Complete Object Modeling using a Volumetric Approach for Mesh Fusion

Giovanni Dainese, Marco Marcon, Augusto Sarti, Stefano Tubaro

Dipartimento di Elettronica e Informazione - Politecnico di Milano
Piazza Leonardo Da Vinci 32, 20133 Milano, Italy
dainese/marcon/sarti/tubaro@elet.polimi.it

1 ABSTRACT

In the past few years several systems for object reconstruction based on the analysis of 2D images have been proposed. In order for such systems to be of practical use, the 3D data extraction process is expected to be fast and reliable. In this paper we propose a general approach for the reconstruction of complete objects based on a mesh fusion algorithm. Every surface patch is obtained as a depth map using an algorithm based on graph cuts theory. Each depth map is then triangulated before using it in a fusion algorithm based on a volumetric function. The result of the process is a closed mesh representing the object surface.

2 INTRODUCTION

Reconstructing an object's tridimensional shape for a set of cameras is a classic vision problem. In the last few years, it has attracted a great deal of interest, partly due to the number of application both in vision and in graphics that require good reconstructions. In order to get the complete model of an object, we must extract 3D informations from a large set of cameras and this often leads to a time-expensive process. In this paper, we show how a divide and conquer approach can be used to speed up the entire process guaranteeing a good precision of the final model. In order to do this task, we chose to create a complete model of the interested object by linking together several surface patches reconstructed rapidly by a graph-cuts algorithm. The linking process is accomplished by a mesh fusion algorithm based on a volume of fluid approach, using a volumetric function.

3 DEPTH MAP RECONSTRUCTION

In this section we show how to reconstruct accurately a portion of the surface of the object present in the analyzed scene. This is the crucial step of the reconstruction process. In fact we will link the surface patches resulting from this step to obtain the final complete object by the fusion algorithm described in the next section. We have chosen to use the known *graph cuts* approach [1, 2, 3, 6], adapting it to the problem of depth map reconstruction.

In the next paragraph we propose a short description of the energy minimization approach; after that we will show how to formulate the problem of depth map reconstruction in term of energy minimization.

3.1 Energy minimization approach

Our approach to depth map reconstruction is similar to some recent work that give strong results for stereo matching and image restoration. It is well known that both problems can be elegantly stated in term of energy minimization [2, 3]. In the last few years powerful energy minimization algorithms have been developed based on graph cuts [2, 4, 5]. This methods are fast enough to be practical, but unlike simulated annealing, graph cuts methods cannot be applied to an arbitrary function. In this paper we will use some recent results [3] that give graph constructions for a quite general class of energy functions.

The energy minimization formalism has several

advantages. It allows a clean specification of the problem to be solved, as opposed to the algorithm used to solve it. In addiction, energy minimization naturally allows the use of soft constraints, such as spatial coherence. In an energy minimization framework, it is possible to cause ambiguities to be resolved in a manner that leads to a spatially smooth answer. Finally, energy minimization avoids being trapped by early hard decision.

3.2 Problem formulation

Suppose we are given n calibrated images of the same scene taken from different viewpoints (or at different moments of time). Let us assume a camera as the preferred one and let $\mathcal{P}$ be the set of pixels of the corresponding image. A pixel $p \in \mathcal{P}$ corresponds to a ray in 3D-space. Consider the first intersection of this ray with an object in the scene. Our goal is to find the depth of this point for all the pixel of the preferred image. So we want to find a labelling $f : \mathcal{P} \rightarrow \mathcal{L}$ where $\mathcal{L}$ is a discrete set of labels corresponding to increasing depths from the preferred camera. Equivalently, we want to obtain the *depth map* of the pixels in the preferred image.

A pair $\langle p, l \rangle$ where $p \in \mathcal{P}$, $l \in \mathcal{L}$ corresponds to some point in 3D-space. We will refer to such points as *3D-points*.

We define our energy function as consisting of two terms:

$$E(f) = E_{data}(f) + E_{smooth}(f)$$

In their work, Kolmogorov and Zabih [1] formulate the problem of scene reconstruction in a slightly different manner that permits to obtain a depth map for every image in the input set by an energy minimization approach. This leads to a computational expensive algorithm whose result is an unorganized cloud of points representing the surface of the visible part of the scene to reconstruct. Moveover, to have an effective reconstruction from the input set, cameras must respect some particular restrictive configuration, whereas with our definition we can treat a very large number of camera configurations without distinctions.

It can be also noted that in our approach the visibility term defined in [1] is no longer necessary . In fact, assuming that the set of label corresponds to the increasing depths from the preferred camera, there cannot exists occluding pixels in the same image and consequently the visibility term become unnecessary. Moreover, also the other terms are quite different. Our data term is defined as follow:

$$E_{data}(f) = \sum_{p \in \mathcal{P}} D(p)$$

where $D(p)$ is a non-positive value which results from the differences in intensity between corresponding pixels. $D(p)$ is computed for every pixel of the preferred image (we indicate this image with the index j) by this steps:

1. from p, we get the corresponding 3D-point by retroprojecting it from the preferred camera center of projection with the selected depth and then we project this 3D-point on each other calibrated image obtaining a set of $n-1$ corresponding pixels $\{q_1, q_2 \dots q_i \dots q_n | i \neq j\}$;

2. on every non-preferred image we compute the SSD (Sum of Square Difference) using a square window centered on q_i and the one centered on p, obtaining the set of values $\{d_1, d_2 \dots d_i \dots d_n | i \neq j\}$;

3. finally,

$$D(p) = min(0, \sum_{\substack{i=1 \\ i \neq j}}^{n} d_i - K) \qquad (1)$$

where K is a positive constant large enough to capture significant variation of the SSD function (a typical value is $K = 30$).

The smoothness term is quite similar to the one used in [1] and its goal is to make neighboring pixels in the preferred image tend to have similar depths. The smoothness term is defined as follow:

$$E_{smooth}(f) = \sum_{\{p,q\} \in \mathcal{N}} V_{\{p,q\}}(f(p), f(q)) \qquad (2)$$

This term involves a notion of neighborhood: we assume that there is a neighborhood system on pixel

$$\mathcal{N} \subset \{\{p, q\} \mid p, q \in \mathcal{P}\}$$

This can be the usual 4-neighborhood system: pixels $p = (p_x, p_y)$ and $q = (q_x, q_y)$ are neighbors if they are in the same image and $|p_x - q_x| + |p_y - q_y| = 1$.

In [1], the function $V_{\{p,q\}}$ assumes the following form:

$$V_{\{p,q\}}(l_p, l_q) = \begin{cases} U_{\{p,q\}} & \text{if } l_p \neq l_q \\ 0 & \text{otherwise} \end{cases} \quad (3)$$

where the $U_{\{p,q\}}$ is the following non-decreasing function:

$$U_{\{p,q\}} = \begin{cases} 3\lambda & \text{if } \Delta I(p,q) < 5 \\ \lambda & \text{otherwise} \end{cases} \quad (4)$$

To make the reconstruction smooth while preserving discontinuities, we choose to follow a particular strategy in the use of the smoothness term. In fact, it is known that graph cuts techniques often yields flat and blocky results. This may not be important for disparity maps, but is crucial for shape reconstruction. To avoid this problem, we make a first cycle of the reconstruction algorithm with a limited set of labels, in order to reach rapidly a value of the energy near to the local minimum that could be got at convergence with the original algorithm. This corresponds to a good approximation of the position of the 3D-points, that can be improved with a second cycle at double resolution where we change the function $V_{\{p,q\}}$ defined in (3) with this new function:

$$\hat{V}_{\{p,q\}}(l_p, l_q) = \begin{cases} U_{\{p,q\}} & \text{if } |l_p - l_q| > z\_threshold \\ 0 & \text{otherwise} \end{cases} \quad (5)$$

In fact, this function relaxes the penalty mechanism of the smoothness term, giving a 0 penalty not only to the neighboring pixels that lie at the same depth but also to the ones that stay sufficiently near one another. The idea is supported by the fact that after the first cycle of the algorithm, only some of the pixels are approximatively well positioned in 3D-space by the consistency measure given by the data term, while the other are positioned only by the effect of the smoothness term which forces them to lie on the same level of neighboring pixel, resulting in flat blocks. Thus, relaxing the constraint imposed by the first smoothness term, neighboring pixels have greater chance to occupy adjacent depths correctly.

3.3 Graph construction

We now show how to efficiently minimize E among all configurations using graph cuts. The output of our method will be a local minimum in a strong sense. In particular, consider an input configuration f and a disparity α. Another configuration f' is defined to be within a single $\alpha-expansion$ of f when for all pixels $p \in \mathcal{P}$ either $f'(p) = f(p)$ or $f'(p) = \alpha$. This notion of an expansion was proposed by [4], and forms the basis for several very effective stereo algorithms [4, 9, 10, 11]. Our algorithm is very straightforward; we simply select in a fixed order a disparity α, and we find the unique configuration within a single $\alpha - expansion$ move (our local improvement step). If this decreases the energy, then we go there; if there is no α that decreases the energy, we are done. Except for the problem formulation and the choice of energy function, this algorithm is identical to the methods of [4, 11]. Differently from the work of [1] we have no restrictions on the initial configuration of the depths. In fact, the absence of a visibility term avoid the problem of the occluding pixels, cited in [1]. The critical step in our method is to efficiently compute the $\alpha - expansion$ with the smallest energy. In this section, we show how to use graph cuts to solve this problem.

3.3.1 Graph cuts

Let $\mathcal{G} = \mathcal{V}, \mathcal{E}$ be a weighted graph with two distinguished terminal vertices $\{s, t\}$ called the source and sink. A *cut* $\mathcal{C} = \mathcal{V}^s, \mathcal{V}^t$ is a partition of the vertices into two sets such that $s \in \mathcal{V}^s$ and $t \in \mathcal{V}^t$. (Note that a cut can also be equivalently defined as the set of edges between the two sets.) The cost of the cut, denoted $|\mathcal{C}|$, equals the sum of the weights of the edges between a vertex in $\mathcal{V}^s$ and a vertex in $\mathcal{V}^t$. The minimum cut problem is to find the cut with the smallest cost. This problem can be solved very efficiently by computing the maximum flow between the terminals, according to a theorem due to Ford and Fulkerson [7]. There are a large number of fast algorithms for this problem (see [8], for example), for example). The worst case complexity is low-order polynomial; however, in practice the running time is nearly linear for graphs with many short paths between the source and the sink, such as the one we will construct. We will use a

result from [3] which says that for energy functions of binary variables of the form

$$E(x_1, \ldots, x_n) = \sum_i E^i(x_i) + \sum_{i<j} E^{i,j}(x_i, x_j) \quad (6)$$

it is possible to construct a graph for minimizing it if and only if each term $E^{i,j}$ satisfies the following condition:

$$E^{i,j}(0,0) + E^{i,j}(1,1) \leq E^{i,j}(0,1) + E^{i,j}(1,0) \quad (7)$$

If these conditions are satisfied then the graph $\mathcal{G}$ is constructed as follows. We add a node $v_i i$ for each variable x_i. For each term $E^{i,j}(x_i, x_j)$ we add edges which are given in the appendix. Every cut on such a graph corresponds to some configuration $x = (x_1, \ldots, x_n)$, and vice versa: if $v_i \in \mathcal{V}^s$ then $x_i = 0$, otherwise $x_i = 1$. Edges on a graph were added in such a way that the cost of any cut is equal to the energy of the corresponding configuration plus a constant. Thus, the minimum cut on $\mathcal{G}$ yields the configuration that minimizes the energy.

3.3.2 $\alpha -$ expansion

In this section we will show how to convert our energy function into the form of equation (6). Note that it is not necessary to use only terms $E^{i,j}$ for which $i < j$ since we can swap the variables if necessary without affecting condition (7). Although pixels can have multiple labels and in general cannot be represented by binary variables, we can do it for the $\alpha - expansion$ operation. Indeed, any configuration f' within a single $\alpha - expansion$ of the initial configuration f can be encoded by a binary vector $x = \{x_p | p \in \mathcal{P}\}$ where $f'(p) = f(p)$ if $x_p = 0$, and $f'(p) = \alpha$ if $x_p = 1$. Let us denote a configuration defined by a vector x as f^x. Thus, we have the energy of binary variables:

$$\widetilde{E}(x) = \widetilde{E}_{data}(x) + \widetilde{E}_{smoothness}(x)$$

where

$$\begin{aligned} \widetilde{E}_{data}(x) &= E_{data}(f^x), \\ \widetilde{E}_{smoothness}(x) &= E_{smoothness}(f^x). \end{aligned}$$

Let's consider each termseparately, and show that each satisfies condition (7).

1. Data term.

   $$\widehat{E}_{data}(x) = \sum_{p \in \mathcal{P}} D(p)$$

   Each term in the sum is a function of one binary variable, and so it automatically satisfies the condition 7. In fact, we have:

   $$E^{i,j}(0,0) + E^{i,j}(1,1) = E^{i,j}(0,1) + E^{i,j}(1,0)$$

   because the four terms are indipendent from the second variable.

2. Smoothness term

   $$\widehat{E}_{smoothness}(x) = \sum_{p,q \in \mathcal{N}} V_{p,q}(f^x(p), f^x(q))$$

   Let's consider a single term $E^{p,q}(x_p, x_q) = V_{p,q}(f^x(p), f^x(q))$. We assumed that $V_{p,q}$ is a metric; thus, we have $V_{p,q}(\alpha, \alpha) = 0$ and then

   $$V_{p,q}(f(p), f(q)) \leq V_{p,q}(f(p), \alpha) + V_{p,q}(\alpha, f(q))$$

   or, equivalently

   $$E^{p,q}(0,0) \leq E^{p,q}(0,1) + E^{p,q}(1,0)$$

   Therefore, condition 7 holds.

3.4 Depth map optimization

Even though the graph cuts algorithm is able to reconstruct an accurate depth map, it works only with a limited set of depths and, thus, it introduce a considerable quantization error in the position of each 3D-point. To overcome this problem, it is necessary an optimization step which yields the depth map more regular. The output of this process is a new depth map, where the discontinuities are preserved while the other parts become smoothed.

To do this work, we consider the depth map as a function of two variables defined on the preferred image and we apply a sequence of bidimensional filters on it. In particular, we start with a median filter to eliminate possible outliers and then we apply a dithering technique: some white noise is added to the depth function and, then, a low pass filter is used to yields the depth map smooth. To preserve discontinuities, the bidimensional low pass

filter keeps the information needed from the neighbors of a pixel only if the depth distance is below a certain threshold. The size of the filters windows and this threshold are empirically chosen on the basis of the current reconstruction. For instance, the median filter must be small to avoid the appearance of flat zones in the depth map (a typical window is a square of 7x7 or 9x9 pixels), while the low pass filter cannot be greater than 3x3 or 5x5, otherwise the reconstructed map become too smooth, losing details. The distance threshold can be measured as a discrete number of depth levels. Typically the number of the limited set of labels must be large enough to capture the details of the object we are reconstructing. In this point of view, two points are along a shape discontinuity if they lay on two depths which differs one another of about 8-12 levels. So we usually choose the threshold as 10 depth levels.

4 MESH FUSION BY A VOLUMETRIC APPROACH

In this section we describe a volumetric approach to the problem of mesh fusion. From the previous section, we have seen how to compute depth maps from a set of input image of a particular scene. In the first paragraph we show how to compute a triangulation of these depth maps, in order to achieve a their topological representation. The result of this process is a set of overlapping meshes that approximate the surface of the object in the scene. In the next paragraph, we show how a fusion algorithm can be used to melt together the various meshes to obtain a final and closed mesh that describe the object entirely.

4.1 Mesh triangulation

From the previous section we learnt how to compute a depth map from a set of images of the interested object. We also said that every map can be seen as a bidimensional function defined on the preferred image. Starting from this point, we can easily implement a triangulation algorithm that produce a mesh from a depth map on the basis of the neighboring pixels. Consider four neighboring points and the six possible connection shown in figure 1:

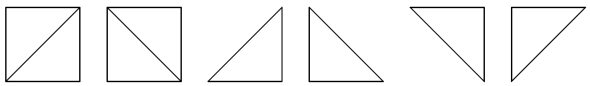


Figure 1: Six possible configurations for the creation of triangles from four neighboring points.

when two neighboring pixels have depths differing by more than some threshold, there is a step discontinuity. The threshold can be directly determined, as the maximum depth difference which has to be considered a surface discontinuity. If a discontinuity is present, a triangle should not be created. Therefore, for four neighboring pixels, we only consider 3D-points that are not along discontinuities. If three of them satisfy this condition, a triangle will be created in one of the last four style in figure 1. If none of the four points are along a discontinuity, two triangles will be created, and the common edge will be the one with the shortest 3D distance, as shown in the first two styles in figure 1.

4.2 A volumetric approach to mesh fusion

Once we have triangulate the depths maps, we are ready to melt them together. We have implemented a method following the subsequent criteria:

- get good results rapidly;
- reproduce accurately the partial meshes obtained from the depth maps in the final mesh, removing the uncertainty due to possible overlappings;
- produce a closed model made by a unique mesh, overcoming the possible lack of information in the object surface.

We have chosen a volumetric representation of the scene that use a voxelset made of cubic voxels, with an approach similar to the already known volume-of-fluid technique. Each voxel can assume a value in the $[-1, +1]$ interval. The entire voxelset can be seen as a volumetric function which represents the surface of the object as the zero levelset. Negative values of the function indicate the space inside the object while positive values stay for external space. Near the surface, each voxel assumes

an intermediate value on the basis of its distance from the closer mesh. The algorithm starts initializing every voxel to the -1 value. By this way the volumetric function represents a solid block where subsequent steps will carve the surface of the object. At this point, we select a mesh and assign a value to each voxel of the voxelset with the steps explained in figure 2 and following this criterion: **a voxel value can only be changed with a greater one**.

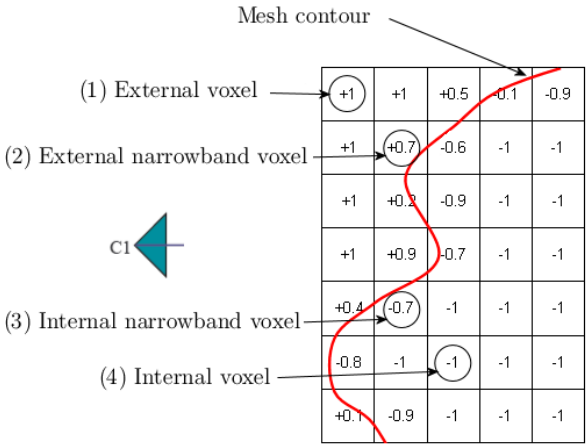


Figure 2: Modelling of the volumetric function from a mesh.

Each voxel falls in one of these four categories on the basis of its position and of the distance between its center and the current mesh:

(1) External voxels: they are situated between the mesh and the current camera and their distance is greater then the voxel semi-diagonal.

(2) External narrowband voxels: they are situated between the mesh and the current camera and their distance is smaller then the voxel semi-diagonal.

(3) Internal narrowband voxels: they are situated after the mesh starting from the current camera and their distance is smaller then the voxel semi-diagonal.

(4) Internal voxels: they are situated after the mesh starting from the current camera and their distance is greater then the voxel semi-diagonal.

Repeating this steps for every mesh will lead to a volumetric function whose zero levelset locates the object surface. The resulting object can be seen as a sort of convex hull obtained by linking together the meshes and taking only the part of the 3D space contained in their intersection.

5 EXPERIMENTAL RESULTS

The proposed algorithm has been applied to a set of images of a synthetic teapot and then to a set of images of a real object, a skull, acquired with a trinocular calibrated camera system. The teapot has been modeled by a 3D software and several snapshots have been rendered from it. The skull has been placed on a turntable and a sequence of snapshots has been taken for every position of the turntable. For both objects, some images have been selected in triplets. From each triplet a depth map is reconstructed. Figure 3(a) shows a triplet of images of the teapot, while figure 3(b) shows the corresponding reconstructed surface patch; the complete object reconstructed by the volumetric algorithm is visualized in figure 3(c). Analogously, figure 4(a) shows a triplet of images of the skull, while figure 4(b) shows the corresponding surface patch; the complete model of the skull is shown in figure 4(c). Both the final teapot and skull model have been obtained by melting together eight surface patches taken from different position.

The parameters K of equation (1) and λ of equation (4) are determined heuristically: optimal values depend on the images we are processing. The parameters can be varied to gain some insight about the algorithm: for big values of λ the smoothness dominates the correlation, resulting in a map with many flat blocks of pixels, whereas little values of λ yields to an irregular depth map with many wrong discontinuities. In our experiment, we chose the values $K = 30$ and $\lambda = 5$.

5.1 Conclusions

3D reconstruction from a set of images is a critical process. In order to perform this task we presented a reconstruction algorithm based on graph cuts theory. We have defined an energy function whose minimum represents the solution to our problem and we implemented a technique to raffinate the obtained depth maps.

A virtue of this approach is the algorithm speed. In fact, we chose to build up a complete model of an object linking together several depth maps, reducing the computational effort either in the time needed and in the memory space required for reconstruct each of them. A volumetric approach has been used to do this task.

A APPENDIX: EDGES FOR TERMS E^i and $E^{i,j}$

In this section we show how to add edges for the terms E^i and $E^{i,j}$ in the equation (6) assuming that condition 7 holds. This is a special case of a more general construction given in [3].

1. **terms in the form E^i:**
   - if $E(0) > E(1)$ then we add an edge (s, v_i) with the weight $E(0)-E(1)$, otherwise we add an edge (v_i, t) with the weight $E(1)-E(0)$;
2. **terms in the form $E^{i,j}$:**
   - if $E(1,0) > E(0,0)$ then we add an edge (s, v_i) with the weight $E(1,0) - E(0,0)$, otherwise we add an edge (v_i, t) with the weight $E(0,0) - E(1,0)$;
   - if $E(1,0) > E(1,1)$ then we add an edge (v_j, t) with the weight $E(1,0) - E(1,1)$, otherwise we add an edge (s, v_j) with the weight $E(1,1) - E(1,0)$;
   - the last edge that we add is (v_i, v_j) with the weight $E(0,1) + E(1,0) - E(0,0) - E(1,1)$.

We have omitted indices i in E^i and i, j in $E^{i,j}$ for simplicity of notation. Of course it is not necessary to add edges with zero weights. Also, when several edges are added from one node to another, it is possible to replace them with one edge with the weight equal to the sum of weights of individual edges.

References

[1] V. Kolmogorov and R. Zabih. "Multi-camera Scene Reconstruction via Graph Cuts". *In European Conference on Computer Vision*, 2002.

[2] V. Kolmogorov and R. Zabih. "Computing Visual Corrispondence with Occlusion via Graph Cuts". *In International Conference on Computer Vision*, 2001.

[3] V. Kolmogorov and R. Zabih. "What energy functions can be minimized via graph cuts?" *In European Conference on Computer Vision*, 2002.

[4] Y. Boykov, O. Veksler and R. Zabih. "Fast Approximate Energy Minimization via Graph Cuts". *IEEE Transaction on Pattern Analysis and Machine Intelligence*, 2001.

[5] Y. Boykov, O. Veksler and R. Zabih. "Markov Random Fields with efficient approximations". *IEEE Conference on Computer Vision and Pattern Recognition*, 1998.

[6] D. Snow, P. Viola and R. Zabih. "Exact Voxel Occupancy with Graph Cuts". *In Proc. Computer Vision and Pattern Recognition Conf.*, 2000.

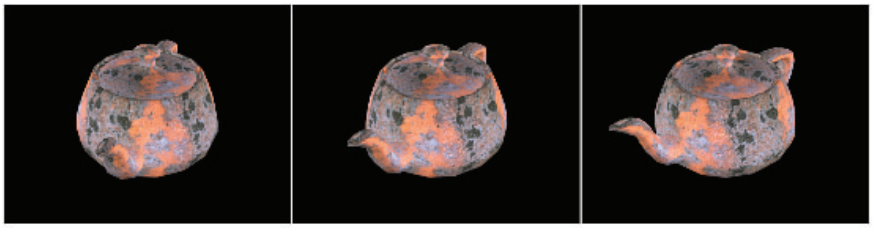
[7] L. Ford and D. Fulkerson. "Flows in Networks". *Princeton University Press*, 1962.

[8] F. Maffioli. "Elementi di programmazione matematica". Casa Editrice Ambrosiana, 2000.

[9] S.B. Kang, R. Szeliski, and J. Chai. Handling occlusions in dense multi-view stereo. *In IEEE Conference on Computer Vision and Pattern Recognition*, 2001. Expanded version available as MSR-TR-2001-80.

[10] S. Birchfield and C. Tomasi. Multiway cut for stereo and motion with slanted surfaces. *In International Conference on Computer Vision*, pages 489-495, 1999.

[11] V. Kolmogorov and R. Zabih. Visual correspondence with occlusions using graph cuts. *In International Conference on Computer Vision*, pages 508-515, 2001.



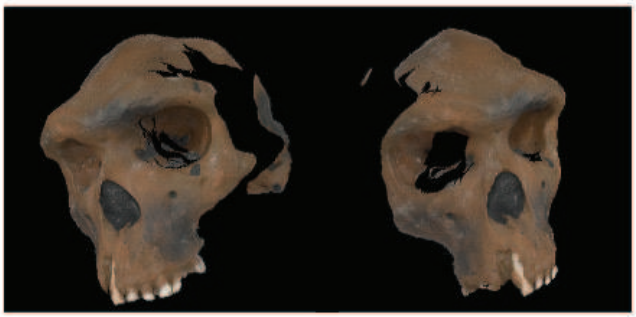
(a)

(b)

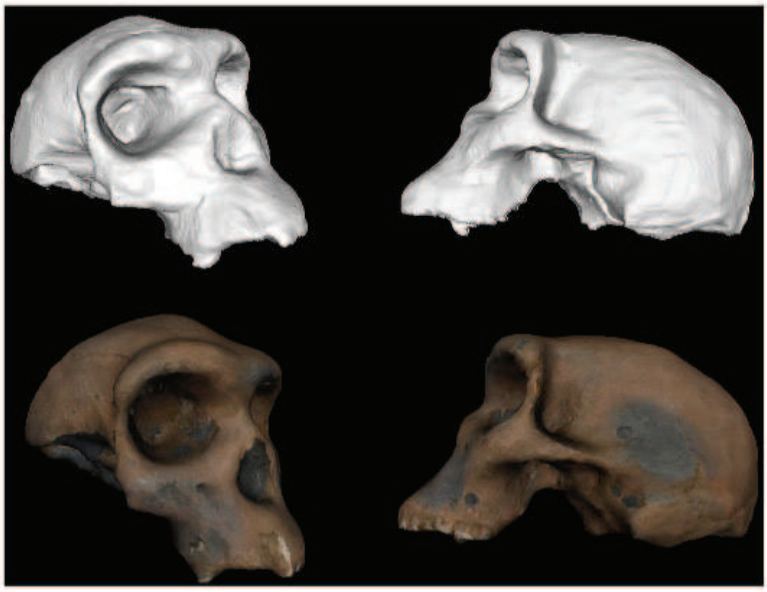
(c)

Figure 3: (a) teapot image triplet. (b) corresponding surface patch (c) complete model of the teapot

(a)



(b)



(c)

Figure 4: (a) skull image triplet. (b) corresponding surface patch (c) complete model of the skull