. In L.S. Davis, editor, *Foundations of Image Understanding*, pages 469–489. Kluwer, Boston, 2001.

[10] H. Hoppe, T. DeRose, T. Duchamp, J. McDonald, and W. Stuetzle. Surface reconstruction from unorganized points. In *SIGGRAPH*, volume 26(2), pages 71–78, July 1992.

[11] J.J. Koenderink. What Does the Occluding Contour Tell us About Solid Shape? *Perception*, 13:321–330, 1984.

[12] K. Kutulakos and S. Seitz. A Theory of Shape by Space Carving. *IJCV*, 38(3):199–218, 2000.

[13] A. Laurentini. The Visual Hull Concept for Silhouette-Based Image Understanding. *IEEE Transactions on PAMI*, 16(2):150–162, February 1994.

[14] S. Lazebnik, E. Boyer, and J. Ponce. On How to Compute Exact Visual Hulls of Object Bounded by Smooth Surfaces. In *CVPR'01*, volume I, pages 156–161.

[15] W.N. Martin and J.K. Aggarwal. Volumetric description of objects from multiple views. *IEEE Transactions on PAMI*, 5(2):150–158, 1983.

[16] W. Matusik, C. Buehler, and L. McMillan. Polyhedral Visual Hulls for Real-Time Rendering. In *Eurographics Workshop on Rendering*, 2001.

[17] W. Matusik, C. Buehler, R. Raskar, S. Gortler, and L. McMillan. Image Based Visual Hulls. In *Siggraph*, pages 369–374, 2000.

[18] W. Niem. Automatic Modelling of 3D Natural Objects from Multiple Views. In *European Workshop on Combined Real and Synthetic Image Processing for Broadcast and Video Production, Hamburg, Germany*, 1994.

[19] G. Slabaugh, B. Culbertson, T. Malzbender, and R. Schafe. A Survey of Methods for Volumetric Scene Reconstruction from Photographs. In *International Workshop on Volume Graphics*, 2001.

[20] S. Sullivan and J. Ponce. Automatic Model Construction, Pose Estimation, and Object Recognition from Photographs using Triangular Splines. *ICCV'98*.

[21] R. Szeliski. Rapid Octree Construction from Image Sequences. *CVGIP*, 58(1):23–32, 1993.