Proceedings of the IEEE Workshop on Applications of Computer Vision, pages 234-241, Breckenridge, Colorado, January 2005.

Visual Hull Construction Using Adaptive Sampling

Ali Erol
Computer Vision Lab.
University of Nevada,
Reno, NV 89557

George Bebis
Computer Vision Lab.
University of Nevada,
Reno, NV 89557

Richard D. Boyle
BioVis Lab
NASA Ames Research Center
Moffett Field, CA 94035

Mircea Nicolescu
Computer Vision Lab.
University of Nevada,
Reno, NV 89557

Abstract

Volumetric visual hulls have become very popular in many computer vision applications including human body pose estimation and virtualized reality. In these applications, the visual hull is used to approximate the 3D geometry of an object. Existing volumetric visual hull construction techniques, however, produce a 3-color volume data that merely serves as a bounding volume. In other words it lacks an accurate surface representation. Polygonization can produce satisfactory results only at high resolutions. In this study we extend the binary visual hull to an implicit surface in order to capture the geometry of the visual hull itself. In particular, we introduce an octree-based visual hull specific adaptive sampling algorithm to obtain a volumetric representation that provides accuracy proportional to the level of detail. Moreover, we propose a method to process the resulting octree to extract a crack-free polygonal visual hull surface. Experimental results illustrate the performance of the algorithm.

1. Introduction

3D reconstruction of a moving object from arbitrary multiple views, using synchronized image sequences, has received considerable attention over the last few years [8]. A technique that supports robust and real time, reconstruction of an object's volume is based on the idea of intersecting the cones obtained by back-projecting the silhouettes of the object, from different viewpoints. The resulting volume is often called the visual hull of the object with respect to the available views [11].

Despite the fact that the visual hull is not an exact reconstruction, it has been used extensively in VR (virtual reality) applications [10] [13] [15] and human body pose estimation [8] [19] [5] [20] [14]. In the case of pose estimation, a 3D human model is usually fit to the visual hull to estimate pose parameters. In VR applications, the purpose is to create new poses of a person's body in a virtual environment. In these cases, the visual hull is textured using color information available in the images and rendered directly at the desired view points.

One approach to visual hull construction is calculating a polygonal surface by intersecting silhouette cones, computed easily by back-projection polygonal approximations of silhouette contours of the object. Since direct 3D intersections suffer from numerical instabilities, recent studies have improved robustness by calculating the intersections on 2D image planes and back-projecting the results [13]. From an application point of view, a polyhedral surface representation of the visual hull might be very practical for texturing and rendering purposes. However, it might not be practical for other tasks such as collision detection [10] in VR applications, or model fitting [5] in pose estimation applications.

Volumetric reconstruction represents an alternative approach to visual hull construction [18]. In this case, the 3D space is divided into elementary cubic elements (i.e., voxels) and tests are performed to label each voxel as being inside, outside or on the boundary of the visual hull. This is done by checking the contents of its projections on all the available binary silhouette images. The output of volumetric methods is either an octree [6] [18], whose leaf nodes cover the entire space or a regular 3D voxel grid [8].

The most important advantage of volumetric-based approaches is their robustness. A clear disadvantage is the high processing power requirements due to the volumetric processing, even in the case of octree-based implementations [6]. Another drawback, which is the main issue addressed in this study, is the lack of an accurate surface representation. The output is a 3-color voxel data representation that merely serves as a bounding volume. This voxel-based representation, called Voxel-Based Visual Hull (VBVH) in the rest of the paper, can provide only rough surface approximations [10]. Polygonization of VBVH can yield satisfactory results only at high enough resolutions, which is definitely not desirable in a real-time application.

In this study, our purpose is to enhance the volumetric visual hull by replacing the voxel-based representation with an octree-based irregular grid of sampled data, which is also common to many computer graphics applications [9]. The octree samples an implicit surface representation, derived from the implicit contours of silhouettes. Given an octree-based VBVH [18] it is straight forward to build an octree that samples the implicit surface visual hull regularly at a narrow band around the

surface (See Section 3). Regular sampling may still be redundant for approximation purposes. In contrast here we will apply a fully irregular sampling strategy that keeps the resolution proportional to the level of detail on the silhouettes.

2. Implicit Surface Visual Hull

To derive an implicit surface of the visual hull, we need to represent the silhouettes as implicit contours. One can imagine many forms of implicit representations but we have chosen to use the signed distance function. One reason for this choice is practical: In most studies, the silhouettes are represented in the form of a binary image obtained through a background subtraction procedure [10] and there exists fast algorithms for calculating the approximate distance transforms of the silhouettes. Moreover, there exist some slow but sub-pixel accurate segmentation algorithms (e.g. geodesic snakes [7]) that output signed distance functions. Another reason is the existence of a geometric interpretation of the signed distance function.

Given the signed distance function of each silhouette, it is possible to calculate the signed distance function of the corresponding silhouette cone. Specifically, given a point in 3D space, the closest point on the cone surface lies on the viewing ray of the silhouette contour point closest to its projection. This is an elegant but expensive way of constructing an implicit silhouette cone, therefore, we follow a much cheaper method, which simply propagates the signed distance along the viewing rays of each point on the image. Assuming $S_i(\mathbf{p}) : \Re^2 \rightarrow \Re$ denotes the signed distance function corresponding to the i'th camera silhouette, then each silhouette cone $C_i(\mathbf{x}) : \Re^3 \rightarrow \Re$ can be calculated to be

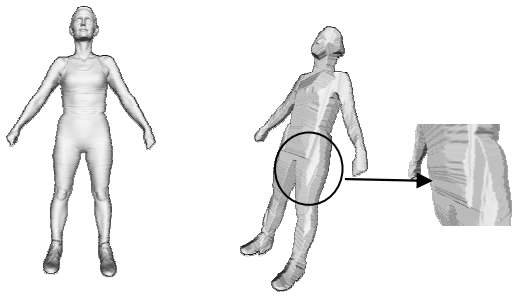
$$C_i(\mathbf{x}) = S_i(\mathrm{P}_i(\mathbf{x})) \qquad (1)$$

where $\mathrm{P}_i(\mathbf{x})$: $\Re^3 \rightarrow \Re^2$ denotes the perspective projection function of the i'th camera. The resulting silhouette cone $C_i(\mathbf{x})$ may also be viewed as a distance function where the distances are measured on the image plane. For visualization purposes, we did not observe any differences between the two silhouette cone calculation methods, however in the case of more detailed processing like model fitting operations the results may be very different.

Assuming that the implicit function is negative inside the silhouette, the intersection of silhouette cones, which we will be called ISVH (Implicit Surface Visual Hull) in the rest of the paper and denoted as $V(\mathbf{x})$, can be calculated to be

$$V(\mathbf{x}) = \max_i C_i(\mathbf{x}) \qquad (2)$$

In principle, $V(\mathbf{x})$ can be processed directly without going through an explicit visual hull construction stage because the geometry of the visual hull is fully captured by the implicit function. It is possible to derive the normal and higher order derivatives at an arbitrary point using the derivatives of Eq. (2) except at the C^1 discontinuities introduced by the maximization operation. All the derivatives are determined by the location of the point, the camera parameters, and the derivatives of the implicit contours. Such a processing scheme requires parallelization and acceleration of projection calculations.



(a) (b)

Figure 1: Direct implicit visual hull polygonization: (a) the synthetic model used to generate silhouettes on 6 cameras, (b) High resolution polygonization result.

Figure 1 demonstrates the visual hull surface extracted from ISVH using the polygonizer provided in [3]. Figure 1(a) shows a synthetic 3D human body model*, which is used to obtain the silhouettes in OpenGL. The silhouettes are first obtained in binary form and then the signed distance function is calculated. An accurate Euclidean distance transform is usually very expensive to calculate so the Chamfer 3,4 metric [16], which can approximate Euclidean distance using integers, is utilized. Subpixel distances are calculated using bilinear interpolation. Figure 1(b) shows a high-resolution surface extraction output. It is interesting to note that the silhouette cone intersections produced some sharp features on the resulting surface. Moreover, the discrete distance transform produces some ripples on the surface.

3. Octree-Based Regular Sampling

Direct processing of the ISVH may not be very practical in certain applications due to the fact that we need to perform projections on a possibly large number of silhouettes. An approximation, independent of the silhouettes, would be more useful for processing purposes. A straightforward way to achieve this is by sampling Eq. (2) on a regular grid using a small enough step to avoid aliasing. Then, tri-linear interpolation may be employed to obtain a continuous approximation. A drawback of using a regular

* Publicly available at Stanford Computer Graphics Laboratory (http://graphics.stanford.edu/ data/3Dscanrep).

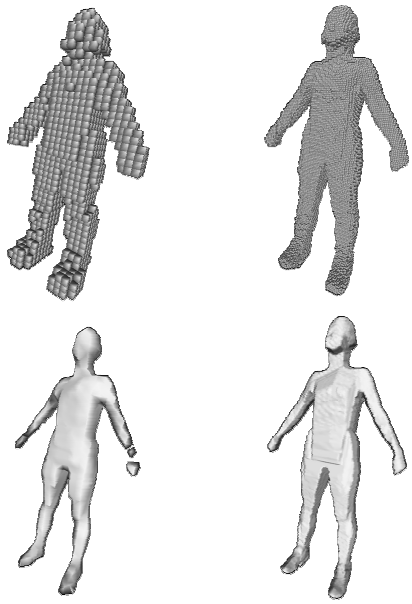
grid is that time complexity becomes proportional to the volume of the grid. Nevertheless, there exist studies that apply regular sampling on the VBVH (e.g., [21] presents a prototype of a real-time, hardware accelerated, system). Another approach might be limiting the accuracy to a narrow band around the surface of the visual hull. In this case, the octree-based visual hull algorithm proposed in [18] can be applied without any modifications.

```
bool ShouldSplit(Voxel v)
{
     int Type = INSIDE;
     for all images I
     {
          Projection P= Projection(I,v);
          if P is outside the silhouette
          {
               Type = OUTSIDE;
               return FALSE;
          }
          else if P encloses silhouette
                              boundary
          {
               Type = BOUNDARY;
          }
     }

     return Type == BOUNDARY;
}

OctreeVisualHull()
{
     Initialize the octree to a single voxel;
     for d=0 to MaximumDepth
     {
          for all voxels v at depth d
          {
               if( ShouldSplit(v) and
                              d<MaximumDepth)
                    Split(v);
          }
     }
}
```

Figure 2: The octree-based visual hull construction algorithm.

In the algorithm (See Figure 2), an octree, which is initially a single node corresponding to the whole bounding box, is refined iteratively up to a predetermined maximum depth by splitting some of the leaf nodes. At each iteration, new voxels at the leaf nodes are classified into one of the following three types: (1) *inside voxels***,** whose projections lie inside all the silhouettes, (2) *outside voxels*, whose projections lie entirely outside in at least one silhouette and (3) *boundary voxels*, whose projections contain the boundary of the silhouette in some of the images. Splitting is applied only to boundary voxels. The boundary leaf nodes in the resulting octree, found at maximum depth, enclose the surface of the visual hull; therefore, time complexity is proportional to the area of the visual hull. A parallel version of this algorithm is presented in [6]. In this case, the sequential algorithm is run over a single image for each silhouette in parallel. Then, the results (i.e., octree representations of the silhouette cones) are intersected using a distributed process.

It should be noted that although the octree performs a non-regular sampling of the bounding volume, we are still referring to it as regular sampling since the surface of the visual hull is captured by a regularly sampled narrow band, made up of maximum resolution nodes (i.e., leafs) in the octree. Computing voxel projections and checking whether they overlap with the silhouettes are critical computational steps, affecting the processing speed of both algorithms. Since calculating and checking exact voxel projections (i.e., 6 or 4 sided polygons) are very expensive, they are approximated usually by their bounding rectangles. This approximation allows fast processing but generates false boundary voxels.



(a) (b)

Figure 3: The octree-based VBVH (Top) and ISVH (Bottom) using octrees with a maximum depth of (a) 6, (b) 8.

Sampling the ISVH requires two further processing stages after octree construction:

1. **Silhouette Cone Intersection:** This stage corresponds to the evaluation of Eq. (2) at the vertices of the leaf nodes of the octree.
2. **Polygonization:** Marching cubes (MC) [12] look-up table is used on the boundary voxels containing zero crossings.

The surface of the VBVH and approximate ISVH corresponding to Figure 1 are shown in Figure 3. Figure 3(a) shows a very low resolution construction. The main difference between the VBVH and the ISVH is the distortion due to aliasing. Point sampling causes loss of visual hull

volume, which violates the bounding volume property of the VH. VBVH does not suffer from aliasing; it is still possible to extract a surface that is guaranteed to enclose the object but it may not approximate small features well (i.e. the arm and wrist). Figure 3(b) shows the output of a higher resolution processing. The resolution is lower than that of Figure 1(b).

4. Adaptive Sampling

The outputs shown in Figure 3 illustrate that different parts of the visual hull surface can be represented reliably at different resolutions. The octree-based sampling described in Section 3 results in a regularly sampled narrow band around the surface, which may still be redundant for approximation purposes. A more efficient alternative is applying an adaptive sampling strategy by considering the fidelity of the representation at each resolution level and stopping when satisfactory results have been obtained.

There already exist many adaptive implicit surface sampling algorithms [4]. It is possible to apply an existing algorithm to sample ISVH but all of the algorithms perform 3D calculations, which require direct processing of the ISVH. A useful property of the visual hull is the reduction of 3D operations to 2D operations on the image planes as in recent surface-based visual hull construction algorithms [13]. Exploiting this property can help to reduce the computational requirements.

The adaptive sampling algorithm requires modifying the three main modules of the regular sampling algorithm described in Section 3:

1. **Octree Construction:** In the regular sampling scheme, every boundary voxel is refined. In the adaptive sampling, only the boundary voxels that do not satisfy an accuracy constraint are split. The Local Feature Size (LFS) function [1] is utilized to derive an adaptive sampling criterion. One important point to mention is that the criterion considers the fidelity of the sampling with respect to each silhouette cone independently. Handling the silhouette cones independently supports a potential parallel implementation similar to the one given in [6].
2. **Silhouette Cone Intersection:** The first stage results in an octree approximating the silhouette cones separately; however, this does not guarantee that the intersection will be a good approximation for the ISVH. In fact, it is possible to loose visual hull volume during intersections. Such situations are detected and corrected in this module.
3. **Polygonization:** A direct application of MC to the leaf nodes produces a surface with cracks. After calculating the MC triangles, the cracks are detected and repaired in this module.

In the following sub-sections, each of these modules are described in detail.

4.1. Octree Construction

When sampling an implicit surface adaptively, local geometric properties of the surface can be used to measure the fidelity of sampling [4]. If we consider a polygonized silhouette cone (e.g., using the algorithm presented in Section 2) where the projections of its vertices are points on the silhouette curve, then the boundary of the projection of the cone is a polygon whose vertices sample the original silhouette curve. In this case, a reasonable adaptation rule is controlling the difference between the polygon boundary and the silhouette curve.

In a generic algorithm, the intersections between the projection of a voxel and the silhouette can be considered as samples. Then, the line segments formed by the samples can be checked to see if they approximate the curve inside the projection.

In a direct approach, the maximum distance between the curve and its polygonal approximation can be used as a measure of accuracy. A faster approach is calculating the maximum curvature on the silhouette curve segment inside the projection of a voxel but it is difficult to estimate curvature robustly. Instead, we have chosen to use the LFS function [1].

LFS is a quantitative measure of level of detail. Given a LFS function, a smooth curve F is called r-sampled ($r \leq 1$) by a set of points S if every $p \in F$ is within a distance $rLFS(p)$ of a sample $s \in S$. The r-sampling of a curve results in a sampling strategy where the sample density is kept proportional to the level of detail. In [1], $LFS(p)$ of a point $p \in F$ is defined to be the distance from a point $p \in F$ to the closest point on the medial axis (MA) of the curve.

A way to make use of r-sampling in adaptive silhouette cone sampling is by checking if the curve inside the projection satisfies the r-sampling criterion for a fixed value of r. If the condition is satisfied for all the voxels on the surface, then the silhouette curve should be r-sampled by the vertices of the silhouette polygon of the cone.

Performing exact r-sampling tests for each voxel requires many calculations similar to the maximum distance measure. Instead, a rough measure based on the minimum LFS inside the projection of the voxel is used: Let d be the length of the diagonal of the bounding rectangle of the projection of a voxel and let m be the minimum LFS value inside the rectangle. Then, boundary voxels satisfying the condition $m \geq \alpha d$ for some parameter α are not split further. Intuitively, the condition keeps the diagonal of the rectangle proportional to the level of detail. In this way, small features are captured by small voxel projections. From the r-sampling point of view, the parameter α enforces an upper bound on the r value.

The adaptive sampling criterion explained above requires some extra post-processing after the distance transform step. First, the MA should be extracted [16]. Then, the LFS of the samples on the curve should be calculated using the chamfer distance transform of MA. The outputs of the pre-processing stages are shown in Figure 4. We use kd-trees to find the minimum LFS value inside the bounding rectangle of the projection of a voxel.

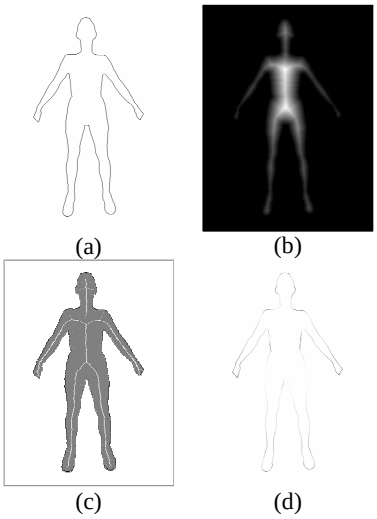


Figure 4: Processing stages of binary silhouettes: (a) silhouette Curve, (b) chamfer distance transform, (c) medial axis, (d) LFS function.

4.2. Silhouette Cone Intersection

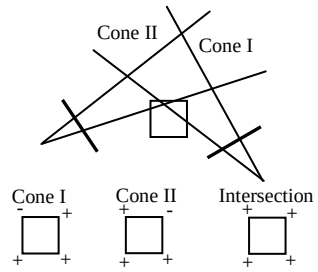


Figure 5: Silhouette cone intersection problem in 2D: Intersection operation results in loss of visual hull area.

After the octree has been constructed, the ISVH should be sampled at the vertices of the leaf nodes of the octree. When the silhouette cones are intersected using Eq. (2), there is a possibility to loose visual hull volume. This problem is demonstrated in the case of a 2D visual hull in Figure 5. Such situations are detected by checking alternating signs on each edge of the voxel. Then, the voxel is split until the lost edge surface intersections are recovered. Since the octree from the previous stage is a good sampling of each silhouette cone, the implicit function values at new vertices are calculated by interpolating existing silhouette cone values. In this way, the expensive step of accessing the silhouette images is avoided.

4.3. Polygonization

When the octree nodes are polygonized independently, some cracks arise because of differences in the resolution between the neighboring octree leafs. This is not a problem in the case of octree-based regular sampling because the neighbors of a voxel on the surface have the same resolution. In the adaptive octree however, the cracks should be removed for visualization purposes.

There are many studies on the polygonization of octrees. Our crack removal procedure has been inspired by [17] and [2], both of which make use of MC-based polygonization. When the voxels are polygonized using the MC look-up table, some triangles, each of which has an edge lying on one of the faces of the voxel, are generated. The edges on voxel faces correspond to the intersection of the face with the surface. When a low-resolution voxel has high resolution neighbors, cracks occur (See Figure 6 for examples). To make things easier, we first get rid of the voxels containing faces with ambiguous sign configurations by splitting the voxel until the ambiguities disappear. Although, It is actually possible to handle ambiguous faces, it requires more complicated processing. Moreover we observed that LFS-based adaptation rule does not generate too many ambiguous faces. When ambiguous faces are eliminated, all the faces of boundary voxels contain at most one triangle edge corresponding to the surface intersection. In this case, two types of cracks are detected by the polygonization algorithm:

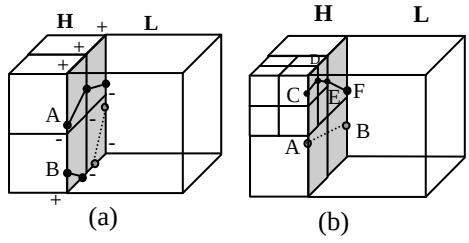


Figure 6: Two types of cracks detected during polygonization.

- **Type I** (See Figure 6 (a)): This type is characterized by the existence of multiple surface intersections on the same edge of the low resolution voxel. For example, A and B in the figure are generated by high resolution voxels and lie on the same edge of the low-resolution voxel. It should be noted that they can not be detected by the low-resolution voxel itself.

- **Type II** (See Figure 6(b)): In the figure high-resolution voxels generate the contour CDEF shown with solid lines on the low-resolution face. Low-resolution voxel generates a single edge AB shown with dashed lines.

The first type of crack is not repaired; instead the low resolution voxel is split until the crack disappears or reduces to one or more Type II crack, which is easy to repair. To repair Type II cracks, it is enough to replace the edge of the low-resolution triangle by the corresponding contour generated by the high resolution voxels. For example in Figure 6(c), the edge AB is replaced by the curve CDEF. When the replacement is made, the triangle generating the edge AB is converted to a closed polygon. Then the centroid of the polygon is calculated and a fan of triangles from the centroid to the polygon vertices is generated.

5. Experimental Results

Execution time and the size of the resulting octree represent good quantitative measures for evaluating the algorithms tested here. Another important measure is the accuracy of the visual hull surface. Projecting every point on the visual hull surface should lie on the boundary of at least one of the silhouettes. Given the ISVH defined in Eq. (2) the *projection error* for a point **x** on the reconstructed surface can be calculated as

$$E(\mathbf{x}) = \left|V(\mathbf{x})\right| \qquad (3)$$

The mean of the projection error for samples of points on the polygonal surface will be used as a quantitative measure of accuracy. In error calculation, the sample points are taken from the triangles on the surface and the number of samples per triangle is kept proportional to the area of each triangle.

The first three rows of Table 1 show some measurements obtained by running the regular sampling algorithm on the silhouettes used in Figure 3 at different resolutions (shown in the first column). The second column shows the time spent (measured on 2.53GHz Pentium 4 with Windows-XP) on signed distance function calculations, which is an overhead compared to the algorithms operating directly on the binary silhouette. For the outputs shown in Figure 3, six 640x480 binary silhouettes were used. The table shows the total time spent for processing all the silhouettes. The third column shows the execution time for the whole octree construction process. It is the total time spent on the octree construction and silhouette cone intersection. The fourth column shows the time spent on polygonization, which is another overhead compared to VBVH. An important observation is the exponential increase in time requirements due to the exponential increase in the visual hull surface area (reported in the fifth column of the table in terms of boundary voxels in the resulting octree). The last column shows the mean projection error.

Table 1: Performance measures of different visual hull construction algorithms.

Res.	Preproc. (Seconds)	Octree (Seconds)	Surface (Seconds)	Bound. Voxels	Proj. Error
D=7	0.05	0.19	0.02	5365	0.23
D=8	0.05	0.52	0.09	21887	0.11
D=9	0.05	1.84	0.50	89169	0.05
α=0.3	0.14	0.18	0.05	5354	0.30

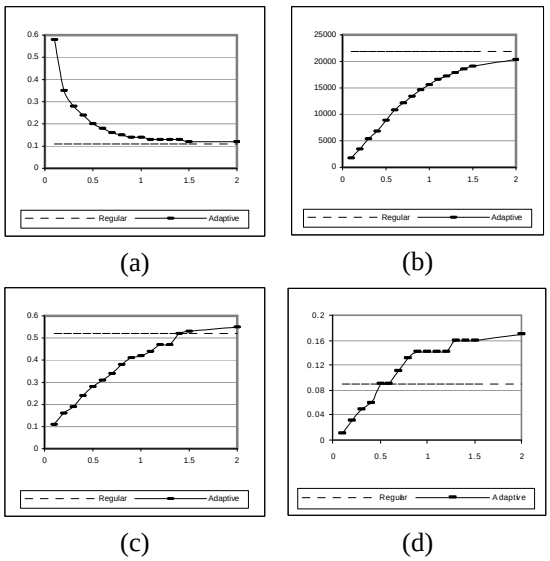
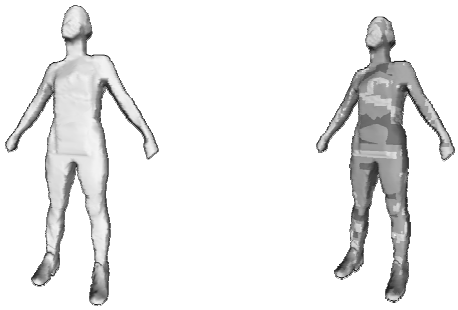


Figure 7: Numerical results for adaptive sampling algorithm. (a) mean projection error, (b) number of boundary voxels in the octree, (c) octree construction time, (d) surface extraction time.

There are two parameters in the case of the adaptive sampling algorithm: i) The maximum depth constraint and ii) the value of the sampling parameter α. In a practical setting, the depth constraint is expected to be present in any application, mainly due to the processing speed requirements (i.e. the minimum resolution that satisfies other requirements of the application would be the choice for the maximum depth). Moreover the existence of camera calibration errors and segmentation errors can also make very high resolution processing unnecessary; there is no benefit in reconstructing noise. In a general setting the complexity of the object's shape and/or the physical size of the details that can be safely segmented from the images (for example we do not expect correct segmentation of the fingers in a typical human body reconstruction scenario) can be used to derive an upper bound on the resolution. When maximum depth is fixed, α becomes a

somewhat finer measure of resolution that determines the quality of the adaptive sampling.

Figure 7 shows the processing speed, number of boundary voxels, and the projection error obtained by running the adaptive sampling algorithm on the silhouettes shown in Figure 3, using different values of α. The maximum depth of the octree is set to 8 by considering the visual and numerical results obtained for the regular sampling algorithm (note that the maximum depth constraint does not apply in the polygonization phase since refinement is the only solution for removing some type of cracks). The horizontal axis in all the graphs shows the

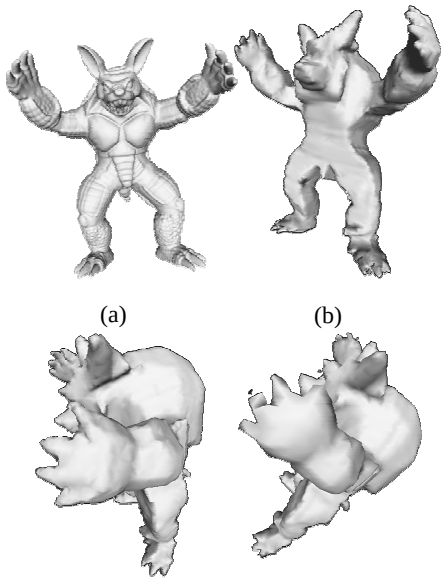


(a) (b)

Figure 8: (a) the output of the adaptive sampling algorithm with α =0.3, (b) the resolution map on the surface.

value of α. The dashed lines show the corresponding values for the regular sampling algorithm with a maximum depth 8 (See row 2 in Table 1). It can be observed that depending on the value of α, it is possible to obtain a faster and more memory efficient system by sacrificing accuracy (i.e., the projection error). The last row of Table 1 shows the numerical results for α=0.3. An important overhead is the extra processing required by the adaptive sampling algorithm. The preprocessing overhead is roughly tripled but an efficient parallelization in the preprocessing stage is possible. In fact, a speed-up factor of 6 (6 silhouettes are used) is reasonable to expect in case of a parallel implementation. In terms of other performance measures, adaptive sampling stands very close to the regular sampling with maximum depth 7 (row 1). When compared to regular sampling at depth 8 (row 2) it is much faster and much more memory efficient but the projection error is larger. Note that we can not expect an inverse behavior in the projection error as the adaptive sampling algorithm produces a much smaller number of boundary voxels than regular sampling (i.e. roughly 4 times). Our purpose in using the adaptive sampling algorithm is mainly to preserve small features of the object by keeping the resolution proportional to the level of detail. We assume that a less accurate approximation is acceptable for larger parts of the object.

The output of the adaptive sampling algorithm for α=0.3 is shown in Figure 8. Figure 8(a) is visually similar to the regular sampling output shown in Figure 3(b). Figure 8(b) shows the resolution map on the surface of the visual hull. The resulting construction consists of voxels at depth 6, 7 and 8. It can be seen that high resolution voxels are used mostly for the finer body parts, which means that the fine structures on the body are approximated properly.



(a) (b)

(c) (d)

Figure 9: The visual hull of another model. (a) Synthetic model, (b) the output of adaptive sampling, (c) a closer look at the claw of the armadillo, (d) the visual hull obtained using regular sampling.

To illustrate the importance of preserving the details, the algorithm was run on another synthetic model shown in Figure 9(a). The sampling parameter α was set to 0.3, and the maximum depth was chosen to be 8. Figure 9(b) shows the output and Figure 9(c) shows the claw, which contains fine details. Figure 9(d) shows the same claw in the case of regular sampling with a maximum depth of 7. Figure 9(d) clearly shows that regular sampling at low resolution introduces distortions and the details of the claws are lost due to aliasing. The adaptive algorithm, however, does not sacrifice accuracy at these fine details. In a sense, it makes use of higher resolution only when necessary.

Finally we present experimental results using real data (See Figure 10). Figure 10-a shows the adaptively sampled visual hull of a walking person using four cameras. Figure 10-b shows the adaptively sampled visual hull of a hand using eight cameras. In both cases maximum depth of the octree was chosen to be 8 and α was set to 0.3. The

maximum resolution corresponds to voxels sizes of 8.6mm and 2.0mm for the human body and the hand respectively. It can again be verified that the finer parts of the objects are captured by the highest resolution voxels.

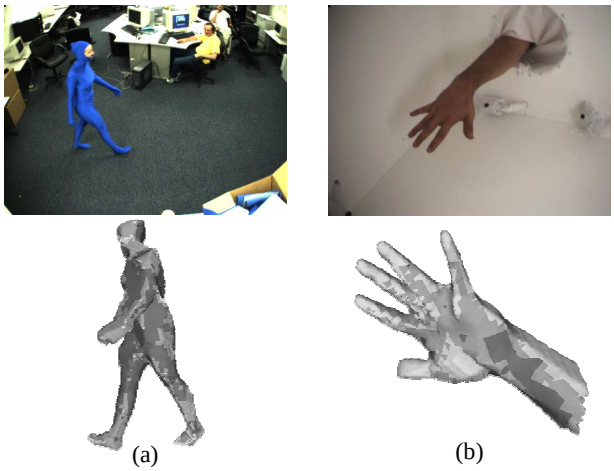


Figure 10: Adaptive sampling results on real data. a) A walking person (top) and her ASVH, b) A hand (top) and its ASVH. The color of the visual hull surfaces is kept proprotional to the resolution.

6. Conclusions

We have extended the conventional VBVH to implicit surfaces using the implicit representations of the input silhouettes. The ISVH is a model of higher quality and can capture the geometry of the visual hull without sacrificing robustness. The higher representation power of ISVH allowed us to design an octree-based adaptive sampling algorithm. The adaptive sampling strategy targets a volumetric representation that provides accuracy proportional to the level of detail. In human body reconstruction, this corresponds to a volume model representing smaller body parts (e.g., hands) with higher accuracy. In the octree-based BVH construction algorithm [18], the only parameter of fidelity is the desired resolution which may be too restrictive in a real time environment. The adaptive sampling algorithm provides a finer fidelity parameter that preserves the details. This should be desirable both in human body pose estimation and VR applications.

Acknowledgments: This research was supported by NASA under grant # NCC5-583.

References

[1] Amenta N., Bern M., Eppstein D., “The crust and the β-skeleton: Combinatorial curve reconstruction,” Graphical Models and Image Processing, 60/2 (2), pp. 125–135, March 1998.

[2] Ashida K., Badler N., "Feature preserving manifold mesh from an octree." Solid Modeling and Applications 2003, pp. 292-297.

[3] Bloomenthal J., "An Implicit Surface Polygonizer", In Graphics Gems IV 324-349, 1994

[4] Bloomenthal J., “Introduction to Implicit Surfaces”, Morgan Kaufmann Publishers, Inc. San Francisco, CA. 1997

[5] Borovikov E., Davis L., "3D Shape Estimation Based on Density Driven Model Fitting", Intl. Symposium on 3D Data Processing, Visualization and Transmission, 2002

[6] Borovikov E., Davis L., “A Distributed System for Real-Time Volume Reconstruction”, Fifth IEEE International Workshop on Computer Architectures for Machine Perception (CAMP'00), Padova, Italy, September 11 - 13, 2000.

[7] Caselles V., Coll B., “Snakes in Movement” SIAM Journal on Numerical Analysis, 33:2445-2456, December 1996

[8] Cheung K. M., Kanade T., Bouguet J., Holler M.,” A real time system for robust 3D voxel reconstruction of human motions”, CVPR 2000, Vol. 2, June, 2000, pp. 714 – 720, 2000.

[9] Frisken, S.F.; Perry, R.N.; Rockwood, A.P.; Jones, T.R., "Adaptively Sampled Distance Fields: A General Representation of Shape for Computer Graphics", ACM SIGGRAPH, ISBN: 1-58113-208-5, pps 249-254, July 2000

[10] Hasenfratz Jean-Marc, Lapierre M., Gascuel J.-D., Boyer E., “Real-Time Capture, Reconstruction and Insertion into Virtual World of Human Actors.” VVG'03, 2003

[11] Laurentini A., “The Visual Hull Concept for Silhouette-Based Image Understanding”, PAMI, Vol. 16, No. 2, 1994

[12] Lorensen W. E., Cline, H. E., "Marching Cubes: a high resolution 3D surface reconstruction algorithm," Computer Graphics, Vol. 21, No. 4, pp 163-169, 1987

[13] Matusik W., Buehler C., McMillan L., "Polyhedral Visual Hulls for Real-Time Rendering", Eurographics Workshop on Rendering, 2001.

[14] Mikic I., Hunter E., Cosman P., Trivedi M., "Articulated Body Posture Estimation from Multi-Camera Voxel Data", IEEE CVPR 2001, Kauai, Hawaii, December 11-13, 2001

[15] Moezzi S., Katkere A., Kuramura D. Y., Jain R., “Reality modeling and visualization from multiple video sequences”, IEEE Computer Graphics and Applications, 16(6):58--63, November 1996.

[16] Remy E., Thiel E., "Medial Axis for Chamfer Distances: Computing Look-Up Tables and Neighbourhoods in 2D or 3D", Pattern Recognition Letters, 23(6):649-661, April 2002.

[17] Shekhar R., Fayyad E., Yagel R., Cornhill J. F., "Octree-Based Decimation of Marching Cubes Surfaces", Conference on Visualization, pp. 335-344, IEEE, October 27- November 1 1996.

[18] Szeliski R., "Rapid Octree Construction from image sequences", CVGIP: Image Understanding, Vol. 58, No. 1, pp. 149-156, 1993.

[19] Theobalt C., Magnor M., Schueler P., Seidel H. P., “Combining 2D Feature Tracking and Volume Reconstruction”, Pacific Graphics 2002, Beijing, China. p.96-103, 2002.

[20] Villa-Uriol M., Sainz M., Kuester F. and Bagherzadeh N., “Automatic creation of three-dimensional avatars”, SPIE and IS&T's Electronic Imaging (EI 2003), Santa Clara, California, USA, January 2003.

[21] Wada T., Wu X., Tokai S., Matsuyama T., “Homography Based Parallel Volume Intersection: Toward Real-Time Volume Reconstruction Using Active Cameras”, Computer Architectures for Machine Perception, pp.331-339, 2000