Carved Visual Hulls for Image-Based Modeling

Yasutaka Furukawa[1] and Jean Ponce[1,2]

[1] Department of Computer Science
University of Illinois at Urbana Champaign, USA
[2] Département d'Informatique
Ecole Normale Supérieure, Paris, France
{yfurukaw,ponce}@cs.uiuc.edu

Abstract. This article presents a novel method for acquiring high-quality solid models of complex 3D shapes from multiple calibrated photographs. After the purely geometric constraints associated with the silhouettes found in each image have been used to construct a coarse surface approximation in the form of a visual hull, photoconsistency constraints are enforced in three consecutive steps: (1) the rims where the surface grazes the visual hull are first identified through dynamic programming; (2) with the rims now fixed, the visual hull is carved using graph cuts to globally optimize the photoconsistency of the surface and recover its main features; (3) an iterative (local) refinement step is finally used to recover fine surface details. The proposed approach has been implemented, and experiments with six real data sets are presented, along with qualitative comparisons with several state-of-the-art image-based-modeling algorithms.

1 Introduction

This article addresses the problem of acquiring high-quality solid models [3] of complex three-dimensional (3D) shapes from multiple calibrated photographs, a process dubbed *image-based modeling*. A popular approach to image-based modeling is to acquire multiple depth maps with a laser range scanner, register them, and merge them into a single 3D model [4, 8, 14]. The relative accuracy of laser-based systems can be as high as 1/10,000 [14]. Comparable (and even higher) accuracy levels have been achieved using "ordinary" cameras in the close-range photogrammetry domain [22]. However, photogrammetric methods typically measure a rather sparse set of point (a few hundreds) and require markers. The accuracy levels currently achieved by automated, markerless approaches to image-based modeling from calibrated photographs (e.g., [7, 10, 11, 15, 18, 23, 20]) are much lower. They are rarely quantified, often because of a lack of ground truth data, but it is probably fair to say that relative accuracies of about 1/200 are the state of the art. As a step toward higher accuracy, we present in this paper a method that combines the geometric and photometric constraints associated with multiple calibrated photographs to recover accurate solid object models in the form of *carved visual hulls* (see [7, 23, 20] for related approaches). The proposed algorithm has been implemented, and experiments with six real data sets are presented. As in previous studies,

[3] In the form of watertight surface meshes, as opposed to the partial surface models typically output by stereo and structure-from-motion systems.

Fig. 1. Overall flow of the proposed approach. Top: one of the 24 input pictures of a toy dinosaur (left), the corresponding visual hull (center), and the rims identified in each strip using dynamic programming (right). Bottom: the carved visual hull after graph cuts (left) and iterative refinement (center); and a texture-mapped rendering of the final model (right). Note that the scales on the neck and below the fin, as well as the undulations of the fin, are recovered correctly, even though the variations in surface height there is well below 1mm for this object about 20cm wide.

the lack of ground truth data has prevented us (so far) from conducting a quantitative assessment of the proposed method, but the qualitative results presented in Figs. 1 and 7 demonstrate the recovery of very fine surface details in all our experiments. Our technique also appears to fare rather well in preliminary —and once again qualitative— comparisons with several state-of-the-art image-based modeling algorithms (Fig. 8).

1.1 Background

Several recent approaches to image-based modeling attempt to recover *photoconsistent* models that minimize some measure of the discrepancy between the different image projections of their surface points. *Space carving* algorithms represent the volume of space around the modeled object by a grid of voxels, and erode this volume by carving away successive layers of voxels with high discrepancy [11, 18]. In contrast, *variational methods* explicitly seek the surface that minimize image discrepancy. Variants of this approach based on *snakes* iteratively deform a surface mesh until convergence [7, 21]. *Level-set* techniques, on the other hand, implicitly represent surfaces as the zero set of a time-varying volumetric density [6, 9]. The *graph cuts* global optimization technique can also be used to avoid local extrema during the search for the optimal surface [16, 23, 20]. The last broad class of image modeling techniques is the oldest one: The *visual hull*, introduced by Baumgart in the mid-seventies [1], is an outer approximation of the observed solid, constructed as the intersection of the visual cones associated with all input cameras. Many variants of Baumgart's original algorithm have also been proposed (e.g., [13, 15, 19]).

1.2 Approach

Hernández Esteban and Schmitt propose in [7] to use the visual hull to initialize the deformation of a surface mesh under the influence of photoconsistency constraints expressed by gradient flow forces [24] (see [9] for a related approach combining geometric and photometric approaches). Although this method yields excellent results, its reliance on snakes for iterative refinement makes it susceptible to local minima. In contrast, Vogiatzis, Torr and Cipolla use the visual hull to initialize the global optimization of a photometric error function [23]. The results are once again impressive, but silhouette consistency constraints are ignored in the minimization process, which may result in excessive carving. In fact, they add an *inflationary ballooning* term to the energy function of the graph cuts to prevent the over-carving, but this could still be a problem, especially in high-curvature regions (more on this in Section 5.2).

To overcome these problems, we propose in this paper a combination of global and local optimization techniques to enforce both photometric and geometric consistency constraints throughout the modeling process. The algorithm proposed by Lazebnik [13] is first used to construct a combinatorial mesh description of the visual hull surface in terms of polyhedral *cone strips* and their adjacency relations (see next section and [13] for details). Photoconsistency constraints are then used to refine this initial and rather coarse model while maintaining the geometric consistency constraints imposed by the visual hull. This is done in three steps: (1) the *rims* where the surface grazes the visual hull are first identified through dynamic programming; (2) with the rims now fixed, the visual hull is carved using graph cuts to globally minimize the image discrepancy of the surface and recover its main features, including its concavities (which, unlike convex and saddle-shape parts of the surface, are not captured by the visual hull); and (3) iterative (local) energy minimization is finally used to enforce both photometric and geometric constraints and recover fine surface details. While geometric constraints have been ignored in [23] in the global optimization process, our approach affords in its first two steps an effective method for enforcing hard geometric constraints during the global optimization process. As demonstrated in Section 5.2, the third step, similar in spirit to the local optimization techniques proposed in [7, 9], remains nonetheless essential in achieving high-quality results. The overall process is illustrated by Fig. 1, and the rest of this paper details each step and presents our implementation and its results, along with preliminary comparative experiments.

2 Identifying Rims on Visual Hull Surfaces

2.1 Visual Hulls, Cone Strips, and Rims

Let us consider an object observed by n calibrated cameras with optical centers $O_1, \ldots, O_n$, and denote by γ_i its apparent contour in the image I_i (Fig. 2(a)). The corresponding *visual cone* is the solid bounded by the surface Φ_i swept by the rays joining O_i to γ_i.[4] Φ_i grazes the object along a surface curve, the *rim* Γ_i. The *visual hull* is the solid formed

[4] We assume here for simplicity that γ_i is connected. As shown in Section 5, our algorithm actually handles apparent contours made of several nested connected components.

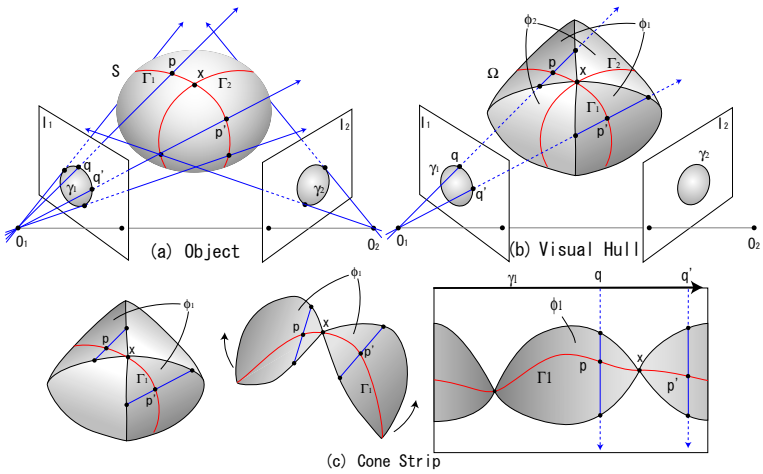

Fig. 2. A visual hull, cone strips and rims: (a) an egg-shaped object is viewed by 2 cameras with optical centers O_1 and O_2; the point x is a frontier point; (b) its visual hull is constructed from two apparent contours γ_1 and γ_2, the surface Ω of the visual hull consisting of two cone strips ϕ_1 and ϕ_2; (c) the cone strip ϕ_1 associated with the first image I_1 is stretched out along the apparent contour γ_1, so a point q on γ_1 corresponds to a vertical line in the right part of the diagram.

by the intersection of the visual cones, and its boundary can be decomposed into a set of *cone strips* ϕ_i formed by patches from the cone boundaries that connect to each other at *frontier points* where two rims intersect (Fig. 2(b)). As illustrated by Fig. 2(c), each strip can be mapped onto a plane by parameterizing its boundary by the arc length of the corresponding image contour. In this figure, a viewing ray corresponds to a vertical line inside the corresponding strip, and, by construction, there must be at least one rim point along any such line (rim points are identified in [3] by the same argument, but the algorithm and its purpose are different from ours). Once the visual hull and the corresponding cone strips have been constructed using the algorithm proposed in [13], the next step is to identify the rim that runs "horizontally" inside each strip (Fig. 2(c)). Since rim segments are the only parts of the visual hull that touch the surface of an object, they can be found as the strip curves that minimize some measure of image discrepancy. The next section introduces such a measure, similar to that used in [6].

2.2 Measuring Image Discrepancy

Let us consider a point p on the visual hull surface. To assess the corresponding image discrepancy, we first use z-buffering to determine the images where it is visible, then select among these the τ pictures with minimal foreshortening. Next, a $\mu \times \mu$ grid is overlaid on a small patch of the surface's tangent plane at p, and τ $\mu \times \mu$ tangent plane "windows" $h_1, \cdots, h_\tau$ are retrieved from the corresponding input images. We normalize the intensity of each window h_i and compute the sum of squared differences (SSD) for each pair. Our final discrepancy measure is thus: $f(p) = \frac{2}{\tau(\tau-1)\mu^2} \sum_{i=1}^{\tau} \sum_{j=i+1}^{\tau} \mathrm{SSD}(h_i, h_j)$. $\tau = 5$ and $\mu = 11$ in all our experiments.

2.3 Identifying a Rim in a Cone Strip

As noted earlier, the image discrepancy function should have small values along rims, thus these curves can be found as shortest paths within the strips, where path length is determined by the image discrepancy function. In our visual hull implementation, a cone strip ϕ_i is represented by the undirected graph G with its polyhedral vertices V and edges E, and it is straightforward to find the shortest path by dynamic programming. However, the idealized situation in Fig. 2 rarely occurs in practice, and the rim may not be a continuous curve in its cone strip (Fig. 3(a)): As shown in [13], the boundaries of the cone strips often loose their singularities (frontier points) to measurement errors, resulting into multiple connected components. In practice, we can still apply dynamic programming to each connected component independently. Harder problems arise from the facts that (1) there may be multiple strip components intersecting the same vertical line (we call them *vertical neighbors*), with the rim being in any one of these; and (2) the rim can be discontinuous at any point inside the strip due to T-junctions. In this work, we assume for simplicity that rim discontinuities occur only at the following two types of strip vertices (Fig. 3(b)): an *opening* vertex v_o whose neighbors v' all verify $v_o \prec v'$, and a *closing* vertex whose neighbors v' all verify $v' \prec v_c$, where "$\prec$" denotes the circular order on adjacent vertices in G induced by the closed curve formed by the apparent contour. Under this assumption, dynamic programming can be still used to find the rim as a shortest path in the *directed* graph G' with vertices V and edges E', defined as follows. Firstly, for each edge (v_i, v_j) in E, we add to E' the *Horizontal* edge (v_i, v_j) if $v_i \prec v_j$, and the edge (v_j, v_i) otherwise. Secondly, to handle discontinuities, we add to E' the *Vertical* directed edges linking each opening (resp. closing) vertex to all vertices immediately following (resp. preceding) it in its vertical neighbors (Fig. 3(c)).

Next, we assign weights to edges in a directed graph G'. For horizontal edges, a weight is the physical edge length multiplied by the average image discrepancy of its two vertices. Vertical edges have weight 0. Then, we decompose the graph G' into connected components (Fig. 3(d)), and use dynamic programming to find the shortest path between the leftmost (start) vertex of each component and its rightmost (goal) vertex. At times, rim discontinuities may occur at other points than those selected by our assumptions. Accordingly, the simple approach outlined above may misidentify parts of the rim. Since the rims are used as hard constraints in the next global optimization step, we want to avoid false positives as much as possible. Among all the vertices identified as the rim points, we filter out false-positives by using the image discrepancy score $f(v)$ and the vertical strip size $g(v)$ at a vertex v. More concretely, a vertex v is detected as a false-positive if either $R/3 < g(v)$ or $R/15 < g(v)$ and $\eta < f(v)$ hold, where R is an average distance from all the vertices V' in the mesh to their center of mass $\sum_{v \in V'} v/|V'|$. η is a threshold for the image discrepancy score, and is selected for each data set in our experiments. Image discrepancy values are blurred along the identified rims before this filtering. Note that when the vertical strip size is small (at most $R/15$), there is little ambiguity in the location of the rim, and the corresponding vertex automatically passes the test according to the above rule.

The next two sections show how to carve the visual hull by combining photoconsistency constraints with the geometric *rim consistency* constraints associated with the

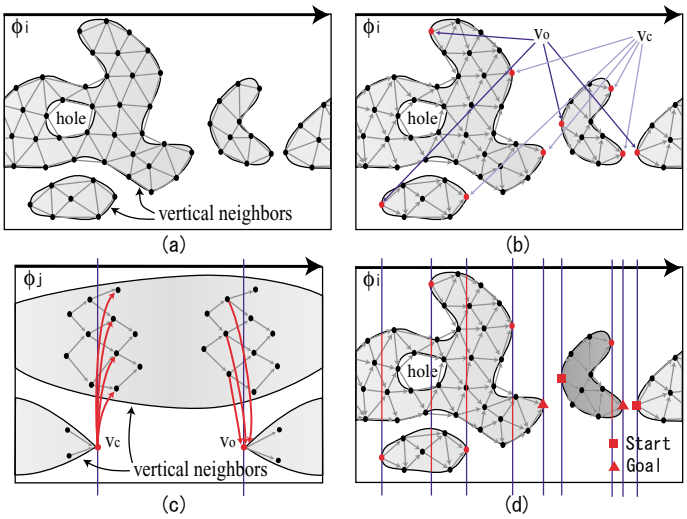

Fig. 3. (a) An undirected graph representing a cone strip ϕ_i. The two leftmost components are vertical neighbors. (b) The opening and closing vertices v_o and v_c of ϕ_i. (c) Illustration of the vertical edge creation process for a different strip ϕ_j. (d) After the horizontal and vertical edges of the directed graph G' associated with ϕ_i have been created, G' is split into two connected components, shown here in different shades of grey, with unique start and goal vertices each.

identified rim segments. We start with a *global optimization* step by graph cuts to recover main surface features. A *local refinement* step is then used to reveal fine details.

3 Global Optimization

In this part of our algorithm, rim consistency is enforced as a hard constraint by fixing the location of the identified rim segments, which split the surface Ω of the visual hull into k connected components $\overline{G}_i$ $(i = 1, \ldots, k)$ (note that the rim segments associated with a single strip may not form a loop, so each graph component may include vertices from multiple strips). To enforce photoconsistency, we independently and iteratively deform the surface of each component $\overline{G}_i$ *inwards* to generate multiple layers forming a 3D graph, associate photoconsistency weights to the edges of this graph, and use graph cuts to carve the surface. [5] The overall process is summarized in Algorithm 1 and detailed in the next two sections.

3.1 Deforming the Surface

To construct the graph associated with each component $\overline{G}_i$ of the visual hull boundary, we first deform the surface *inwards* (remember that the visual hull is an *outer*

[5] The graph associated with a voxel grid serves as input in typical applications of graph cuts to image-based modeling (e.g., [2, 10, 16, 23]). The surface deformation scheme is proposed here instead to take advantage of the fact that the visual hull is already a good approximation.

Algorithm 1 Carving $\overline{G}_i$ with graph cuts.

$J \leftarrow \overline{G}_i$; {$J$ will contain ρ layers of the mesh.}
for $j = 2$ to ρ **do**
  for $k = 1$ to λ **do** {Apply λ deformation steps to $\overline{G}_i$.}
    for each vertex $v \in \overline{G}_i$ except for the boundary **do**
      $v \leftarrow v - \frac{\varepsilon}{\lambda}(\zeta_1 \ f(v) + \zeta_2)\mathbf{N}(v) + \mathbf{s}(v)$;
    end for
  end for
  $J \leftarrow J \cup \overline{G}_i$; {Add a layer.}
end for
Add vertical, horizontal, and diagonal edges to J, and compute their weights;
Use graph cuts to find a minimum cut in J.

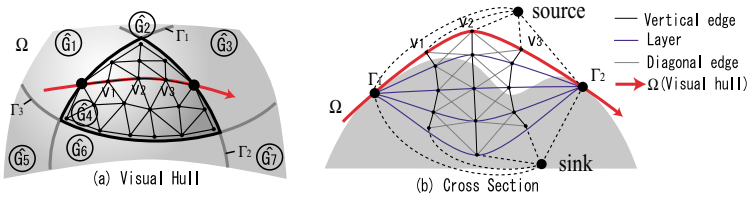

Fig. 4. Deforming the surface for graph cuts: (a) the surface Ω of the visual hull is decomposed into multiple independent components $\overline{G}_i$; (b) the deformation process is illustrated for the cross section of $\overline{G}_4$ that contains vertices v_1, v_2, and v_3.

object approximation) to create multiple offset layers. Note that the photoconsistency function is evaluated at all the vertices in each layer, and their surface normals are estimated by using the corresponding layer. At every iteration, we move every vertex v in $\overline{G}_i$ (except for the boundaries) along its surface normal $\mathbf{N}(v)$, and apply smoothing: $v \leftarrow v - \frac{\varepsilon}{\lambda}(\zeta_1 \ f(v) + \zeta_2)\mathbf{N}(v) + \mathbf{s}(v)$, where $\varepsilon, \zeta_1, \zeta_2$ are scalar constants, $f(v)$ is the image discrepancy function defined earlier, $\mathbf{N}(v)$ is the unit surface normal, and $\mathbf{s}(v)$ is a smoothness term of the form $-\beta_1 \Delta v + \beta_2 \Delta\Delta v$ suggested in [5]. λ iterations are performed to generate each layer, and at total ρ layers are generated during the deformation process. Note that using $f(v)$ yields an adaptive deformation scheme: the surface shrinks faster where the image discrepancy function is larger, which is expected to provide better surface normal estimates. We use $\zeta_1 = 100$, $\zeta_2 = 0.1$, $\beta_1 = 0.4$, $\beta_2 = 0.3$, $\rho = 30$, and $\lambda = 20$ in all our experiments, which have empirically given good results for our test objects. On the other hand, ε, which determines an offset between adjacent layers, should depend on the depth of a surface from the visual hull boundary, and is set manually for each object, typically to about 0.3 times the average edge length in $\overline{G}_i$.

3.2 Building a Graph and Applying Graph Cuts

After a set of layers J has been created, three types of edges are added, as shown in Fig. 4. Vertical edges connect the offset instances of the same vertex in adjacent layers, horizontal edges connect vertices in the same layer, and diagonal edges connect vertices

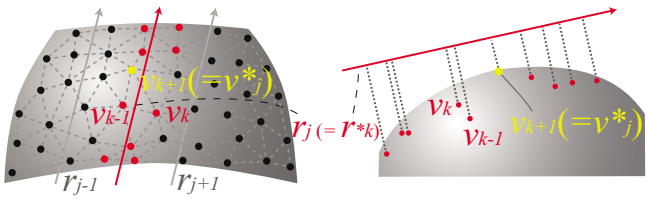

Fig. 5. The rim consistency force is computed for a viewing ray r_j, then distributed to all the vertices V_j whose closest ray is r_j. Here v_{k+1} is the closest vertex v^*_j to r_j.

in adjacent layers. As before, photoconsistency values are computed at all the vertices in J, and a simple variant of the technique proposed in [2] is used to compute edge weights. Concretely, the weight of an edge (v_i, v_j) is computed as $w_{ij} = \frac{\alpha(f(v_i)+f(v_j))(\delta_i+\delta_j)}{d(v_i,v_j)}$, where $f(v_i)$ is the photoconsistency function value at a vertex v_i, $d(v_i, v_j)$ is the length of the edge, and δ_i is a measure of the sparsity of vertices around v_i, approximated by the average distance to the adjacent vertices. Intuitively, weights should be large where vertices are sparse. We use $\alpha = 1, 10$, and 5 for horizontal, vertical, and diagonal edges, respectively, in all our experiments, which accounts for the fact that edges are not uniformly distributed around a vertex. Lastly, we connect all the vertices in the top (resp. bottom) layer to the source (resp. sink) node with infinite edge weights.

3.3 Practical Matters

For the global optimum provided by graph cuts to be meaningful, the edge weights must accurately measure the photoconsistency, which in turn requires good estimates of the normals in the vicinity of the actual surface. For parts of the surface far from the visual hull boundary, normal estimates computed at each vertex from neighbors in the same layer may be inaccurate. In practice, this suggests applying the surface deformation and graph cuts procedure to each component of the graph $\overline{G}_i$ several times, each iteration improving the accuracy of the normals and of the photoconsistency function, and therefore the quality of its global optimum. Note that after the pure inward deformation of the first iteration, the mesh is allowed to deform both inwards and outwards —while remaining within the visual hull— along the surface normals. Empirically, three iterations have proven sufficient to recover the main surface features in all our experiments.

4 Local Refinement

In this final step, we iteratively refine the surface while enforcing all available photometric and geometric information. At every iteration, we move each vertex v along its surface normal by a linear combination of three terms: an image discrepancy term, a smoothness term, and a rim consistency term. The image discrepancy term is simply the first derivative of $f(v)$ along the surface normal. The smoothness term is the same as in the previous section. The rim consistency term is similar to the one proposed in [7]: Consider an apparent contour γ represented by a discrete set of points q_j together

with the corresponding viewing rays r_j. We add rim consistency forces to vertices as follows (Fig. 5): Let us define $d(v_k, r_j)$ as the distance between the vertex v_k and a viewing ray r_j; we find the *closest* viewing ray $r_k^* = \text{argmin}_{r_j} d(v_k, r_j)$ to every vertex v_k. Next, if V_j denotes the set of all the vertices v_k whose closest viewing ray is r_j (i.e., $r_k^* = r_j$), we find the vertex v_j^* in V_j closest to r_j (i.e., $v_j^* = \text{argmin}_{v_k \in V_j} d(v_k, r_i)$). Note that a surface satisfies the rim consistency conditions if and only if $d(v_j^*, r_j) = 0$ for all viewing rays r_j. Therefore, we add an appropriately weighted force whose magnitude is proportional to $\overline{v_j^* r_j}$ to all vertices in V_j, where $\overline{v_k r_j}$ is the *signed* distance between the vertex v_k and a viewing ray r_j, with a positive sign when the projection of v_k lies inside the contour γ and negative otherwise. Concretely, we add to the vertex v_k in V_j the force $\mathbf{r}(v_k) = \overline{v_j^* r_j} \frac{\exp(-(\overline{v_k r_j} - \overline{v_j^* r_j})^2 / 2\sigma^2)}{\sum_{v_{k'} \in V_j} \exp(-(\overline{v_{k'} r_j} - \overline{v_j^* r_j})^2 / 2\sigma^2)} \mathbf{N}(v_k)$, where $\mathbf{N}(v_k)$ is the unit surface normal in v_k. The basic structure of the algorithm is simple. At every iteration, for each vertex v, we compute three terms and move v along its surface normal by their linear combinations: $v \leftarrow v + \mathbf{s}(v) + \mathbf{r}(v) - \kappa \nabla f(v) \cdot \mathbf{N}(v)$. κ is a scalar coefficient and is set depending on the object and the resolution of the mesh. After repeating this process until convergence—typically from 20 to 40 times, we remesh and increase the resolution, and repeat the same process until the image projections of the edges in the mesh become approximately 2 pixels in length. Typically, the remeshing operation is performed three times until the mesh reaches the final resolution.

5 Implementation and Results

5.1 Implementation

We have implemented the proposed approach in C++. The bottleneck of the computation is the global optimization and the local refinement steps, each of which takes about two hours for our large data sets such as the first toy dinosaur, the toy mummy, and the two human skulls, with a 3.0 GHz Pentium 4. The remaining steps including the visual hull construction and the rim identification take at most twenty minutes. Note that we have assumed so far that a single apparent contour is extracted from each input image. In fact, handling multiple nested components only requires a moderate amount of additional bookkeeping, whose description is omitted here for brevity. Note also that our algorithm does *not* require all silhouette holes to be found in each image: For example, silhouette holes are ignored for the human model shown in Fig. 7, while the apparent contour components associated with holes are explicitly used for the human skull models. In practice, the surface of an object may not be Lambertian. We identify and reject for each patch the input images where it may be highlighted by examining the mean intensity and color variance. The chain rule is used to compute the derivative of $f(v)$ along the surface normal as a function of image derivatives, which in turn are estimated by convolving the input images with the derivatives of a Gaussian function. Finally, the topology of an object's surface is not necessarily the same as that of its visual hull. We allow the topology of the deforming surface to change in the local refinement step, using a method similar to that of [12]: As resolution increases and edges are split, it may happen that three vertices in a shrinking area of the surface are connected to each other

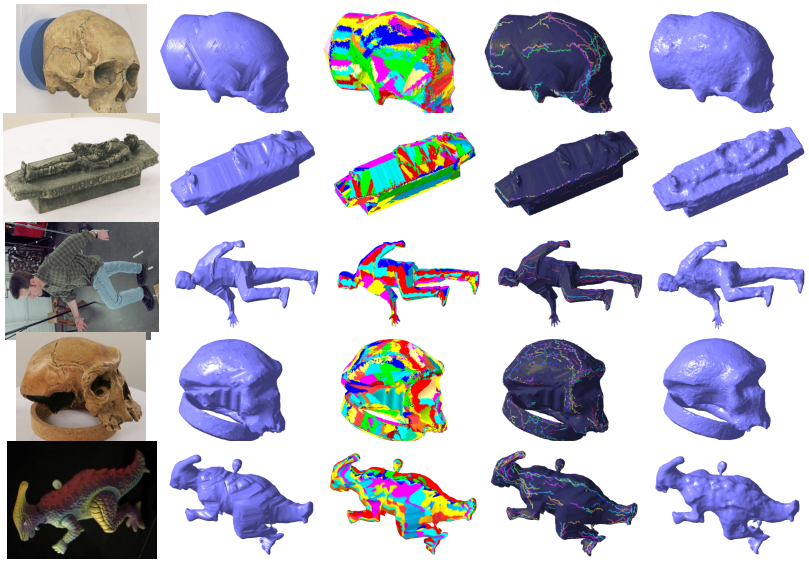

Fig. 6. From left to right, an input image, a visual hull, cone strips on the visual hull boundary, identified rim segments, and a surface after graph cuts for the remaining five objects.

without forming a face. In this case, we cut the surface at the three vertices into two open components, and add a copy of the triangle to both components.

5.2 Results

We have conducted experiments with strongly calibrated cameras and six objects: two toy dinosaurs, two human skulls (modern man and *Homo Heidelbergensis*), a toy mummy, and a person. Four of the six data sets, captured in our laboratory, consist of 24 images, with an image size of about $2000 \times 2000\text{pixel}^2$. The two exceptions are the person (courtesy of S. Sullivan), that only appears in 11 pictures, with an image size of roughly $2000 \times 1300\text{pixel}^2$, and the second dinosaur (courtesy of S. Seitz) that appears in 21 images, with an image size of about $640 \times 480\text{pixel}^2$. In all cases, contours have been extracted interactively.

Figure 1 illustrates the successive steps of our algorithm in a case of the first toy dinosaur. This object is about 20cm in diameter, with fine surface details including fin undulations, and scales in the neck. These details are well captured by the model, even though the corresponding height variations are a fraction of 1mm. Figure 6 shows input images and intermediate results for the remaining five objects. As can be seen in the figures, rim points have been successfully identified, especially at high-curvature parts of the surface. Our rim-discontinuity assumption (Section 2.3) breaks at the cloth of the standing human model, due to its complicated fold structure and the sparse input viewpoints, while the assumption rarely fails in the other data sets. Nonetheless, spurious rim points have been detected and filtered out by our conservative post-processing in all the data sets. With the help of the identified rim segments, the graph cuts step

recovers the main surface structures rather well, including large concavities, while preserving high-curvature structural details, such as the fingernails of the first dinosaur, the fingers of the person, the cheekbones of the two skulls, and the metal bar sticking out from the second dinosaur. Figure 7 shows shaded and texture-mapped renderings of the final models including several close-ups. Note that some of the surface details are not recovered accurately. In some cases, this is simply due to the fact that the surface is not visible from any cameras: the bottom part of the first dinosaur, for example. In other cases, this is due to failures of our algorithm: For example, the eye sockets of the skulls are simply too deep to be carved away by graph cuts or local refinement. The human is a particularly challenging example, because of the extremely complicated folds of the cloth, and its high-frequency stripe patterns. Nonetheless, our algorithm has performed rather well in general, correctly recovering minute details such as the sutures of the skulls, the large concavity in the mummy's chest, much of the shirt fold structure in the human example, as well as the high-curvature structural details mentioned earlier.

To evaluate the contributions of each step in our approach, we have performed the following two experiments: First, we have implemented and added the ballooning term introduced in [23] to the energy function in the graph cuts step, while removing the hard constraints enforced by the identified rim segments to see its effects on the over-carving problem mentioned earlier (Fig. 8, first row). Note that the layer-based graph representation is still used in this experiment, instead of the voxel representation used in [23]. The leftmost part of the figure shows the result of our graph cuts step (with fixed rim segments), and the remaining three columns illustrate the effects of the ballooning term with three different weights associated with it, the weight being zero at the left and increasing to the right. As shown by the figure, high-curvature surface details have not been preserved with the ballooning term. Even in the third column of the figure, where the ballooning term is too high to preserve surface details in other parts of the surface, the fingers almost disappear. This may be due in part to the fact that photometric consistency measurements become unreliable at high-curvature parts of a surface which, on the other hand, tend to generate highly reliable rim consistency constraints. We have also tested our algorithm without its graph cuts phase, yielding a purely local method comparable to those proposed in [7, 9]. Figure 8 (second row) shows two examples: the graph cuts step being included in the left part of the diagram, and omitted in the right part. As expected, local minimum problems are apparent in the latter case. Of course, it would be highly desirable to conduct more comparisons with native implementations of the algorithms proposed in [7, 9, 23], but we do not have access to these (yet). In the mean time, we have tried an implementation of *voxel coloring* [11, 18], kindly provided by S. Seitz, on two of our examples (Fig. 8, bottom). The results appear rather noisy compared to ours (see Fig. 7), probably due to the lack of regularization, and several concavities are missed in the two objects (e.g., the chest of the mummy).

As noted before, we have been unable (so far) to conduct a quantitative assessment of our algorithm due to the lack of ground truth data. A qualitative assessment can be obtained by α-blending surface textures backprojected from different images: They will only appear consistent when the geometry is correct. Figure 9 shows the results of such an experiment. Blended textures on the surface after visual hull construction, global surface carving, and final local refinement are shown, from left to right, for the first

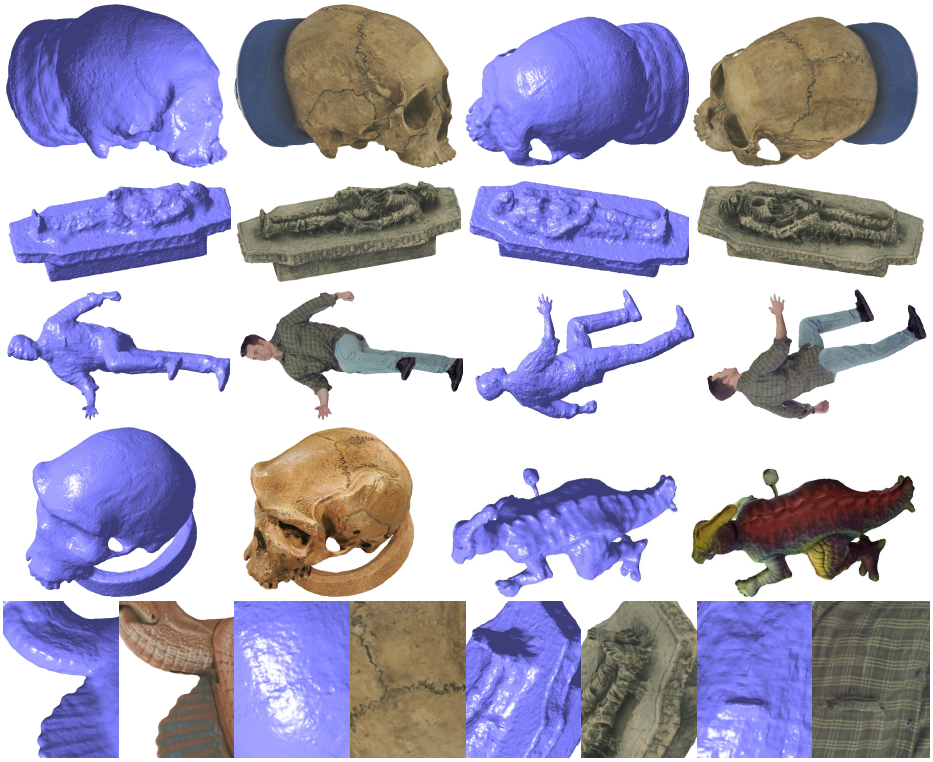

Fig. 7. Experimental results. See text for details, and the video submitted as supplemental material for animations.

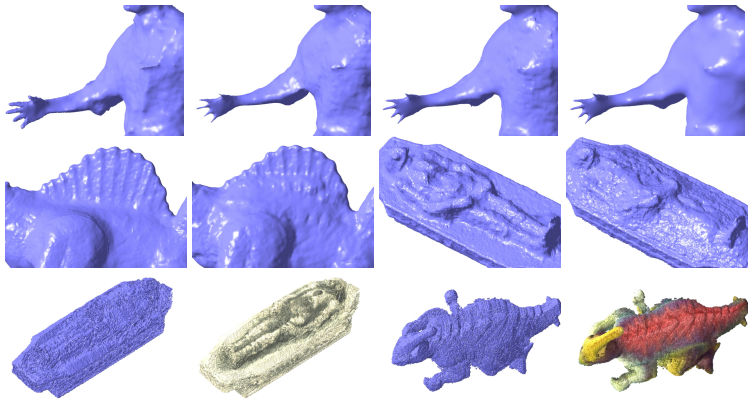

Fig. 8. A preliminary comparative evaluation of our algorithm. Top: comparison with our implementation of a variant of the method proposed by Vogiatzis *et al.* [23]. Middle: comparison with a purely local method initialized with the visual hull surface, akin to those proposed by Hernández Esteban and Schmitt [7], and Keriven [9]. Bottom: comparison with the voxel coloring method of Seitz [18]. See text for details.

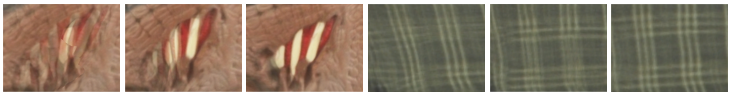

Fig. 9. Assessing the accuracy of the reconstruction: α-blended surface textures backprojected from different images are shown for (from left to right) the visual hull, the surface obtained after graph cuts, and the final surface after local refinement for details of the dinosaur and shirt surfaces. See text for details.

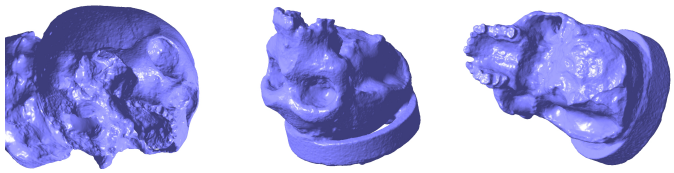

Fig. 10. Preliminary results combining carved visual hulls with wide-baseline stereo. Large concavities such as eye sockets are successfully recovered.

dinosaur and the human figure. It is clear that the backprojected textures are consistent on the final surfaces.

6 Conclusions and Future Work

We have proposed a method for acquiring high-quality geometric models of complex 3D shapes by enforcing the photometric and geometric consistencies associated with multiple calibrated images of the same solid, and demonstrated the promise of the approach with six real data sets and some preliminary qualitative evaluation experiments. Next on our agenda is a quantitative assessment of our algorithm using a measuring device such as a laser theodolite to recover accurate ground truth at a number of key points. One of the limitations of our current approach is that it cannot handle concavities too deep to be carved away by the graph cuts or local refinement steps. To overcome this problem. we plan to combine our approach with recent work on sparse wide-baseline stereo from interest points (e.g., [17]) in order to incorporate stronger geometric constraints in the carving and local refinement stages, and Fig. 10 shows the results of a preliminary experiment. Attempting, as in [21], to explicitly handle non-Lambertian surfaces is of course of interest. Finally, we plan to follow the lead of photogrammetrists and add a final *simultaneous camera calibration* stage, where both the camera parameters and the surface shape are refined simultaneously using bundle adjustment [22].

Acknowledgments. This research was partially supported by the National Science Foundation under grant IIS-0312438, and the Beckman Institute. We thank Svetlana Lazebnik for providing the original visual hull software used in our implementation, and Steve Seitz for the toy dinosaur data set together with his voxel coloring software. We thank Jodi Blumenfeld and Steven R. Leigh for providing us with the two human skull data sets. Lastly, we want to thank Sariel Har-Peled and Theodore Papadopoulo for discussions on the global optimization method.

References

1. B.G. Baumgart. *Geometric modeling for computer vision*. Stanford University, 1974.
2. Y. Boykov and V. Kolmogorov. Computing geodesics and minimal surfaces via graph cuts. In *ICCV*, pages 26–33, 2003.
3. Kong Man Cheung, Simon Baker, and Takeo Kanade. Visual hull alignment and refinement across time: A 3d reconstruction algorithm combining shape-from-silhouette with stereo. In *CVPR*, June 2003.
4. B. Curless and M. Levoy. A volumetric method for building complex models from range images. In *SIGGRAPH*, pages 303–312. ACM Press, 1996.
5. H. Delingette, M. Hebert, and K. Ikeuchi. Shape representation and image segmentation using deformable surfaces. *IVC*, 10(3):132–144, 1992.
6. O. Faugeras and R. Keriven. Variational principles, surface evolution, PDE's, level set methods and the stereo problem. *IEEE Trans. Im. Proc.*, 7(3):336–344, 1998.
7. C. Hernández Esteban and F. Schmitt. Silhouette and stereo fusion for 3D object modeling. *CVIU*, 96(3):367–392, 2004.
8. D.F. Huber and M. Hebert. 3D modeling using a statistical sensormodel and stochastic search. In *CVPR*, volume 1, pages 858–865, 2003.
9. R. Keriven. A variational framework to shape from contours. Technical Report 2002-221, ENPC, 2002.
10. V. Kolmogorov and R. Zabih. Multi-camera scene reconstruction via graph cuts. In *ECCV*, volume 3, pages 82–96, 2002.
11. K.N. Kutulakos and S.M. Seitz. A theory of shape by space carving. *IJCV*, 38(3), 2000.
12. Jacques-Olivier Lachaud and Annick Montanvert. Deformable meshes with automated topology changes for coarse-to-fine 3d surface extraction. *Medical Image Analysis*, 3(2):187–207, 1999.
13. S. Lazebnik. Projective visual hulls. Technical Report MS Thesis, UIUC, 2002.
14. M. Levoy, K. Pulli, B. Curless, S. Rusinkiewicz, D. Koller, L. Pereira, M. Ginzton, S. Anderson, J. Davis, J. Ginsberg, J. Shade, and D. Fulk. The digital michelangelo project: 3D scanning of large statues. In *SIGGRAPH*, pages 131–144, 2000.
15. W. Matusik, H. Pfister, A. Ngan, P. Beardsley, R. Ziegler, and L. McMillan. Image-based 3D photography using opacity hulls. In *SIGGRAPH*, 2002.
16. S. Paris, F. Sillion, and L. Quan. A surface reconstruction method using global graph cut optimization. In *ACCV*, January 2004.
17. F. Schaffalitzky and A. Zisserman. Viewpoint invariant texture matching and wide baseline stereo. In *ICCV*, 2001.
18. S.M. Seitz and C.R. Dyer. Photorealistic scene reconstruction by voxel coloring. In *CVPR*, pages 1067–1073, 1997.
19. S. Sinha and M. Pollefeys. Visual hull reconstruction from uncalibrated and unsynchronized video streams. In *Int. Symp. on 3D Data Processing, Visualization & Transmission*, 2004.
20. S. Sinha and M. Pollefeys. Multi-view reconstruction using photo-consistency and exact silhouette constraints: A maximum-flow formulation. In *ICCV*, 2005.
21. S. Soatto, A.J. Yezzi, and H. Jin. Tales of shape and radiance in multiview stereo. In *ICCV*, pages 974–981, 2003.
22. V. Uffenkamp. State of the art of high precision industrial photogrammetry. In *Third International Workshop on Accelerator Alignment*, Annecy, France, 1993.
23. George Vogiatzis, Philip H.S. Torr, and R. Cipolla. Multi-view stereo via volumetric graph-cuts. In *CVPR*, pages 391–398, 2005.
24. C. Xu and J.L. Prince. Gradient vector flow: A new external force for snakes. In *CVPR*, pages 66–71, 1997.