Int J Comput Vis (2009) 81: 53–67
DOI 10.1007/s11263-008-0134-8

Carved Visual Hulls for Image-Based Modeling

Yasutaka Furukawa · Jean Ponce

Received: 27 October 2006 / Accepted: 28 February 2008 / Published online: 29 March 2008

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 seven real data sets are presented, along with qualitative and quantitative comparisons with several state-of-the-art image-based-modeling algorithms.

Keywords Visual hull · Graph cuts · Multi-view · Stereo · Rim · Silhouettes

Y. Furukawa (✉) · J. Ponce
Department of Computer Science and Beckman Institute, University Of Illinois, Urbana, IL 61801, USA
e-mail: yfurukaw@uiuc.edu

J. Ponce
e-mail: jean.ponce@ens.fr

J. Ponce
Département d'Informatique, Ecole Normale Supérieure, Paris, France

1 Introduction

1.1 Background

This article addresses the problem of acquiring high-quality solid models[1] of complex three-dimensional (3D) shapes from multiple calibrated photographs, a process dubbed *image-based modeling* (Seitz and Dyer 1997; Roy and Cox 1998; Kutulakos and Seitz 2000; Kolmogorov and Zabih 2002; Matusik et al. 2002; Hernández Esteban and Schmitt 2004; Pons et al. 2005; Vogiatzis et al. 2005; Sinha and Pollefeys 2005; Tran and Davis 2006; Furukawa and Ponce 2006, 2007; Goesele et al. 2006; Hornung and Kobbelt 2006; Strecha et al. 2006; Habbecke and Kobbelt 2007). The quality of reconstructions has been rapidly improving in the last decade due to the developments of digital photography and the sophistication of proposed algorithms. According to a recent study (Seitz et al. 2006), state-of-the-art image based modeling algorithms achieve an accuracy of about 1/200 (1 mm for a 20 cm wide object) from a set of low resolution (640×480 pixel^2) images. The potential applications range from the construction of realistic object models for the film, television, and video game industries, to the quantitative recovery of metric information (metrology) for scientific and engineering data analysis.

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

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

carving away successive layers of voxels with high discrepancy (Kutulakos and Seitz 2000; Seitz and Dyer 1997). 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 (Hernández Esteban and Schmitt 2004; Soatto et al. 2003). *Level-set* techniques, on the other hand, implicitly represent surfaces as the zero set of a time-varying volumetric density (Faugeras and Keriven 1998; Keriven 2002). The *graph cuts* global optimization technique can also be used to avoid local extrema during the search for the optimal surface (Roy and Cox 1998; Paris et al. 2004; Vogiatzis et al. 2005; Sinha and Pollefeys 2005; Furukawa and Ponce 2006; Tran and Davis 2006; Hornung and Kobbelt 2006). The last broad class of image modeling techniques is the oldest one: The *visual hull*, introduced by Baumgart (1974) in the mid-seventies, 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 (Matusik et al. 2002; Sinha and Pollefeys 2004; Lazebnik et al. 2007).

1.2 Approach

Hernández Esteban and Schmitt (2004) propose to use the visual hull to initialize the deformation of a surface mesh under the influence of photoconsistency constraints expressed by gradient flow forces (Xu and Prince 1997): See Keriven (2002) 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 (Vogiatzis et al. 2005). 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 Sect. 6). Tran and Davis (2006) propose to first compute a visual hull while identifying "Constraint Points" that are likely to be on the surface of an object, then use the identified points as constraints in applying the graph cuts to prevent the over-carving. Sinha and Pollefeys (2005) proposed an algorithm to reconstruct a surface that exactly satisfies all the silhouette constraints while maximizing the photometric consistencies. However, their method needs to construct a complicated graph structure and has been tested on only relatively simple objects.

The fundamental approach of our method is similar to that of Tran and Davis's work (2006), however is different in that our method makes use of combinatorial structures of a visual hull to obtain constraints that are used with the graph cuts. Our method also has a final refinement step, which is nonetheless essential in achieving high-quality reconstructions, after the graph cuts. In particular, 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 et al. (2007) 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 Lazebnik et al. (2007) 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 Vogiatzis et al. (2005) 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 Sect. 6, the third step, similar in spirit to the local optimization techniques proposed in Hernández Esteban and Schmitt (2004), Keriven (2002), remains nonetheless essential in achieving high-quality results. The overall process is illustrated in Fig. 1, and the rest of this paper details each step and presents our implementation and the results with seven real data sets along with some qualitative and quantitative comparative experiments. Results in Figs. 1 and 12 demonstrate the recovery of very fine surface details in all our experiments. A preliminary version of this article appeared in Furukawa and Ponce (2006).

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.[2] Φ_i grazes the object along a surface curve, the *rim* Γ_i. The *visual hull* is the solid formed by

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

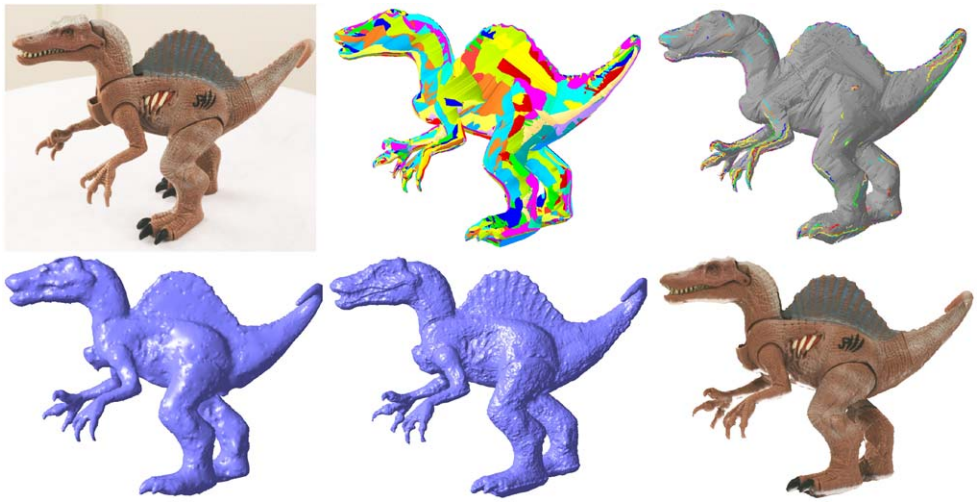

Fig. 1 Overall flow of the proposed approach. *Top*: one of the 24 input pictures of a toy dinosaur (*left*), the corresponding visual hull (*center*), and the rims identified in each strip using dynamic programming (*right*). *Bottom*: the carved visual hull after graph cuts (*left*) and iterative refinement (*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 1 mm for this object about 20 cm wide

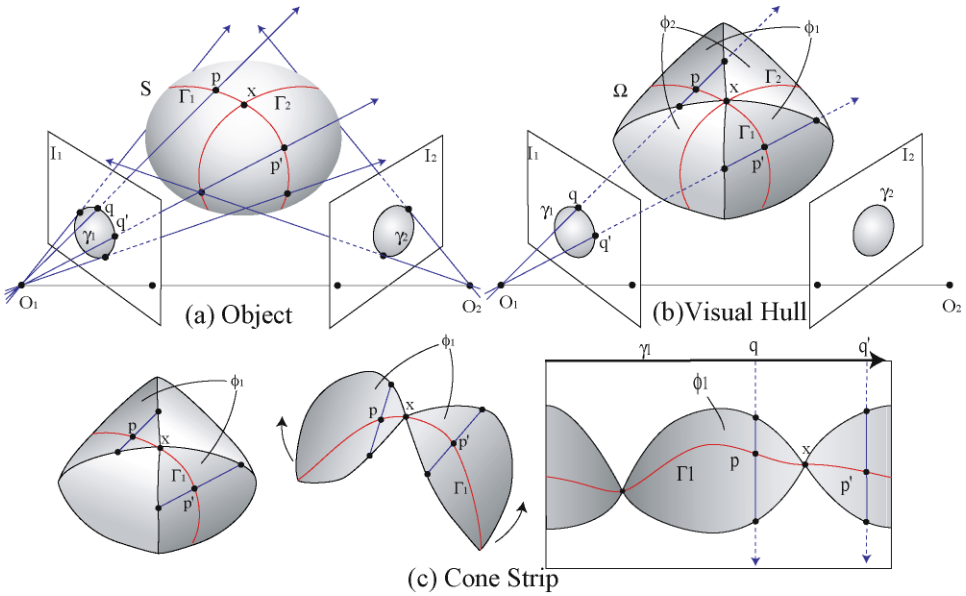

Fig. 2 A visual hull, cone strips and rims: (**a**) an egg-shaped object is viewed by 2 cameras with optical centers O_1 and O_2; the point x is a frontier point; (**b**) its visual hull is constructed from two apparent contours γ_1 and γ_2, the surface Ω of the visual hull consisting of two cone strips ϕ_1 and ϕ_2; (**c**) the cone strip ϕ_1 associated with the first image I_1 is stretched out (*middle figure*) 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

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 Cheung et al. 2003; Tran and Davis 2006 by the same argument, but the algorithms and their purposes are different from ours). In particular, we use the exact visual hull algorithm proposed in Lazebnik et al. (2007) to obtain a triangulated mesh model representing a visual hull. The algorithm also outputs a set of triangles on the mesh that belongs to each cone strip. 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 Faugeras and Keriven (1998).

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 so that its image projection becomes approximately $\mu \times \mu$ pixel^2, and τ $\mu \times \mu$ tangent plane "windows" $h_1, \ldots, h_\tau$ are retrieved from the corresponding input images. The image discrepancy score is finally computed as

$$f(p) = \frac{2}{\tau(\tau-1)} \sum_{i=1}^{\tau} \sum_{j=i+1}^{\tau} 1 - \exp\left(-\frac{(1-\mathrm{NCC}(h_i,h_j))^2}{2\sigma_1^2}\right),$$

where $\mathrm{NCC}(h_i, h_j)$ is the normalized cross correlation between h_i and h_j. $\tau = 5$, $\mu = 11$, and $\sigma_1 = 0.8$ are used throughout our experiments. We also denote by $f^*(p)$ an average NCC score:

$$f^*(p) = \frac{2}{\tau(\tau-1)} \sum_{i=1}^{\tau} \sum_{j=i+1}^{\tau} \mathrm{NCC}(h_i,h_j).$$

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: First, a cone strip, which should consist of multiple components connected through frontier points, can result into multiple horizontally separated components (*horizontal neighbors*) due to measurement errors (Fig. 3(a)); Second, even in the absence of any measurement errors nor noises, a cone strip can have multiple components intersecting the same vertical line with the rim being in any one of these (*vertical neighbors*); Third, 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 right or left end points of each connected strip component, in other words, 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

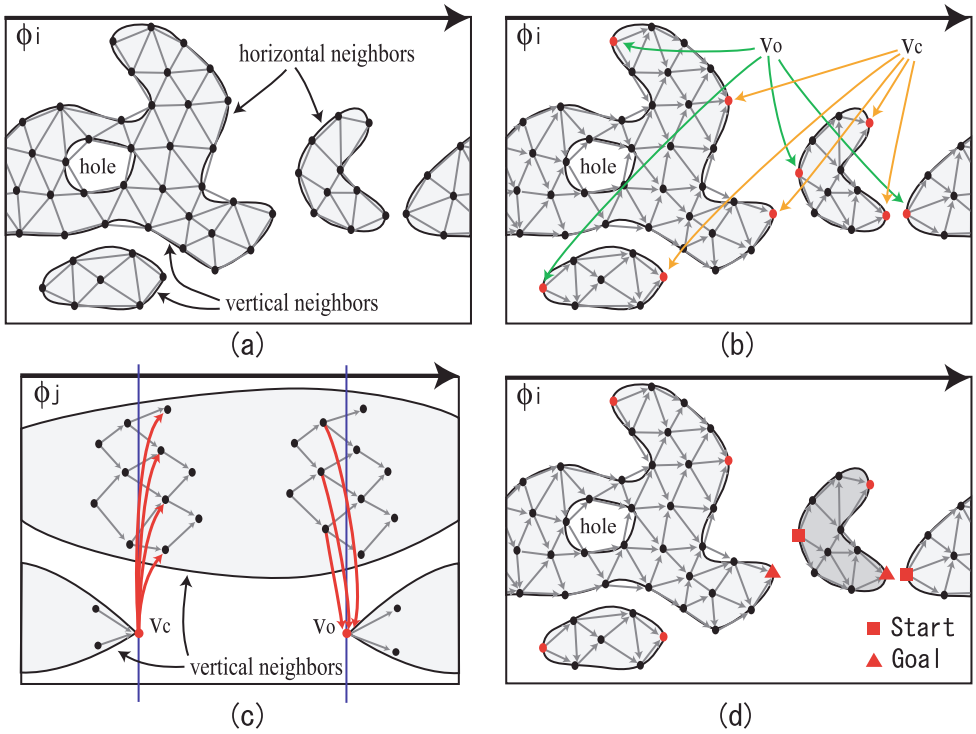

Fig. 3 (**a**) An undirected graph representing a cone strip ϕ_i. The two leftmost components are vertical neighbors. (**b**) The opening and closing vertices v_o and v_c of ϕ_i. (**c**) Illustration of the vertical edge creation process for a different strip ϕ_j. (**d**) After the horizontal and vertical 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. Note that vertical edges are omitted for the purpose of illustration

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 average NCC score $f^*(v)$ defined earlier and the vertical strip size $g(v)$ at a vertex v. More concretely, a vertex v is detected as a false-positive if either $4R \cdot \iota < g(v)$ or $R \cdot \iota < g(v)$ and $f^*(v) < \eta$ hold, where R is an average distance from all the vertices V' in the mesh to their center of mass $\sum_{v \in V'} v/|V'|$. ι and η are thresholds for the vertical strip size and the image discrepancy score, respectively, and are selected for each data set in our experiments (see Table 1 for an actual choice of parameters). If there exists multiple components along a viewing ray, the vertical strip size $g(v)$ is simply computed as a distance between the closest and the farthest points on the cone strip. Intuitively, a rim point is filtered out when its corresponding vertical strip size is too large (the first condition) and when the vertical strip size is not small enough and the average NCC score is worse than η (the second condition). Note that when the vertical strip size is very small, 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 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* (remember that the visual hull is an *outer* object approximation) to generate multiple layers forming a 3D graph J_i, associate photoconsistency weights to the edges of this graph, and use graph cuts to carve the surface.[3] The overall process is summarized in Fig. 4 and detailed in the next two sections.

3.1 Deforming the Surface to Set Vertices

In this section, after initializing the vertices of J_i by those in $\overline{G_i}$, which will be the first layer of the graph J_i, the surface is iteratively deformed inwards to generate more vertices, which, in turn, will form additional layers of J_i. More precisely, at each iteration of the deformation process, we move every vertex v in $\overline{G_i}$ (except for the boundaries) along its surface normal $\mathbf{N}(v)$ and apply smoothing:

$$v \leftarrow v - \frac{\varepsilon}{\lambda}(\zeta_1 f(v) + \zeta_2)\mathbf{N}(v) + \mathbf{s}(v), \tag{1}$$

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 Delingette et al. (1992). For each vertex v, we keep track of how much it has moved along its surface normal direction, and every time the accumulated distance exceeds ε, its coordinate is added to J_i as a vertex, and the distance is reset to 0. Note that using $f(v)$ in (1) 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. During deformations, surface normals and smoothness terms are re-estimated by using the current surface at every iteration, while photoconsistency functions $f(v)$ are evaluated at all the vertices (except for the boundaries) once in every λ iterations for efficiency. We use $\zeta_1 = 100$, $\zeta_2 = 0.1$, $\beta_1 = 0.4$, $\beta_2 = 0.3$, $\rho = 40$, and $\lambda = 20$ in all our experiments, which have empirically given good results for our test objects. ε determines an offset between vertically adjacent vertices, and is set to be 0.5 times the average edge length in $\overline{G_i}$. Note that the choice of parameters ε and ρ is essentially difficult,

[3] The graph associated with a voxel grid serves as input in typical applications of graph cuts to image-based modeling (e.g., Roy and Cox 1998; Boykov and Kolmogorov 2003; Kolmogorov and Zabih 2002; Paris et al. 2004; Vogiatzis et al. 2005; Tran and Davis 2006; Hornung and Kobbelt 2006). The surface deformation scheme is proposed here instead to take advantage of the fact that the visual hull is already a good approximation.

because a surface of an object must lie between the top and the bottom layers of J in order for the graph cuts step to work, but we do not know in advance which parameter set can guarantee such a condition.

3.2 Building a Graph and Applying Graph Cuts

After setting the vertices of J_i, two types of edges are added as shown in Fig. 5. Let us denote an array of vertices generated from v_k as $\{v_k^0, v_k^1, \ldots\}$ in an order of creation. Firstly, a horizontal edge $(v_k^j, v_{k'}^j)$ is added to J_i if v_k and $v_{k'}$ are neighbors in $\overline{G}_i$. Note that vertices connected by horizontal edges form an offset layer of $\overline{G}_i$, and the top-most layer is identical with $\overline{G}_i$. Secondly, a vertical edge (v_k^i, v_k^{i+1}) is added to connect the offset instances of the same vertex in adjacent layers. A simple variant of the technique proposed in Boykov and Kolmogorov (2003) is used to compute edge weights by using photoconsistency values that have already been computed in J_i during the deformation procedure. 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, which is approximated by ε times the squared of the average distance from v_i to the adjacent vertices in the same layer. Intuitively, weights should be large where vertices are sparse. We use $\alpha = 1.0$ and 6.0 for horizontal and vertical edges, respectively, in all our experiments, which accounts for the fact that edges are

Input: A connected component $\overline{G}_i$ on a visual hull boundary
Output: A carved visual hull model inside $\overline{G}_i$

$J \leftarrow \overline{G}_i$; //$J$ will contain a 3D graph.
ForEach vertex $v \in \overline{G}_i$ except for the boundary
  $d(v) \leftarrow 0$; //$d(v)$ keeps track of how much v has moved along its surface normal.
EndFor
For $j = 1$ to ρ
  Recompute $f(v)$ for each vertex;
  For $k = 1$ to λ
    ForEach vertex $v \in \overline{G}_i$ except for the boundary
      Recompute $\mathbf{N}(v)$ and $s(v)$;
      $v_{move} \leftarrow -\frac{\varepsilon}{\lambda}(\zeta_1 f(v) + \zeta_2)\mathbf{N}(v) + \mathbf{s}(v)$;
      $v \leftarrow v + v_{move}$;
      $d(v) \leftarrow d(v) - v_{move} \cdot \mathbf{N}(v)$;
      If $\varepsilon < d(v)$
        $d(v) \leftarrow 0$; //Every time $d(v)$ exceeds ϵ, a new vertex is added to V.
        Add the current v to J;
      EndIf
    EndFor
  EndFor
EndFor
Add vertical and horizontal edges to J, and compute their weights;
Use graph cuts to find a minimum cut in J.

Fig. 4 Algorithm description of the graph cuts step. This procedure is applied to every connected component on a visual hull boundary, which is surrounded by identified rim segments

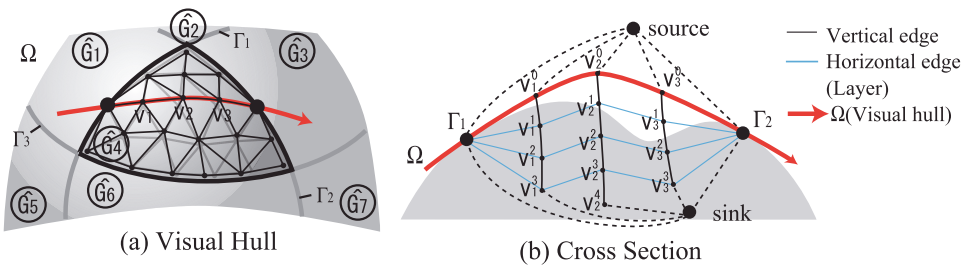

Fig. 5 Deforming the surface for graph cuts: (**a**) the surface Ω of the visual hull is decomposed into multiple independent components $\overline{G}_i$; (**b**) the layers generated by the deformation process is illustrated for the cross section of $\overline{G}_4$ that contains vertices v_1, v_2, and v_3

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. Note that generated layers in J_i may not necessarily have the same topology due to the adaptive deformation scheme and the smoothness term in (1): Different vertices may be registered to J_i for different times, and bottom layers in J_i may miss some of the vertices as shown in Fig. 5.

3.3 Practical Matters

While graph cuts is a global optimization tool, we apply the surface deformation and graph cuts procedure multiple times in practice for the following reasons. First, as we have already mentioned, an object must lie between the top and the bottom layers of J_i in order for the graph cuts step to work. However, we do not know, in advance, a choice of parameters ε and ρ that satisfies this condition. Furthermore, in some cases, any choice of parameters may not satisfy the condition due to the complicated shape of a visual hull (see Fig. 11 for an actual example). Second, for the global optimum provided by graph cuts to be meaningful, the edge weights must accurately measure the photo-consistency, 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, and multiple graph cuts application helps in estimating better surface normals and photo-consistency functions. 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. Also note that after each graph-cuts application, we remesh the surface to keep regular triangulations (see Sect. 5 for more details). Empirically, four 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 Hernández Esteban and Schmitt (2004). 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. 6). 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

$$\mathbf{r}(v_k) = \overline{v_j^* r_j} \frac{\exp(-(\overline{v_k r_j} - \overline{v_j^* r_j})^2 / 2\sigma_2^2)}{\sum_{v_{k'} \in V_j} \exp(-(\overline{v_{k'} r_j} - \overline{v_j^* r_j})^2 / 2\sigma_2^2)} \mathbf{N}(v_k),$$

where $\mathbf{N}(v_k)$ is the unit surface normal in v_k.

The basic structure of the algorithm is simple (see Fig. 7). 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. See Sect. 5 for more details of the remeshing operations.

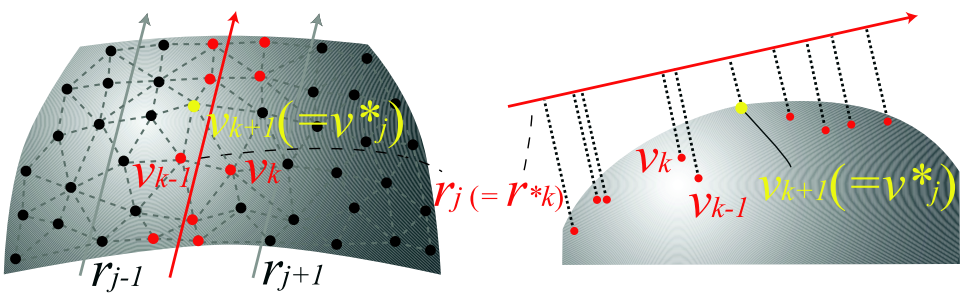

Fig. 6 The rim consistency force is computed for a viewing ray r_j, then distributed to all the vertices V_j whose closest ray is r_j. Here v_{k+1} is the closest vertex v_j^* to r_j

5 Implementation Details

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: Nested apparent contours can be simply treated as independent contours from different images in our rim-identification, graph-cuts, and local refinement steps. 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* data set shown in Fig. 12, while the apparent contour components associated with holes are explicitly used for the *Twin* model. 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.

Input: A carved visual hull model after the graph cuts step
Output: A refined final surface

REPEAT
Remesh and increase the resolution of the mesh;
REPEAT
  Compute three forces at all the vertices;
  For each vertex $\mathbf{v}$ on the mesh
    $\mathbf{v} \leftarrow \mathbf{v} + \mathbf{s}(v) + \mathbf{r}(v) - \kappa \nabla f(v) \cdot \mathbf{N}(v)$;
  EndFor
until convergence;
until mesh reaches the desired resolution.

Fig. 7 Algorithm description of the local refinement step

As we have described, remeshing operations are applied to a mesh in two places of our algorithm: (1) after each graph cuts application; and (2) during the local refinement step. The remeshing procedure consists of a single parameter ξ and a sequence of edge splits, collapses, and swaps (Hoppe et al. 1993). More precisely, we contract an edge if its length is less than $\xi/2$, split it if its length is more than 2ξ, and swap edges if the degrees of vertices become closer to six with the operation. Note that we control the resolution of a mesh by changing the value of ξ during the local refinement step: we keep on decreasing the value of ξ, until the image projections of edges become approximately 2 pixels in length.

Finally, the topology of an object's surface is not necessarily the same as that of its visual hull. Therefore, we allow the topology of the deforming surface to change in the remeshing procedure, using a method similar to that of Lachaud and Montanvert (1999). While remeshing, it may happen that three vertices in a shrinking area of the surface are connected to each other 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 to fill the holes.

6 Experimental Results

We have conducted experiments with strongly calibrated cameras and seven objects: a toy dinosaur, a human skull

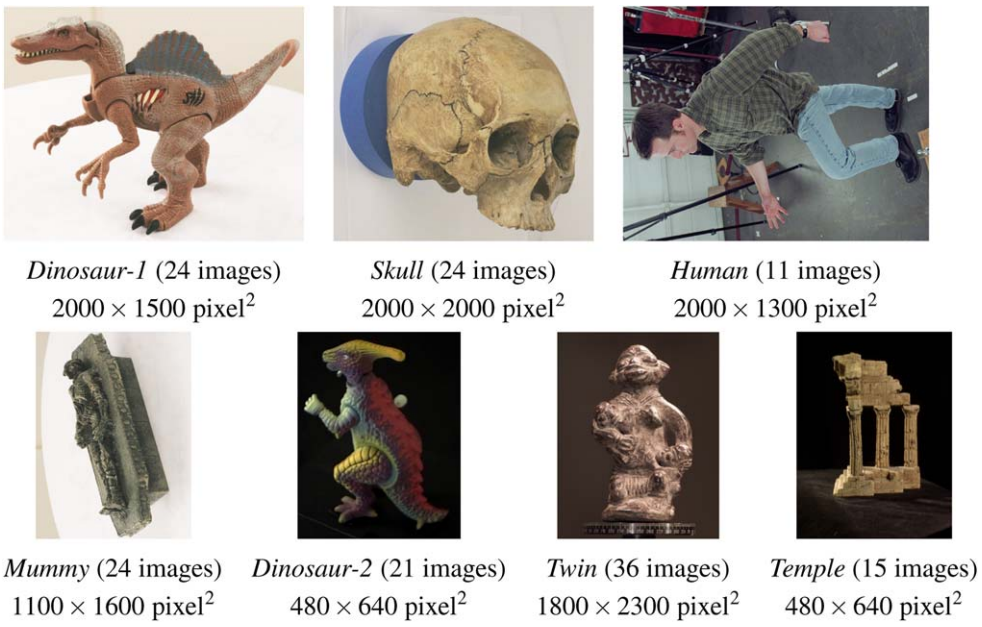

Fig. 8 Sample input images for the data sets used in our experiments. The number of images in a data set and their approximate sizes are also shown

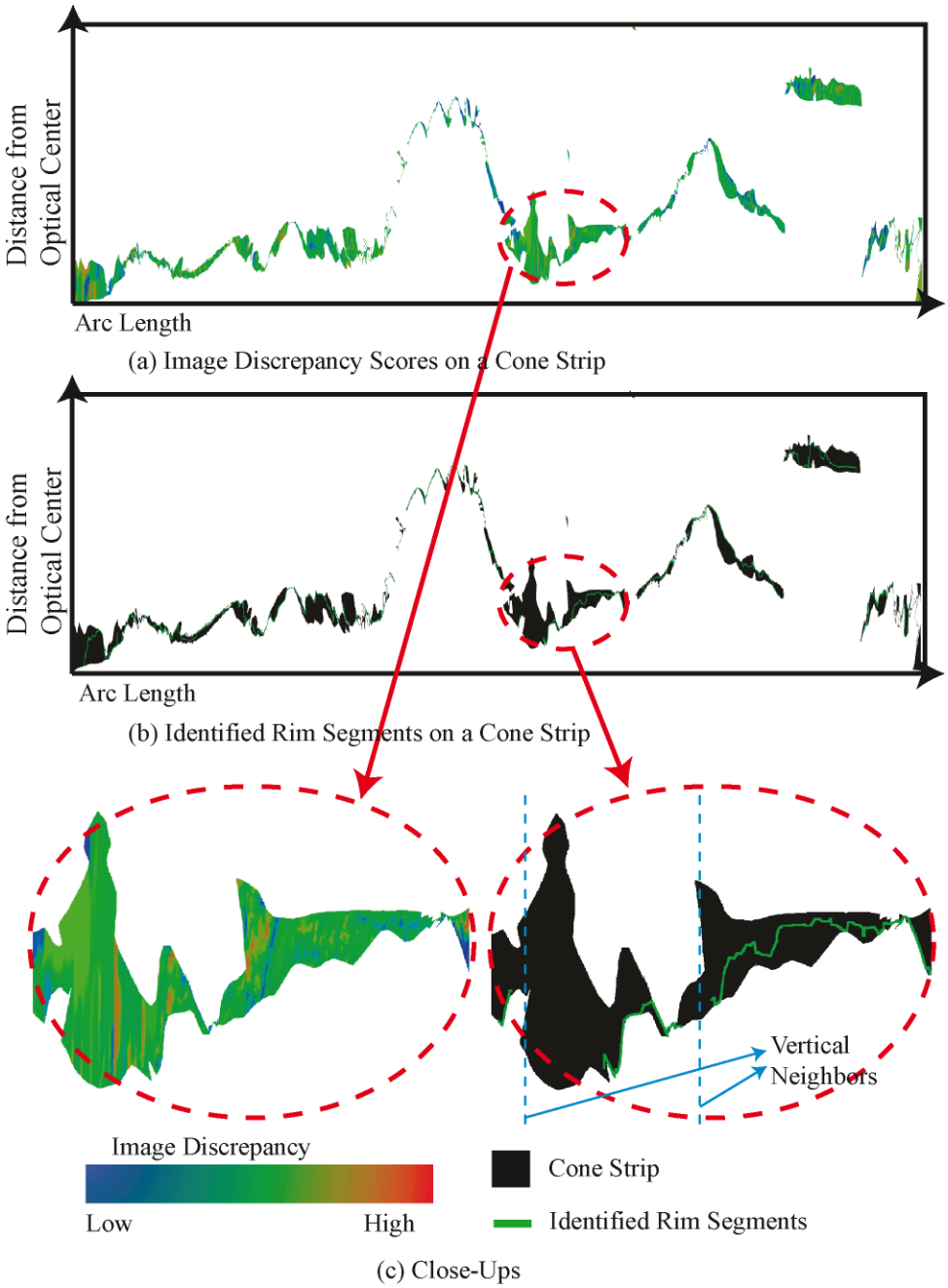

Fig. 9 (Color online) A cone strip, the evaluated image discrepancy scores, and its corresponding identified rim segments are shown for one of the input image contours of the *Human* data set. The cone strip is mapped onto a plane by the parameterization described in Sect. 2. *Black regions* and *green curves* represent the cone strip and the identified rim segments, respectively. *Blue lines* in (**c**) illustrate cases where there exists multiple strip components (*vertical neighbors*) along a single viewing ray

(courtesy of J. Blumenfeld and S. Leigh), a standing human (courtesy of S. Sullivan), a toy mummy, another toy dinosaur (courtesy of S. Seitz), a statue (courtesy of C. Hernández Esteban and F. Schmitt), and a plaster reproduction of "Temple of the Dioskouroi" (courtesy of S. Seitz, B. Curless, J. Diebel, D. Scharstein, and R. Szeliski). Contours have been extracted interactively. A sample input image, the number of input images, and their approximate sizes are shown for each data set in Fig. 8.

6.1 Intermediate Results

Figure 9 illustrates a cone strip, the evaluated image discrepancy scores, and corresponding identified rim segments for an image contour of the *Human* data set. A cone strip is mapped onto a plane by parameterizing the horizontal axis by an arc length of the corresponding image contour, and the vertical axis by a distance from the optical center of the corresponding camera as in Fig. 3. Although *Human* is a relatively simple data set with only 11 input images, as

Table 1 Details on the rim-identification step. The second and the third columns of the table lists thresholds (ι, η) used in the rim identification step for each data set. The *fourth* and the *fifth columns* show rim filtering ratios: a ratio of rim segments that have been filtered out as outliers. The *right three columns* of the table list sizes of the three largest connected components on the visual hull boundary that are surrounded by identified rim segments. See text for details

Data set	Thresholds		Filtering ratio		Sizes of components		
	ι	η	Ave (%)	Min (%)	N_1 (%)	N_2 (%)	N_3 (%)
Dinosaur-1	0.015	0.9	83.2	56.7	96.2	1.02	0.63
Skull	0.06	0.7	69.2	44.2	99.9	0.062	0.051
Mummy	0.02	0.8	66.7	44.8	99.8	0.085	0.041
Human	0.05	0.8	27.8	16.7	93.2	3.4	2.1
Dinosaur-2	0.03	0.7	60.7	52.8	99.9	0.029	0.027
Twin	0.03	0.8	75.0	25.0	55.4	18.4	7.9
Temple	0.06	0.7	32.9	16.4	97.1	1.70	0.79

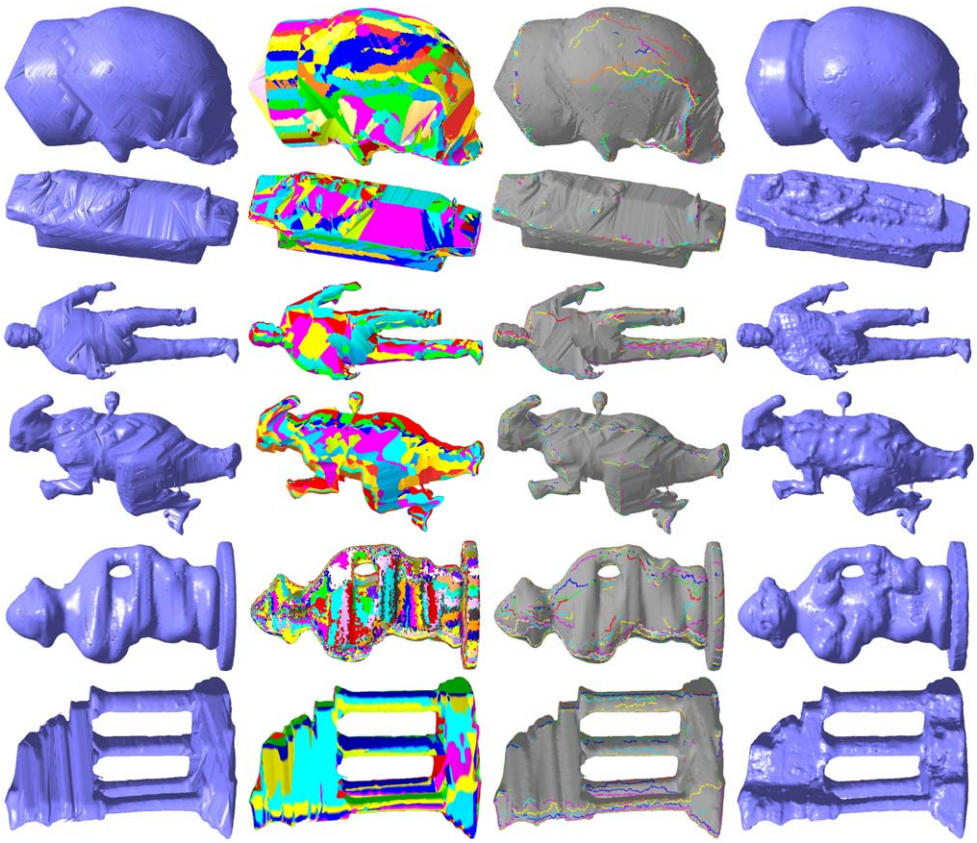

Fig. 10 From *left* to *right*, a visual hull, cone strips on the visual hull boundary, identified rim segments, and a surface after graph cuts for the six objects

the figure shows, the cone strip is pretty complicated and has *vertical neighbors* at many vertical lines (e.g., blue lines in Fig. 9(c)). Nonetheless, rim segments have been successfully identified, especially where cone strips are narrow.

Figures 1 and 10 illustrate the successive steps of our algorithm for all the data sets used in our experiments. 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 (Sect. 2.3) breaks at complicated surface structures, such as the cloth of the standing human model. In fact, in a few cases, false rim segments have not been completely removed and have caused a problem in the graph cuts step, for instance, near the nose and the eyes of the *Twin* model (see Sect. 6.3 for more discussions). Nonetheless, spurious segments have been detected and filtered out rather well by our aggressive post-processing in all the models. With the help of the identified rim segments, the graph cuts step recovers the main surface structures pretty well, including large concavities, while preserving high-curvature structural details, such as

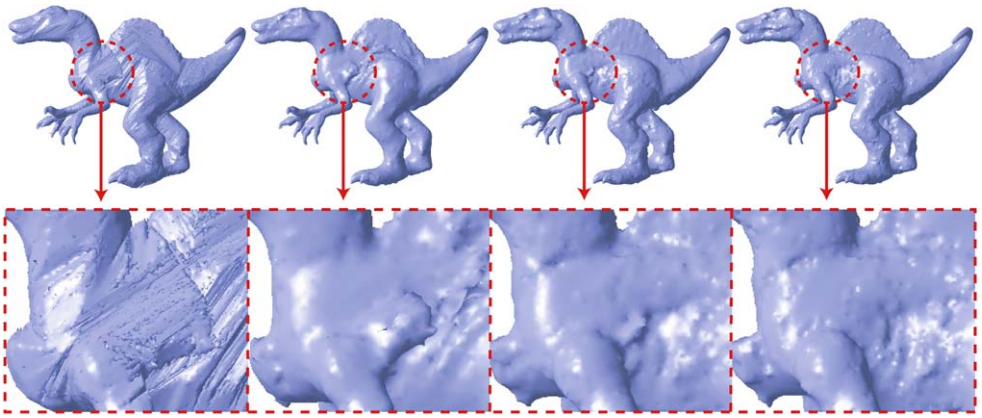

Fig. 11 (Color online) From *left* to *right*, starting from a visual hull, the graph cuts is applied to the model multiple times. *Red circles* illustrate a typical situation where the multiple graph cuts applications are necessary to reconstruct correct structures

the fingernails of the first dinosaur, the fingers of the person, the cheekbones of the skull, and the metal bar sticking out from the second dinosaur. Table 1 lists a pair of parameters (ι, η) used in the rim-identification step, a filtering ratio (how many percentages of the identified rim points have been filtered out as outliers), and sizes of the largest connected components surrounded by identified rim-segments, for each data set. Note that since a filtering ratio is computed for each image contour, the average and the minimum value of all the image contours are reported for each data set. The size of a connected component is calculated as a ratio of the number of vertices inside the component except for the boundary, against the number of vertices of the whole visual hull model except for the identified rim points. As the table shows, the filtering ratio varies depending on a data set and an image contour, but in general is around 60–80% due to our aggressive filtering. The table also illustrates a fact that a visual hull boundary is mostly covered by a single large connected component except for the *Twin* data set, which has many input images, and hence, many rim curves.

Figure 11 shows successive surface evolutions during the multiple graph cuts applications. Red circles in the figure illustrate a typical situation where the multiple applications are necessary to reconstruct correct structures: a part of the visual hull model of *Dinosaur-1* cannot be completely carved away by a single graph cuts application due to its complicated shape.

6.2 Final Results

Figures 1, 12 and 13 show shaded and texture-mapped renderings of the final 3D 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 (see Sect. 6.3 for another example with failures). 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 fin undulations and scales in the neck of the first toy dinosaur, which corresponding height variations are a fraction of 1 mm, 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. The proposed approach is implemented in C++, and Table 2 lists running times of four different steps of our algorithm for each data set. As the table shows, the bottleneck of the computation is the global optimization and the local refinement steps, each of which takes about two hours for most of the data sets and approximately four hours for the largest *Twin* model with a 3.4 GHz Pentium 4.

6.3 Comparisons

To evaluate the contributions of each step in our approach, we have performed the following experiments. First, we have implemented and added the ballooning term introduced in Vogiatzis et al. (2005) 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. 14, first row). Note that the layer-based graph representation is still used in this experiment, instead of the voxel representation used in Vogiatzis et al. (2005). 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 rightmost 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. It is because the

Fig. 12 Shaded and texture-mapped renderings of the final 3D models

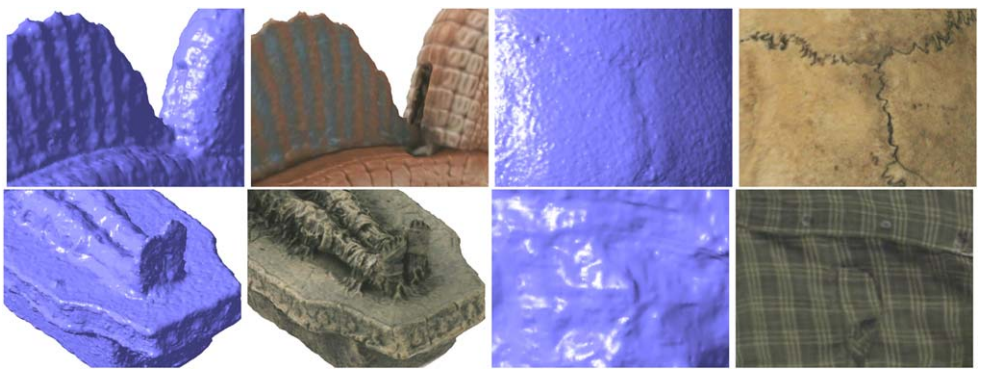

Fig. 13 Close-ups of reconstructions

Table 2 Running times of our algorithm, and the numbers of vertices and faces of final 3D models

	Dinosaur-1	*Skull*	*Human*	*Mummy*	*Dinosaur-2*	*Twin*	*Temple*
Visual hull	8.5 (min)	5.5 (min)	1 (min)	3 (min)	1.5 (min)	49 (min)	1 (min)
Rim-identification	3 (min)	4 (min)	5 (min)	9 (min)	3 (min)	7 (min)	2 (min)
Graph cuts	81 (min)	159 (min)	82 (min)	85 (min)	87 (min)	264 (min)	61 (min)
Local refinement	133 (min)	154 (min)	113 (min)	101 (min)	49 (min)	225 (min)	75 (min)
Number of vertices	272912	374057	481629	399688	440753	606669	328139
Number of faces	545820	748118	963254	799372	881502	1213338	656286

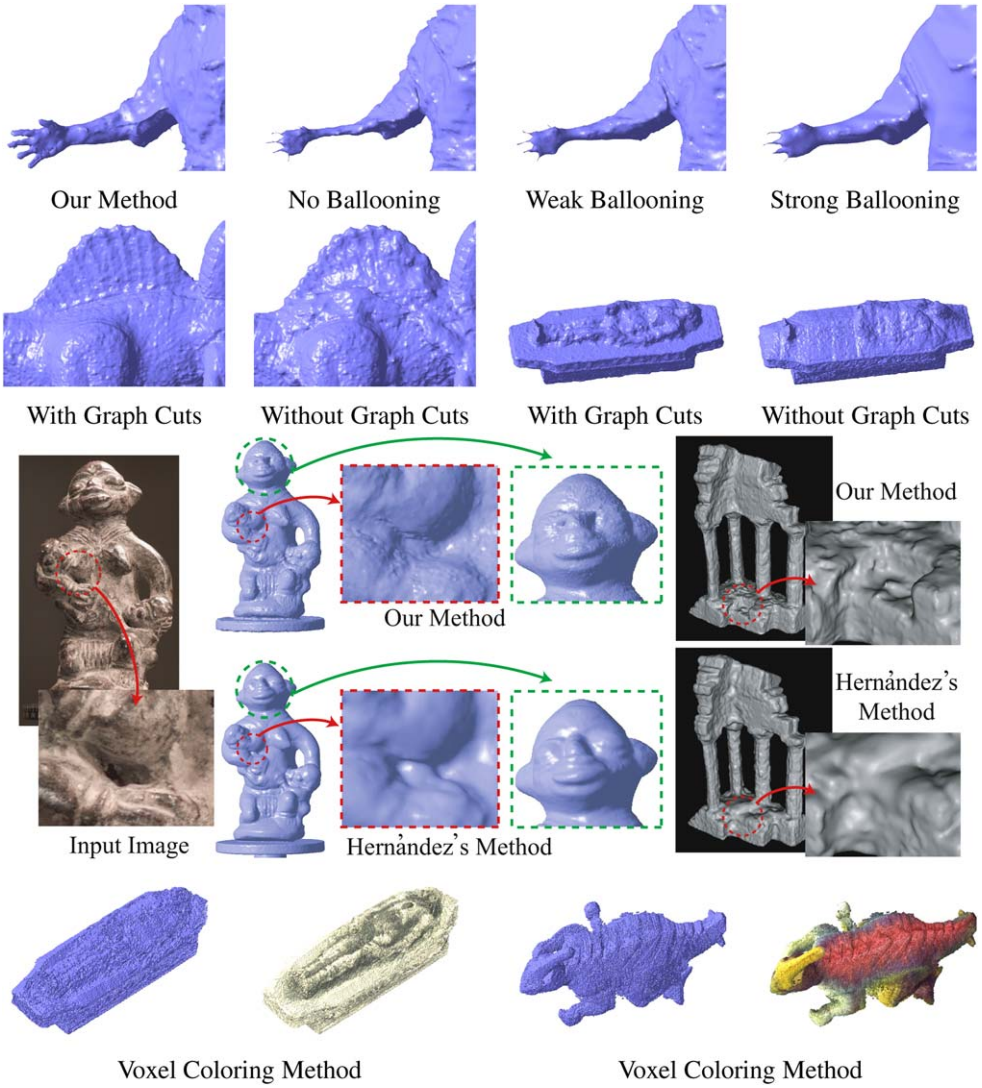

Fig. 14 A comparative evaluation of our algorithm. *First row*: comparison with our implementation of a variant of the method proposed by Vogiatzis et al. (2005). *Second row*: comparison with a purely local method initialized with the visual hull surface. *Third row*: comparison with a method by Hernández Esteban and Schmitt (2004). *Fourth row*: comparison with the voxel coloring method of Seitz and Dyer (1997). See text for details

Table 3 Quantitative evaluations on the *Temple* data set. The accuracy measure shows the distance d (in mm) that brings 90% of the result within the ground-truth surface, while the completeness measure shows the percentage of the ground-truth surface that lies within 1.25 mm of the result

	Accuracy (90%)	Completeness (1.25 mm)
Furukawa et al. (2007)	0.62 [mm]	99.2 [%]
Proposed Approach	0.75 [mm]	97.1 [%]
Hernández et al. (2004)	0.75 [mm]	95.3 [%]
Pons et al. (2005)	0.90 [mm]	95.4 [%]
Strecha et al. (2006)	1.05 [mm]	94.1 [%]
Tran et al. (2006)	1.53 [mm]	85.4 [%]
Vogiatzis et al. (2005)	2.77 [mm]	79.4 [%]

graph cuts framework is basically not suitable for recovering high-curvature structures. Note that in the Euclidean space, a minimal surface, which is an output of the graph cuts algorithm,[4] has zero mean curvature all over the surface by definition. 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. Second, we have tested our algorithm without its graph cuts phase, yielding a purely local method. Figure 14 (second row) shows two examples: the graph cuts step being included in

[4]See Boykov and Kolmogorov (2003) for the relationship between the minimal surface and the minimum cut.

the left part of the diagram, and omitted in the right part. As expected, local minimum problems are apparent in the latter case. Third, we have also compared our algorithm with the method proposed in Hernández Esteban and Schmitt (2004) on the *Twin* and *Temple* data sets. The two models of the *Twin* data set look pretty close to each other, which is partly because our local refinement step is basically the same as their refinement procedure (except that gradient flow forces instead of the direct derivatives are used to compute the image discrepancy term in their method), but there exists some differences. First, our method does not recover correct structures near the nose of the statue (illustrated by green circles), which is due to mistakes made in the rim-identification step. Note that our graph cuts step is vulnerable to a single mistake in the rim-identification step, and this is the reason why we have a conservative post-processing. However, it is still very difficult to avoid all the false-positives in some occasions as in Fig. 14 (see our future work in Sect. 7 to resolve this issue). On the other hand, our method outperforms Hernández Esteban's method in reconstructing a concave structure near the right hand of the statue as shown by the red circles (although the difference is not significant), which is also illustrated by the results on the *Temple* data set. Since their method is essentially a local iterative deformation, they have a problem in avoiding local minima, especially in challenging situations (e.g., sharp concave structures with weak textures), while our method has the global optimization step to handle them. Lastly, we have tried an implementation of *voxel coloring* (Kutulakos and Seitz 2000; Seitz and Dyer 1997), kindly provided by S. Seitz, on two of our examples (Fig. 14, bottom). The results appear rather noisy compared to ours (see Fig. 12), probably due to the lack of regularization, and several concavities are missed in the two objects (e.g., the chest of the mummy). Lastly, Table 3 presents some quantitative evaluations on the *Temple* data set with top performing multi-view stereo algorithms presented at the *Multi-View Stereo Evaluation* website (Seitz et al. 2007). As the table shows, the proposed method has achieved the second best result both in terms of accuracy and completeness. Note that our algorithm uses manually extracted silhouettes unlike the other methods, which gives us an undeniable advantage and prevents us from officially participating to the competition. Also note that the *Multi-View Stereo Evaluation* website provides data sets for one more object (*dino*), but we could not test our algorithm because the model is not fully visible in some views and exact silhouettes cannot be extracted there.

7 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 seven real data sets and some comparative experiments with state-of-the-art image-based modeling algorithms. One of the limitations of our current approach is that it cannot handle concavities too deep to be carved away by the graph cuts. The method is also vulnerable to mistakes made in the rim-identification step. To overcome these problems, we plan to combine our approach with recent work on sparse wide-baseline stereo from interest points (e.g., Schaffalitzky and Zisserman 2001) in order to incorporate stronger geometric constraints in the carving and local refinement stages (Furukawa and Ponce 2007). Attempting, as in Soatto et al. (2003), 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 (Uffenkamp 1993).

Acknowledgements 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 *Dinosaur-2* data set together with his voxel coloring software. We thank Jodi Blumenfeld and Steven R. Leigh for providing us with the *Skull* data set, and thank Carlos Hernández Esteban and Francis Schmitt for the *Twin* data set with their reconstruction result. We want to thank Sariel Har-Peled and Theodore Papadopoulo for discussions on the global optimization procedure. Lastly, we want to thank Daniel Scharstein for the quantitative evaluation of the *Temple* data set.

References

Baumgart, B. (1974). *Geometric modeling for computer vision*. Ph.D. thesis, Stanford University.

Boykov, Y., & Kolmogorov, V. (2003). Computing geodesics and minimal surfaces via graph cuts. In *ICCV* (pp. 26–33).

Cheung, K. M., Baker, S., & Kanade, T. (2003). Visual hull alignment and refinement across time: a 3D reconstruction algorithm combining shape-from-silhouette with stereo. In *CVPR*.

Delingette, H., Hebert, M., & Ikeuchi, K. (1992). Shape representation and image segmentation using deformable surfaces. *IVC*, *10*(3), 132–144.

Faugeras, O., & Keriven, R. (1998). Variational principles, surface evolution, PDE's, level set methods and the stereo problem. *IEEE Transactions on Image Processing*, *7*(3), 336–344.

Furukawa, Y., & Ponce, J. (2006). Carved visual hulls for image-based modeling. *ECCV*, *1*, 564–577.

Furukawa, Y., & Ponce, J. (2007). Accurate, dense, and robust multi-view stereopsis. In *CVPR*.

Goesele, M., Curless, B., & Seitz, S. M. (2006). Multi-view stereo revisited. In *CVPR* (pp. 2402–2409).

Habbecke, M., & Kobbelt, L. (2007). A surface-growing approach to multi-view stereo reconstruction. In *CVPR*.

Hernández Esteban, C., & Schmitt, F. (2004). Silhouette and stereo fusion for 3D object modeling. *CVIU*, *96*(3), 367–392.

Hoppe, H., DeRose, T., Duchamp, T., McDonald, J., & Stuetzle, W. (1993). Mesh optimization. In *SIGGRAPH* (pp. 19–26). New York: ACM Press.

Hornung, A., & Kobbelt, L. (2006). Hierarchical volumetric multi-view stereo reconstruction of manifold surfaces based on dual graph embedding. In *CVPR* (pp. 503–510).

Keriven, R. (2002). *A variational framework to shape from contours* (Technical Report 2002-221). ENPC.

Kolmogorov, V., & Zabih, R. (2002). Multi-camera scene reconstruction via graph cuts. *ECCV*, *3*, 82–96.

Kutulakos, K., & Seitz, S. (2000). A theory of shape by space carving. *IJCV*, *38*(3), 199–218.

Lachaud, J.-O., & Montanvert, A. (1999). Deformable meshes with automated topology changes for coarse-to-fine 3D surface extraction. *Medical Image Analysis*, *3*(2), 187–207.

Lazebnik, S., Furukawa, Y., & Ponce, J. (2007). Projective visual hulls. *International Journal of Computer Vision*, *74*(2), 137–165.

Matusik, W., Pfister, H., Ngan, A., Beardsley, P., Ziegler, R., & McMillan, L. (2002). Image-based 3D photography using opacity hulls. In *SIGGRAPH*.

Paris, S., Sillion, F., & Quan, L. (2004). A surface reconstruction method using global graph cut optimization. In *ACCV*.

Pons, J.-P., Keriven, R., & Faugeras, O. D. (2005). Modelling dynamic scenes by registering multi-view image sequences. In *CVPR (2)* (pp. 822–827).

Roy, S., & Cox, I. J. (1998). A maximum-flow formulation of the N-camera stereo correspondence problem. In *ICCV* (p. 492).

Schaffalitzky, F., & Zisserman, A. (2001). Viewpoint invariant texture matching and wide baseline stereo. In *ICCV*.

Seitz, S., & Dyer, C. (1997). Photorealistic scene reconstruction by voxel coloring. In *CVPR* (pp. 1067–1073).

Seitz, S. M., Curless, B., Diebel, J., Scharstein, D., & Szeliski, R. (2006). A comparison and evaluation of multi-view stereo reconstruction algorithms. *CVPR*, *1*, 519–528.

Seitz, S. M., Curless, B., Diebel, J., Scharstein, D., & Szeliski, R. (2007). *Multi-view stereo evaluation*. http://vision.middlebury.edu/mview/.

Sinha, S., & Pollefeys, M. (2004). Visual hull reconstruction from uncalibrated and unsynchronized video streams. In *Int. symp. on 3D data processing, visualization & transmission*.

Sinha, S., & Pollefeys, M. (2005). Multi-view reconstruction using photo-consistency and exact silhouette constraints: a maximum-flow formulation. In *ICCV*.

Soatto, S., Yezzi, A., & Jin, H. (2003). Tales of shape and radiance in multiview stereo. In *ICCV* (pp. 974–981).

Strecha, C., Fransens, R., & Gool, L. V. (2006). Combined depth and outlier estimation in multi-view stereo. In *CVPR* (pp. 2394–2401).

Tran, S., & Davis, L. (2006). 3D surface reconstruction using graph cuts with surface constraints. In *ECCV* (pp. II: 219–231).

Uffenkamp, V. (1993). State of the art of high precision industrial photogrammetry. In *Third international workshop on accelerator alignment*. Annecy, France.

Vogiatzis, G., Torr, P. H., & Cipolla, R. (2005). Multi-view stereo via volumetric graph-cuts. In *CVPR* (pp. 391–398).

Xu, C., & Prince, J. (1997). Gradient vector flow: a new external force for snakes. In *CVPR* (pp. 66–71).