Silhouette and Stereo Fusion for 3D Object Modeling

Carlos Hernández Esteban and Francis Schmitt

Signal and Image Processing Department, CNRS UMR 5141
Ecole Nationale Supérieure des Télécommunications, France

Abstract

In this paper, we present a new approach to high quality 3D object reconstruction. Starting from a calibrated sequence of color images, the algorithm is able to reconstruct both the 3D geometry and the texture. The core of the method is based on a deformable model, which defines the framework where texture and silhouette information can be fused. This is achieved by defining two external forces based on the images: a texture driven force and a silhouette driven force. The texture force is computed in two steps: a multi-stereo correlation voting approach and a gradient vector flow diffusion. Due to the high resolution of the voting approach, a multi-grid version of the gradient vector flow has been developed. Concerning the silhouette force, a new formulation of the silhouette constraint is derived. It provides a robust way to integrate the silhouettes in the evolution algorithm. As a consequence, we are able to recover the contour generators of the model at the end of the iteration process. Finally, a texture map is computed from the original images for the reconstructed 3D model.

Key words: 3D reconstruction, deformable model, multigrid gradient vector flow, visual hull, texture.

1 Introduction

As computer graphics and technology become more powerful, attention is being focused on the creation or acquisition of high quality 3D models. As a result, a great effort is being made to exploit the biggest source of 3D models: the real world. Among all the possible techniques of 3D acquisition, there is one which is especially attractive: the image-based modeling. In this kind of approach, the only input data to the algorithm are a set of images, possibly calibrated. Its main advantages are the low cost of the system and the possibility of immediate color.

Email addresses: `carlos.hernandez@enst.fr`, `francis.schmitt@enst.fr` (Carlos Hernández Esteban and Francis Schmitt).

The main disadvantage is the quality of the reconstructions compared to the quality of more active techniques (range scanning or encoded-light techniques). In this paper we present an image-based modeling approach which affords high quality reconstructions by mixing two complementary image data into a same framework: silhouette information and texture information. Our two main contributions are a new approach to the silhouette constraint definition and the high quality of the overall system.

2 Related Work

Acquiring 3D models is not an easy task and abundant literature exists on this subject. There are three major approaches to the problem of 3D real model representation: pure image-based rendering techniques, hybrid image-based techniques, and 3D scanning techniques. Pure image-based rendering techniques as [1,2] try to generate synthetic views from a given set of original images. They do not estimate the real 3D structure behind the images, they only *interpolate* the given set of images to generate a synthetic view. Hybrid methods as [3–6] make a rough estimation of the 3D geometry and mix it with a traditional image-based rendering algorithm in order to obtain more accurate results. In both types of methods, the goal is to generate coherent views of the real scene, rather than obtain metric measures of it. In opposition to these techniques, the third class of algorithms tries to recover the full 3D structure. Among the 3D scanning techniques, two main groups are to be distinguished: active methods and passive ones. Active methods use a controlled source of light such as a laser or a coded light in order to recover the 3D information [7–9]. Passive methods use only the information contained in the images of the scene [10]. They can be classified according to the type of information they use. A first class consists of the shape from silhouette methods [11–16]. They obtain an initial estimation of the 3D model known as visual hull. They are robust and fast, but because of the type of information used, they are limited to simple shaped objects. We can find commercial products based on this technique. Another approach includes the shape from shading methods. They are based on the diffusing properties of Lambertian surfaces. They mainly work for 2.5D surfaces and are very dependent on the light conditions. A third class of methods use the color information of the scene. The color information can be used in different ways, depending on the type of scene we try to reconstruct. A first way is to measure color consistency to carve a voxel volume [17,18]. But they only provide an output model composed of a set of voxels, which makes it difficult to obtain a good 3D mesh representation. In order to solve this problem, the authors of [19] and [20] propose to use the color consistency measure to guide a deformable model. An additional problem of color consistency algorithms is that they compare absolute color values, which makes them sensitive to light condition variations. A different way of exploiting color is to compare local variations of the texture, as done in cross-correlation methods [21,22]. As a specialization of the color-based group, there are specific methods that try to use at the same time another type of information such as silhouettes [17,23,24], radiance [25] or shading [26]. Although very good results are obtained, the quality is still limited, and the main problem is the way the fusion of different data is done. Some authors, such as [17,23], use a volume grid for the fusion. Others, like [26,25,24], use a deformation model framework. The algorithm we present in this paper can be classified in this latter group. We perform the fusion of both silhouettes and texture information by a deformation model evolution. The main difference with the methods mentioned above is the way the fusion is accomplished, which enables us to obtain very high quality reconstructions. A similar approach to our work has been recently proposed in [27]. A deformable model is also used to fusion texture and silhouette information. However, the objectives of their work are not the same as ours. They are interested in dynamic 3D shape reconstruction of moving persons while our specific aim is high quality 3D and color reconstructions of museological objects.

3 Algorithm Overview

The goal of the system is to be able to reconstruct a 3D object from a sequence of geometrically calibrated images. To do so, we dispose of several types of information contained in the images. Among all the information available, shading, silhouettes and features of the object are the most useful for shape retrieval. Shading information needs a calibration of the light sources, which implies an even more controlled environment for the acquisition. The use of the silhouettes requires a good extraction of the object from the background, which is not always easy to accomplish. Finally, of all the features available from an object, such as texture, points, contours, or more complicated forms, we are mainly interested in texture, whenever it exists. Since exploiting shading imposes heavy constraints in the acquisition process, the information we will use consists of silhouettes and texture. The next step is to decide how to mix these two types of information to work together. As we will see, this is not an easy task because those types of information are very different, almost ”orthogonal”.

3.1 Classical Snake vs. Level-Set Methods

Deformation models offer a well-known framework to optimize a surface under several kinds of information. Two different related techniques can be used depending on the way the problem is posed: a classical snake approach [28] or a level-set approach [29]. The main advantage of the snake approach is its simplicity of implementation and parameter tuning. Its main drawback is the constant topology constraint. Level-set based algorithms have the advantage of an intrinsic capability to overcome this problem but its main disadvantages are the computation time and the difficulty in controlling the topology. Computation time can be addressed using a narrow band implementation [30]. Controlling the topology is a more difficult problem but, the authors of [31] have recently proposed an interesting way of avoiding topology changes in level set methods. Despite these improvements, level-set methods remain complex and expensive when dealing with high resolution deformable models (9 to 11 grid levels). Since this is our principal objective, we have chosen to use the classical snake as framework for the fusion of silhouette and stereo data. This implies that the topology has to be completely recovered before the snake evolution occurs as discussed in Section 4. Since the proposed way to recover the right topology is the visual hull concept, the topology recovery will depend on the intrinsic limitations of the visual hull. This implies that there exist objects for which we are unable to recover the correct topology (no silhouettes seeing a hole) that could be potentially reconstructed using a level-set method (the correct topology being recovered with the stereo information). We observe that, in practice, if we dispose of enough views, the visual hull provides the correct topology for most of the common objects; therefore, this is not a severe handicap.

3.2 The Classical Snake Approach

The deformable model framework allows us to define an optimal surface which minimizes a global energy $\mathscr{E}$. In general, this energy will be non-convex with possible local minima. In our case, the minimization problem is posed as follows: find the surface S of $\mathbb{R}^3$ that minimizes the energy $\mathscr{E}(S)$ defined as follows:

$$\mathscr{E}(S) = \mathscr{E}_{tex}(S) + \mathscr{E}_{sil}(S) + \mathscr{E}_{int}(S), \tag{1}$$

where $\mathscr{E}_{tex}$ is the energy term related to the texture of the object, $\mathscr{E}_{sil}$ the term related to the silhouettes and $\mathscr{E}_{int}$ is a regularization term of the surface model. Minimizing Eq. (1) means finding S_{opt} such that:

$$\begin{aligned} \nabla\mathscr{E}(S_{opt}) &= \nabla\mathscr{E}_{tex}(S_{opt}) + \nabla\mathscr{E}_{sil}(S_{opt}) + \nabla\mathscr{E}_{int}(S_{opt}) = 0, \\ &= \mathscr{F}_{tex}(S_{opt}) + \mathscr{F}_{sil}(S_{opt}) + \mathscr{F}_{int}(S_{opt}) = 0, \end{aligned} \tag{2}$$

where ∇ is the gradient operator, and $\mathscr{F}_{tex}$, $\mathscr{F}_{sil}$ and $\mathscr{F}_{int}$ represent the forces that drive the snake. Equation (2) establishes the equilibrium condition for an optimal solution, where the three forces cancel each other out. A solution to Eq. (2) can be found by introducing a time variable t for the surface S and solving the following differential equation:

$$S_t = \mathscr{F}_{tex}(S) + \mathscr{F}_{sil}(S) + \mathscr{F}_{int}(S). \tag{3}$$

The discrete version becomes:

$$S^{k+1} = S^k + \Delta t(\mathscr{F}_{tex}(S^k) + \mathscr{F}_{sil}(S^k) + \mathscr{F}_{int}(S^k)). \tag{4}$$

Once we have sketched the energies that will drive the process, we need to make a choice for the representation of the surface S. This representation defines the way the deformation of the snake is done at each iteration. Among all the possible surface representations, we have chosen the triangular mesh because of its simplicity and well known properties.

To completely define the deformation framework, we need an initial value of S, i.e., an initial surface S_0 that will evolve under the different energies until convergence.

In this paper, we describe the snake initialization in Section 4, the force driven by the texture of the object in Section 5, the force driven by the silhouettes in Section 6, how we control the mesh evolution in Section 7 and the texture mapping procedure in Section 8. We finally discuss our results in Section 9.

4 Snake Initialization

The first step in our minimization problem is to find an initial surface *close enough* to the object surface in order to guarantee a good convergence of the algorithm. *Close* has to be considered in a geometrical and topological sense. The geometric distance between the initial and the object surfaces has to be reduced in order to limit the number of iterations in the surface mesh evolution process and thereby the computation time. The topology of the initial surface is also very important since classical deformable models maintain the topology of the mesh during its evolution. On the one hand, this imposes a strong constraint that makes the initialization a very important step since the initial surface must capture the topology of the object surface. On the other hand, the topology-constant property of a classical snake provides more robustness to the evolution process.

If we make a list of possible initializations, we can establish an ordered list, where the first and simplest initialization is the bounding box of the object. The next simplest surface is the convex hull of the object. Both the bounding box and the convex hull are unable to represent surfaces with a genus greater than 0. A more refined initialization, which lies between the convex hull and the real object surface is the visual hull [32]. The visual hull can be defined as the intersection of all the possible cones containing the object. In practice, a discrete version is usually obtained by intersecting the cones generated by back projecting the object silhouettes of a given set of views. As a difference with the convex hull, it can represent surfaces with an arbitrary number of holes. However, this does not imply that it is able to completely recover the topology of the object and, what is even worse, the topology of the visual hull depends on the discretization of the views (see Fig. 1).

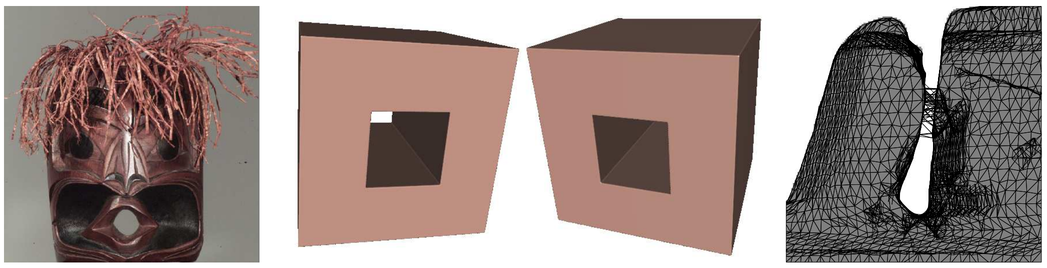

Fig. 1. Different topological problems. Left: example of a topology that cannot be captured by the visual hull concept. Middle: example of topological problem arising with a finite number of cameras. The first camera is able to recover the right topology whereas the second camera is not. Right: bad topology caused by the resolution of the visual hull construction algorithm. We show an original silhouette superposed to the visual hull mesh.

Computing the visual hull from a sequence of images is a very well known problem of computer vision and computer graphics [12,14,4]. Different approaches exist, depending on the type of output, way of representation and fidelity to the theoretical visual hull. In our case, we are interested in methods producing good quality meshes (manifold, smooth, triangles with an aspect ratio close to 1, defined as the ratio of the circumradius of the triangle to twice its inradius), even if the fidelity is not very high. In addition to a good quality mesh, another primary requirement is to obtain the right topology. Volume carving methods are a good choice because of the high quality output meshes that we can obtain through a marching cube [33] or marching tetrahedron algorithm. The degree of precision is fixed by the resolution of the volume grid, which can be adapted according to the required output resolution. But this adaptability can also generate additional problems of topology: if the resolution of the grid is low compared to the size of the visual hull structures, the aliasing produced by the sub-sampling may produce topological artifacts that the theoretic visual hull does not have. To sum up, three different sources of deviation may arise between the real object topology and the computed visual hull topology:

- **Errors due to the nature of the visual hull** (see Fig. 1 left). Real objects may have holes that cannot be seen as a silhouette hole from any point of view. The visual hull will then fail to represent the correct topology for this kind of object.
- **Errors due to the use of a finite number of views** (see Fig. 1 middle). They can be solved by having the adequate points of view that allow recovering the right topology of the real object.
- **Errors due to the implementation algorithm** (see Fig. 1 right). They are caused by the numerical precision or the subsampling of the silhouettes. They can be avoided by increasing the precision of the algorithm or by filtering the silhouettes.

In practice, we use an octree-based carving method followed by a marching tetrahedron meshing algorithm and a mesh simplification. In order to initialize the octree, an initial bounding box can be analytically computed from the 2D silhouette bounding boxes. The 3D back projection of *n* 2D bounding boxes defines a 3D convex

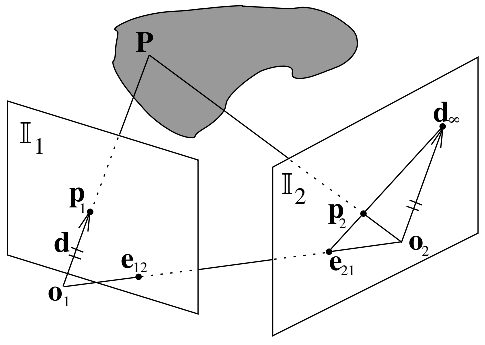

Fig. 2. Epipolar geometry.

hull formed by $4n$ half planes. The bounding box of the convex hull can be analytically computed by a simplex optimization method for each of the 6 variables defining the bounding box.

5 Texture Driven Force

In this section we further develop the texture force $\mathscr{F}_{tex}$ appearing in Eq. (2). This force contributes to recovering the 3D object shape by exploiting the texture of the object. We want this force to maximize the image coherence of all the cameras that see the same part of the object. It is based on the following projective geometry property: if two cameras see the same surface, then the two images are related by a geometric transformation that depends only on the 3D geometry of the object. This property is only fully valid under the common hypothesis of perfect projective cameras, perfect Lambertian surface and same lighting conditions. Different approaches exist to measure the coherence of a set of images, but they can be classified into two main groups whether they make a punctual radiometric comparison (e.g. photo-consistency measures as in voxel coloring [18]) or a spatial comparison of relative radiometric distributions (e.g. cross-correlation measures). We have chosen the normalized cross-correlation because of its simplicity and robustness in the presence of highlights and changes of the lighting conditions. Using the example of Fig. 2, the normalized cross-correlation $C(\mathbf{p}_1,\mathbf{p}_2)$ between pixels $\mathbf{p}_1$ and $\mathbf{p}_2$ is defined as follows:

$$C(\mathbf{p}_1,\mathbf{p}_2) = \mathbf{n}_1 \cdot \mathbf{n}_2, \quad \mathbf{n}_j = \frac{\mathbb{I}_j(N(\mathbf{p}_j)) - \overline{\mathbb{I}_j(N(\mathbf{p}_j))}}{||\mathbb{I}_j(N(\mathbf{p}_j)) - \overline{\mathbb{I}_j(N(\mathbf{p}_j))}||}, j = 1,2, \tag{5}$$

where $N(\mathbf{p}_j)$ is a neighborhood around $\mathbf{p}_j$ in image $\mathbb{I}_j$, and $\mathbb{I}_j(N(\mathbf{p}_j))$ is the vector of the image values in this neighborhood. This measure compares the intensity distributions inside the two neighborhoods. It is invariant to changes of the mean intensity value and of the dynamic range inside the neighborhoods.

Once we have a criterion to measure the image coherence, we can now recover the

3D geometry by maximizing the criterion for a given set of views. Two different types of approaches for this optimization have been proposed in the literature. In the first type, the texture similarity is computed using the current shape estimation. It permits explicitly computing visibility and using it in the texture similarity computation. If the measure is improved by deforming the model locally, then the model is updated and the process iterated as in [26,24,27]. Level-set based methods as in [21,22] explore a volumetric band around the current model but they still remain locally dependent on the current model shape. Since the exploration does not test all the possible configurations, the algorithm can fail because of local maxima of the texture coherence criterion. Besides, computing visibility with the current model does not always imply having a more accurate texture criterion. If the current model is far away from the real shape, the visibility can be wrong and so can the texture criterion. However, if the current model is close to the real shape, taking visibility into account can improve the final result. The second type of approaches refers to the use of a model-independent texture criterion to test all the possible configurations. In order to improve even more the robustness, we can accumulate the criterion values into a 3D grid by using a voting approach as in [17,34]. Since our initialization (visual hull) can be far away from the real surface, we believe that model-dependent criteria may have local minima problems. For this reason, we prefer to use a model-independent voting approach. Voting favors robustness in the presence of highlights and passing from the image information to a more usable information of the sort "probability of finding a surface" [35].

5.1 *Proposed Voting Approach*

In this section we first state the geometric relations between images. Then we present how this relation is used to optimize the texture coherence, and finally we detail our complete voting approach.

We first define the projective geometry relation between any 3D point $\mathbf{P}$ and its projections $\mathbf{p}_1$ and $\mathbf{p}_2$ in the two images $\mathbb{I}_1$ and $\mathbb{I}_2$(see Fig. 2). Let $\mathbf{P}(\delta)$ be the optic ray generated by the optical center $\mathbf{O}_1$ and the direction $\mathbf{d}$ defined by a pixel $\mathbf{p}_1$ of $\mathbb{I}_1$ as follows:

$$\mathbf{P}(\delta) = \mathbf{O}_1 + \delta\mathbf{d}, \;\; \delta \in [0,\infty). \tag{6}$$

Let $\mathscr{P}_2$ be the projection matrix of the second camera, the projection of the optic ray into $\mathbb{I}_2$ can be computed as:

$$\mathbf{p}_2(\delta) \sim \mathscr{P}_2\mathbf{P}(\delta). \tag{7}$$

Let $\mathbf{d}_\infty$ and $\mathbf{e}_{21}$ be the projections in $\mathbb{I}_2$ of the infinity point $\mathbf{P}(\delta \to \infty)$ and the origin $\mathbf{O}_1$ of the optic ray respectively:

$$\mathbf{e}_{21} \sim \mathscr{P}_2\mathbf{O}_1, \; \mathbf{d}_\infty \sim \mathscr{P}_2\mathbf{P}(\delta \to \infty). \tag{8}$$

We have then the following relations:

$$\mathbf{p}_2(\delta) \sim \mathbf{e}_{21} + \delta \mathbf{d}_\infty \sim \begin{bmatrix} e^x_{21} + \delta d^x_\infty \\ e^y_{21} + \delta d^y_\infty \\ e^z_{21} + \delta d^z_\infty \end{bmatrix} \sim \begin{bmatrix} \frac{e^x_{21}+\delta d^x_\infty}{e^z_{21}+\delta d^z_\infty} \\ \frac{e^y_{21}+\delta d^y_\infty}{e^z_{21}+\delta d^z_\infty} \\ 1 \end{bmatrix}. \tag{9}$$

For a given 3D depth δ, its relationship with the 2D distance between the epipole $\mathbf{e}_{21}$ and $\mathbf{p}_2(\delta)$ can be obtained as follows:

$$||\mathbf{p}_2(\delta) - \mathbf{e}_{21}|| = \sqrt{\left(\frac{e^x_{21} + \delta d^x_\infty}{e^z_{21} + \delta d^z_\infty} - \frac{e^x_{21}}{e^z_{21}}\right)^2 + \left(\frac{e^y_{21} + \delta d^y_\infty}{e^z_{21} + \delta d^z_\infty} - \frac{e^y_{21}}{e^z_{21}}\right)^2}, \tag{10}$$

and after simplification:

$$||\mathbf{p}_2(\delta) - \mathbf{e}_{21}|| = \frac{\delta}{e^z_{21} + \delta d^z_\infty} \sqrt{a^2 + b^2}, \tag{11}$$

where

$$\begin{aligned} a &= d^x_\infty - d^z_\infty e^x_{21} / e^z_{21}, \\ b &= d^y_\infty - d^z_\infty e^y_{21} / e^z_{21}. \end{aligned} \tag{12}$$

This simple formula allows the passage from the 2D pixel distance $||\mathbf{p}_2(\delta) - \mathbf{e}_{21}||$ to the 3D metric depth δ. It applies also for any other view $\mathbb{I}_j$ seeing the same optic ray, and links together all the corresponding 2D distances $||\mathbf{p}_j(\delta) - \mathbf{e}_{j1}||$.

Let us consider our problem of 3D recovery from texture. We want to optimize, for a given pixel in one image, the texture coherence with the other images. An optic ray can be defined by the pixel, and we search for the 3D point $\mathbf{P}$ belonging to the optic ray that maximizes the normalized cross-correlation with the other images. This can be done by sampling the projection of the optic ray in every image. In practice, the knowledge of the visual hull, which is an upper-bound of the object, allows us to accelerate computations. For a given pixel, the 3D depth to scan is fixed by the intersection of its optic ray with the visual hull. This intersection gives a depth interval in which we know that the object is contained. According to Eq. (11), this depth interval can be translated into pixel intervals for the correlation computation.

Equation (11) enables multi-correlation algorithms to stay in the pixel space for all the computations. For a same pixel, individual correlations are computed with the different images and merged into a unique reference system. Using several correlation curves for a same pixel allows us to make a more robust decision. The criterion used to find the best candidate depth δ along the optic ray is based on the correlation values but also on the coherence between the correlation curves. In our implementation, the correlation for a given pixel is computed with a fixed number of cameras as in [36]. For typical sequences of 36 images, a good compromise between computation time, accuracy and robustness is obtained by using the 4 nearest cameras ($\pm$ 10 deg and $\pm$ 20 deg). This configuration is also used in [17], and can resist up to 2 bad correlation curves due to highlights, occlusions or non-coherent textures occurring in the corresponding images.

A question arises concerning the extraction of the maximum of texture coherence from a sequence of images. A first simple answer is to estimate a 3D point on the surface for every pixel in every image:

```
pseudo code of the greedy algorithm
For each image in imageList
  For each pixel in image
    Compute the depth interval from the visual hull
    Compute the correlation curves
    Transform all the curves into the same coordinate system
    Find the best candidate depth
    If correlation value < threshold continue with next pixel
    Compute the 3D position P of the candidate depth
    Add the correlation value to the voxel grid containing P
```

The problem with this algorithm is the computation time. For large images (2000 x 3000), the computation time can reach 16 hours on a fast machine. This time can be strongly reduced with almost no loss because of the redundancy of the computation. The redundancy can be classified into two main groups: redundancy inside an image and redundancy between different images.

The redundancy between images is caused by the fact that several images see at the same time the same piece of surface. If we have already computed a surface estimation using one image, we can back project the 3D points into the next image, giving an initial estimation of the distance to the surface. The problem is that if the previous image did not correlate well, errors may propagate and prevent the following images from attenuating it.

The redundancy inside an image can be exploited using a previous knowledge of the content of the image. In our case, it is a picture of an object and we can expect it to be locally continuous. This implies that, if the surface is correctly seen and if there is no occlusion, the depth values for neighboring pixels should not be very different. This can be exploited in order to reduce the depth interval for the correlation criterion. In the greedy algorithm, for each pixel, we test the entire depth interval defined by the visual hull without taking into account if its neighbors have already found a coherent surface. To be able to benefit from already computed correlations, the image can be partitioned into different resolution layers as shown in Fig. 3.

The greedy algorithm is first run on the lowest resolution layer (black pixels in Fig. 3), with the depth intervals defined by the visual hull. For consecutive layers, the depth intervals are computed using the results of the precedent layer. To estimate the depth interval of a pixel based on the results of the previous layer, a record of the correlation values is maintained in order to control the reliability of the estimation. The theoretical maximum improvement that we can reach with this method in the case of 3 layers as illustrated in Fig. 3 is 16 times faster than the greedy method. This case corresponds to a black layer computation time much higher than the grey and white ones. In practice, the improvement is around 5 or 6 times faster for well-textured images. The worst case corresponds to non textured images where correlations become unreliable. The depth interval estimation fails, necessitating the use of the greedy method.

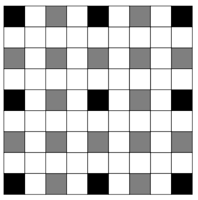
Fig. 3. Example of an image partition into 3 different resolution layers.

Besides the improvement in computation time, an easy improvement in storage space is to substitute the 3D volume grid by a more compact octree structure. The result of the correlation step will be a 3D octree containing the accumulated hits of all the pixel estimations. The new algorithm can be coded as:

```
pseudo code of the redundancy-based algorithm
For each image in imageList
  For each layer in image
    For each pixel in layer
      If layer = first layer
         Compute the depth interval from the visual hull
      Else
         Compute the depth interval from the previous layer
      Compute the correlation curves
      Transform all the curves into the same coordinate system
      Find the best candidate depth
      If correlation value < threshold continue with next pixel
      Compute the 3D position P of the candidate depth
      Add the correlation value to the octree voxel containing P
```

This volume by itself cannot be used as a force to drive the snake. A possible force could be the gradient of the correlation volume. The objection is that it is a very local force defined only in the vicinity of the object surface. The proposed solution to this problem is to use a gradient vector flow (GVF) field to drive the snake.

5.2 Octree-based Gradient Vector Flow

The GVF field was introduced by [37] as a way to overcome a difficult problem of traditional external forces: the capture range of the force. This problem is caused by the local definition of the force, and the absence of an information propagation mechanism. To eliminate this drawback, and for all the forces derived from the gradient of a scalar field, they proposed to generate a vector field force that propagates the gradient information. The GVF of a scalar field $f(x,y,z) : \mathbb{R}^3 \mapsto \mathbb{R}$ is defined as the vector field $\mathbf{F} = [u(x,y,z), v(x,y,z), w(x,y,z)] : \mathbb{R}^3 \mapsto \mathbb{R}^3$ that minimizes the following energy functional $\mathscr{E}_{GVF}$:

$$\mathscr{E}_{GVF} = \int \mu||\nabla \mathbf{F}||^2 + ||\mathbf{F} - \nabla f||^2 ||\nabla f||^2, \tag{13}$$

where μ is the weight of the regularization term and $\nabla \mathbf{F} = [\nabla u, \nabla v, \nabla w]$. The solution to this minimization problem has to satisfy the Euler equation:

$$\mu \nabla^2 \mathbf{F} - (\mathbf{F} - \nabla f)||\nabla f||^2 = 0, \tag{14}$$

where $\nabla^2 \mathbf{F} = [\nabla^2 u, \nabla^2 v, \nabla^2 w]$ and ∇^2 is the Laplacian operator. A numerical solution can be found by introducing a time variable t and solving the following differential equation:

$$\mathbf{F}_t = \mu \nabla^2 \mathbf{F} - (\mathbf{F} - \nabla f)||\nabla f||^2. \tag{15}$$

The GVF can be seen as the original gradient smoothed by the action of a Laplacian operator. This smoothing action allows eliminating strong variations of the gradient and, at the same time, propagating it. The degree of smoothing/propagation is controlled by μ. If μ is zero, the GVF will be the original gradient, if μ is very large, the GVF will be a constant field whose components are the mean of the gradient components.

Since our data have been stored in an octree structure, the GVF has to be computed on a multi-resolution grid. For this, we need to be able to:

- define the gradient operator and the Laplacian operator in the octree grid;
- define how to interpolate between voxels with different sizes.

In three dimensions, the gradient and Laplacian operators are defined as:

$$\nabla f = [f_x, f_y, f_z], \quad \nabla^2 f = f_{xx} + f_{yy} + f_{zz}. \tag{16}$$

In the case of a regular grid with a spacing of $[\Delta x, \Delta y, \Delta z]$, the first and second derivatives can be approached by central finite differences:

$$\begin{aligned} f_x &\approx \frac{f(x+\Delta x,y,z) - f(x-\Delta x,y,z)}{2\Delta x}, \\ f_{xx} &\approx \frac{f(x+\Delta x,y,z) - 2f(x,y,z) + f(x-\Delta x,y,z)}{\Delta x^2}. \end{aligned} \tag{17}$$

If the grid is not regular, then the finite differences will not be centered. An easy way to find the equivalent formulas for a non-regular grid is to estimate the parabolic curve ax^2+bx+c that passes through 3 points (Fig. 4), and compute the derivatives of the estimated curve [38]. After solving the equation system, we find:

$$
\begin{aligned}
f_x &\approx \frac{1}{(\delta+\Delta)}\left(\frac{f(x+\Delta)-f(x)}{\Delta/\delta}-\frac{f(x-\delta)-f(x)}{\delta/\Delta}\right)=2ax+b,\\
f_{xx} &\approx \frac{2}{(\delta+\Delta)}\left(\frac{f(x+\Delta)-f(x)}{\Delta}+\frac{f(x-\delta)-f(x)}{\delta}\right)=2a.
\end{aligned}
\tag{18}
$$

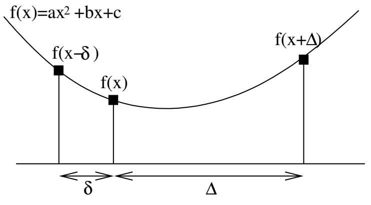

Fig. 4. Parabolic curve passing through 3 points.

As far as the interpolation is concerned, and to simplify the computation, we have to add a constraint to the topology of the multi-resolution grid: the difference of resolution in the neighborhood of a voxel, including the voxel itself, cannot be greater than one level. This is not a strong constraint since the resolution of the octree needs to change slowly if we want good numerical results in the computation of the GVF.

There exist three different scenarios in the multi-resolution numerical algorithm. The first one is when the current voxel and all its neighbors are the same size (see Fig. 5(a)). In this case, computations are done as with a mono-resolution grid. The second one is when the current voxel is bigger than or equal to its neighbors (see Fig. 5(b)). For those voxels with the same size, computations are carried out in an ordinary way. For those which are smaller, a mean value is simply used to get the correct value in the scale of the current voxel:

$$
f_x(A)\approx\frac{f(EFGH)-f(D)}{2\delta},\quad f_y(A)\approx\frac{f(B)-f(C)}{2\delta}. \tag{19}
$$

The third case corresponds to the current voxel being smaller than or equal to its neighbors (see Fig. 5(c) and (d)). We illustrate two different configurations, and in both we want to compute the gradient at the point A. In Fig. 5(c) we need the value of the function f at the points E, F, BC and CD:

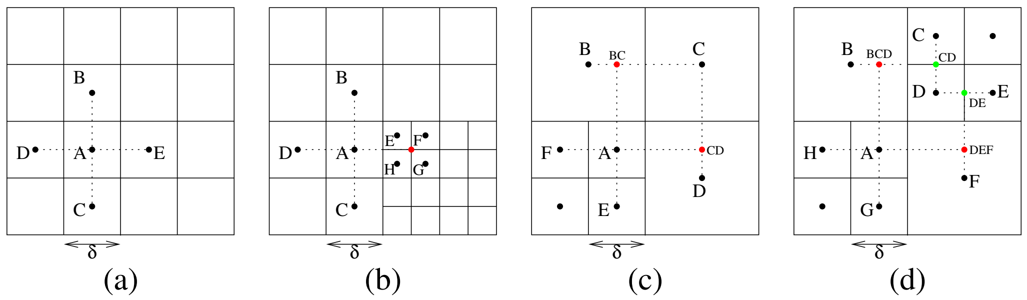

Fig. 5. Value interpolations (2D example): (a) Mono-grid case. (b) The current voxel is bigger that its neighbors. (c),(d) The current voxel is smaller than its neighbors.

$$\begin{aligned} f_x(A) &\approx \frac{1}{(\delta+1.5\delta)}\left(\frac{f(CD)-f(A)}{1.5}-\frac{f(F)-f(A)}{1/1.5}\right), \\ f_y(A) &\approx \frac{1}{(\delta+1.5\delta)}\left(\frac{f(BC)-f(A)}{1.5}-\frac{f(E)-f(A)}{1/1.5}\right). \end{aligned} \tag{20}$$

In the example shown in Fig. 5(d) the values BCD and DEF are obtained by interpolating B with CD, and DE with F, respectively. If we translate these examples into 3D, we have an additional interpolation along the new dimension for the points BC and CD in Fig. 5(c), and BCD and DEF in Fig. 5(d).

In Fig. 6 we compare the result of a 3D GVF computation for $\mu = 0.1$ using a regular grid and the octree approach. The scalar field f used in the example is defined as:

$$f(x,y,z) = \begin{cases} 1 \text{ for } z \in [34,36] \\ 0 \; else \end{cases} .$$

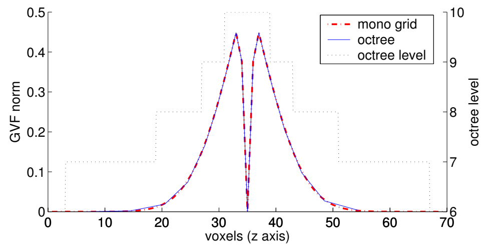

Fig. 6. mono grid vs. octree grid GVF.

We can appreciate the accuracy of the multi-grid computation compared to the mono-grid one. We can hardly see any difference between both curves, only when the octree resolution becomes very low (voxels 20 and 50). Mean values of computation speed up for 10 levels of resolution are between 2 and 3 times as fast as the mono-grid version while storage space is reduced between 10 and 15 times.

6 Silhouette Driven Force

The silhouette force is defined as a force that makes the snake match the original silhouettes of the sequence. If it is the only force of the snake, the model should converge towards the visual hull. Since we are only interested in respecting silhouettes, the force will depend on the self occlusion of the snake. If there is a part of the snake that already matches a particular silhouette, the rest of the snake is not concerned by that silhouette, since the silhouette is already matched. If we compare a visual hull and the real object, we see that the entire real object matches the silhouettes, but not all the points of the object. The object concavities do not obey any silhouette because they are occluded by a part of the object that already matches the silhouettes. The main problem is how to distinguish between points that have to obey the silhouettes (contour generators) and those that do not have to. To solve this problem, the silhouette force can be decomposed into two different components: a component that measures the silhouette fitting, and a component that measures how strongly the silhouette force should be applied. The first component is defined as a distance to the visual hull. For a 3D vertex $\mathbf{v}$ on the mesh of the snake, this component can be implemented by computing the smallest *signed* distance d_{VH} between the silhouette contours and the projection of the point into the corresponding silhouette:

$$d_{VH}(\mathbf{v}) = \min_i d(S_i, \mathscr{P}_i \mathbf{v}). \tag{21}$$

A positive distance means that the projection is inside the silhouette and a negative distance that the projection is outside the silhouette. Using only this force would make the snake converge towards the visual hull.

The second component measures the occlusion degree of a point of the snake for a given view point c. The view point is chosen as the camera that defines the distance to the visual hull:

$$\begin{aligned} \alpha(\mathbf{v}) &= \begin{cases} 1 & \text{for } d_{VH}(\mathbf{v}) \leq 0 \\ \frac{1}{(1+d(S_c^{snake}, \mathscr{P}_c \mathbf{v}))^n} & \text{for } d_{VH}(\mathbf{v}) > 0 \end{cases}, \\ c(\mathbf{v}) &= arg \min_i d(S_i, \mathscr{P}_i \mathbf{v}). \end{aligned} \tag{22}$$

In the definition of α, there are two cases. If d_{VH} is negative, it means that the point is outside the visual hull. In that case, the force is always the maximum force. For a point inside the visual hull, c is the camera that actually defines its distance to the visual hull d_{VH}. S_c^{snake} is the silhouette created by the projection of the snake into the camera c. The power n controls the decreasing ratio of α. This function gives the maximum silhouette force to the points that belong to the contour generators. All the other points, which are considered as concavities, are weighted inversely to their corresponding snake silhouette distance $d(S_c^{snake}, \mathscr{P}_c \mathbf{v})$. This allows the points of the snake to *detach* themselves from the visual hull. A big value of n allows an easier detachment. But in that case the force becomes too local and does not allow smooth transitions between concavities and contours. The value used in practice is $n = 2$, which is a compromise between smoothness and concavity recovery.

The final silhouette force for a given point of the snake is a vector directed along the normal to the snake surface $\mathbf{N}(\mathbf{v})$ and its magnitude is the product of both components:

$$\mathscr{F}_{sil}(\mathbf{v}) = \alpha(\mathbf{v}) d_{VH}(\mathbf{v}) \mathbf{N}(\mathbf{v}) \tag{23}$$

The silhouette force definition we give here is somewhat similar to the silhouette force defined in [27]. The main difference with our formulation is their binary definition of the $\alpha(\mathbf{v})$ function. They only apply the silhouette force to the contour generator vertices, i.e., vertices where $\alpha(\mathbf{v}) = 1$. This hinders smooth transitions between contour generators and concavities. Whereas in our case, we allow a smooth transition between concavities and contour generators (α values range from 0 to 1). The size of the transition region is controlled by the exponent n in Eq. (22). Smooth transitions allow us to better deal with incoherences between silhouettes and stereo.

7 Mesh Control

Having defined the texture and silhouette forces $\mathscr{F}_{tex}$ and $\mathscr{F}_{sil}$, i.e. the external forces, the last force to detail is the internal force $\mathscr{F}_{int}$. The goal of the internal force is to regularize the effect of the external forces. We define the internal regularization as the Laplacian operator, i.e., a force that tries to move a given mesh point $\mathbf{v}$ to the gravity center of its 1-ring neighborhood:

$$\mathscr{F}_{int}(\mathbf{v}) = \frac{1}{m} \sum_{j \in N_1(\mathbf{v})} \mathbf{v}_j - \mathbf{v}, \tag{24}$$

where $\mathbf{v}_j$ are the neighbors of $\mathbf{v}$ and m is the total number of these neighbors (valence). If the only force of the snake is the internal force, the mesh will collapse under the action of the barycentric filtering.

Since the texture force $\mathscr{F}_{tex}$ can sometimes be orthogonal to the surface of the snake, we do not use the force $\mathscr{F}_{tex}$ itself but its projection $\mathscr{F}_{tex}^N$ onto the surface normal:

$$\mathscr{F}_{tex}^N(\mathbf{v}) = (\mathscr{F}_{tex}(\mathbf{v}) \cdot \mathbf{N}(\mathbf{v}))\mathbf{N}(\mathbf{v}). \tag{25}$$

This avoids problems of coherence in the force of neighbor points and helps the internal force to keep a well-shaped surface.

The snake evolution process (Eq. (4)) at the k^{th} iteration can then be written as the evolution of all the points of the mesh $\mathbf{v}_i$:

$$\mathbf{v}_i^{k+1} = \mathbf{v}_i^k + \Delta t(\mathscr{F}_{tex}^N(\mathbf{v}_i^k) + \beta \mathscr{F}_{sil}(\mathbf{v}_i^k) + \gamma \mathscr{F}_{int}(\mathbf{v}_i^k)), \tag{26}$$

where Δt is the time step and β and γ are the weights of the silhouette force and the regularization term relative to the texture force. Equation (26) is iterated until convergence of all the points of the mesh is achieved. The time step Δt has to be chosen as a compromise between the stability of the process and the convergence time. An additional step of remeshing is done at the end of each iteration in order to maintain a minimum and a maximum distance between neighbor points of the mesh. This is obtained by a controlled decimation and refinement of the mesh. The decimation is based on the edge collapse operator and the refinement is based on the $\sqrt{3}$-subdivision scheme [39].

8 Texture Mapping

Once the 3D snake has converged, we can map the original set of color images into the 3D object surface to create a texture map. The method used to create the texture map is a particle-based approach such as the one described in [40,41]. This approach has been extended to filter the highlights that may be present in the images. For each vertex of the mesh, we compute the two cameras that best see it without occlusion and without highlights. For a given triangle $V_{i=0,1,2}$, we dispose of a total of three (possibly distinct) pairs of cameras that best see that triangle, one pair of cameras per vertex. Then the triangle is divided into a number of particles (see Fig. 7), and for each particle and for each pair of cameras we interpolate the particle color as in [40]. This gives us three colors per particle $rgb_{i=0,1,2}$. The final particle color is obtained by bilinear interpolation using the local parameterization (s,t) of the particle inside the triangle: $rgb = (1-s-t)\cdot rgb_0 + s\cdot rgb_1 + t\cdot rgb_2$. This allows us to filter highlights while preserving the continuity of the texture between adjacent triangles.

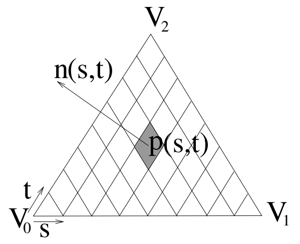

Fig. 7. Color particle.

9 Results

In this section we present a few results obtained with the proposed approach. All the reconstructions presented in this paper are obtained from a single axis rotation sequence of 36 images, using a fixed camera and a turntable where the object is posed. Intrinsic and extrinsic calibration parameters are recovered using the method described in [42]. The image resolution is 2008x3040 pixels for all the objects except for those of Fig. 18, where the image resolution is 4000x4000 pixels. The values of β and γ change very little from one object to another. Because the snake iteration is always done in the voxel coordinate system of the GVF octree, the value of β only depends on the ratio between the images size and the octree size. A typical value is between 0.1 and 0.2. Typical values of γ are between 0.1 and 0.25, depending on the required smoothness.

Computation times are dominated by the correlation voting step: a typical computation time for 36 images of 6 Mpixels is of 3 hours on a Pentium4 1.4GHz machine.

In Fig. 8 we present the complete reconstruction steps of the Twins model. We can appreciate the differences between the visual hull initialization (Fig. 8 left) and the final reconstruction (Fig. 8 middle). In Fig. 8 right we show the total force on each vertex after convergence. The jeopardized pattern observed on the force sign (red for positive sign, blue for negative sign) visually indicates convergence. Figure 9 illustrates the influence of the silhouette force. The support of the object does not provide any texture information and cannot be reconstructed using only the texture force (Fig. 9 left). Adding the silhouette constraint solves this problem and guarantees the convergence towards the visual hull in this region (Fig. 9 middle). As shown in Fig. 9 right, the support is completely driven by the silhouette force.

In Fig. 10 we illustrate the different forces used in the deformable model. Ten octree

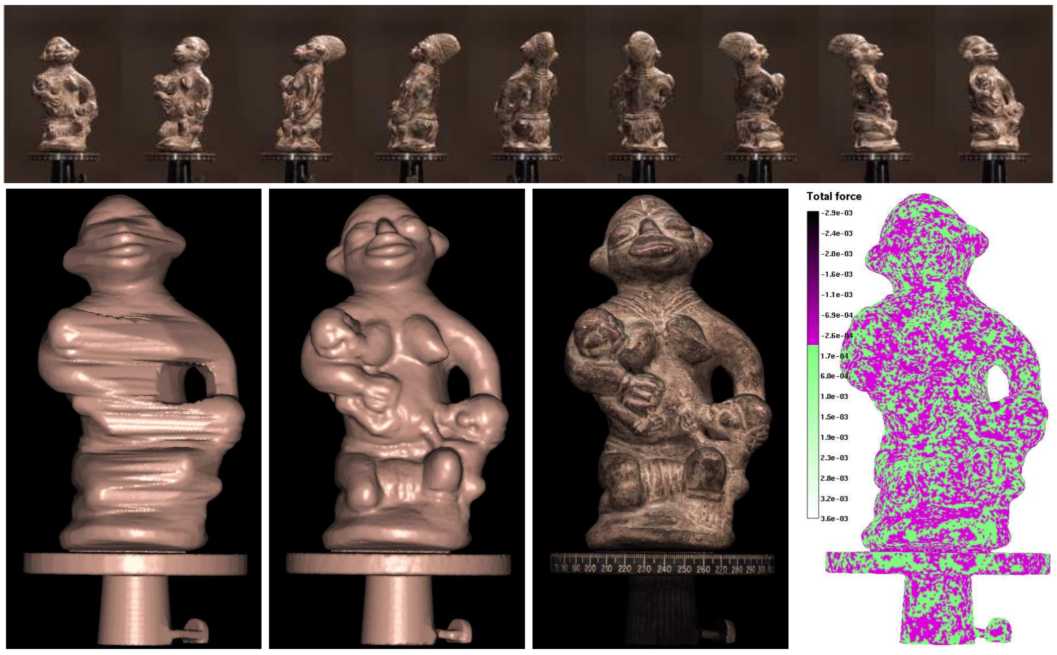

Fig. 8. Different steps in the reconstruction process of the Twins object. Top: some of the original images. Bottom: from left to right, visual hull initialization, final model, texture mapping and total force after convergence. The reconstructed model has 83241 vertices.

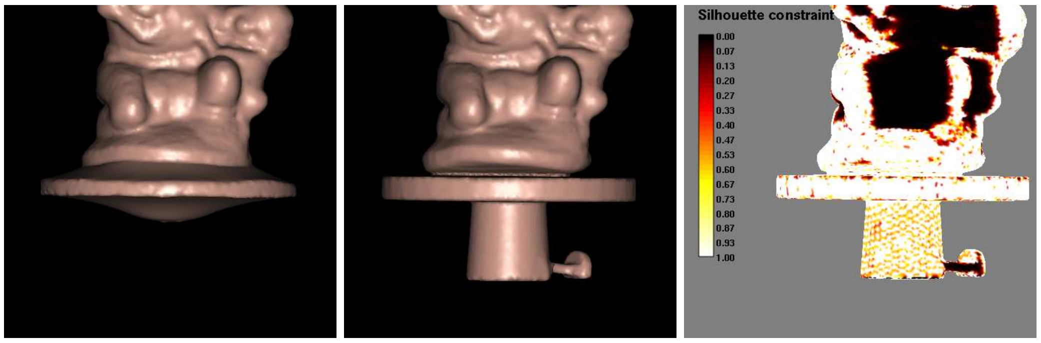

Fig. 9. Twins model detail after convergence. Left: evolution under texture force only. Parts of the object with no texture disappear. Middle: evolution under both the texture force and the silhouette force. The parts of the object with no texture follow the silhouette force (visual hull). Right: α component of the silhouette force after convergence.

levels are used in the voting approach (top left), which provides a high precision in the gradient vector flow computation (top middle and top right). At the end of the iterative process, a steady-state for the entire mesh is achieved, and concavities are automatically detected (bottom right).

A complete reconstruction is presented in Fig. 11 using both silhouette and texture information. We are able to recover many details with high accuracy (observe, for instance, the quality of reconstruction of the bracelet and of the rope).

In Fig. 12 we present a comparison between a laser acquisition method (top left) and the proposed method (top middle). We have less resolution in our reconstructed

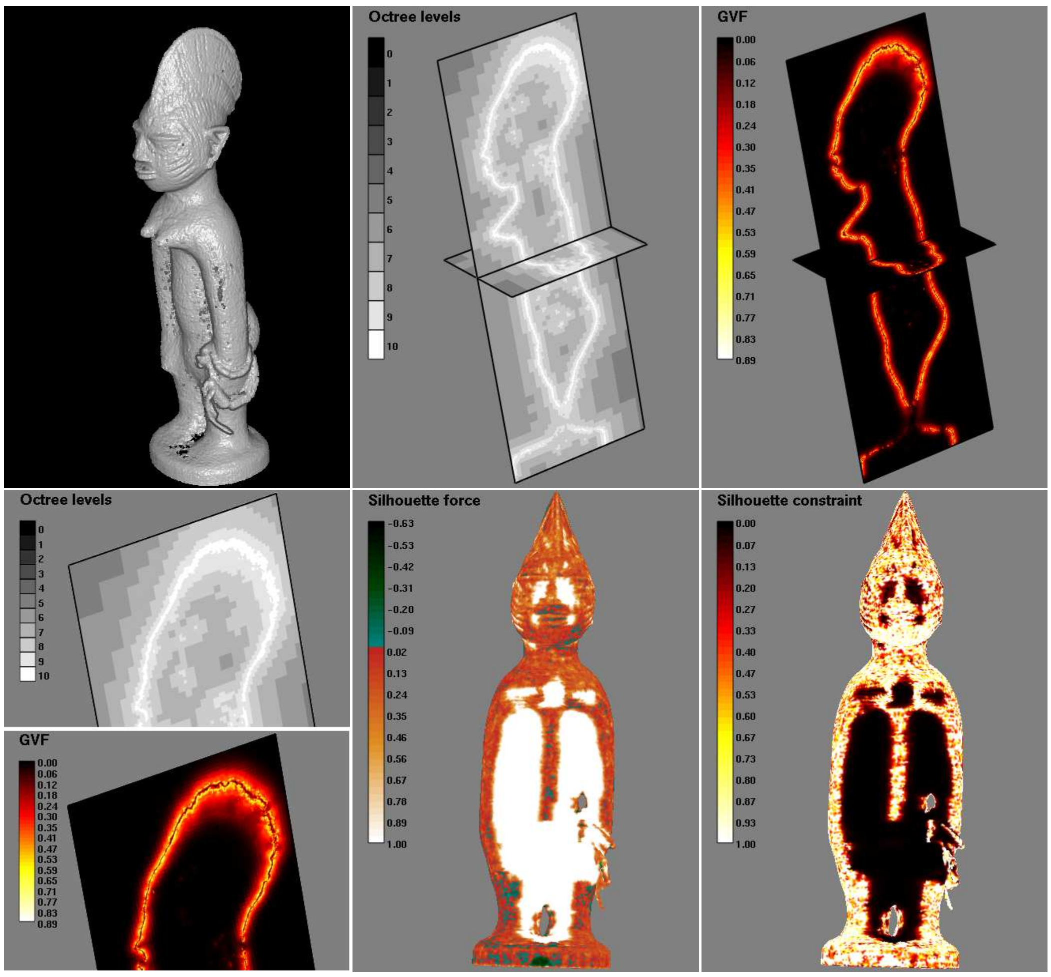

Fig. 10. External forces used in the reconstruction of the African model. Top left: volume rendering of the correlation voting volume. Top middle: the octree partition used in the computation of the gradient vector flow field. Top right: norm of the gradient vector flow field. Bottom left: detail of the octree partition and the gradient vector flow volume. Bottom middle: d_{VH} silhouette component after convergence. Bottom right: α component of the silhouette force after convergence.

model due to the stereo limitation and the regularization term. However, the mesh quality is quite good and the main object concavities are well recovered too. In addition, we provide a high quality texture map (top right).

In Fig. 13 and Fig. 14 we present two cases where the lack of good image correlations is handled differently by the proposed approach. In Fig. 13 left we can clearly distinguish the textured materials from the dark and bright non-textured material. The latter produces bad correlation results (Fig. 13 middle). The final model will converge to the textured surface whenever it exists and to the visual hull everywhere else (Fig. 13 right), producing some surface reconstruction errors in this area. In Fig. 14 the problem does not lie in the existence of texture but in the presence of highlights on the object surface. The difference with the previous case is that even if for a given camera the correlation does not work, the same surface is

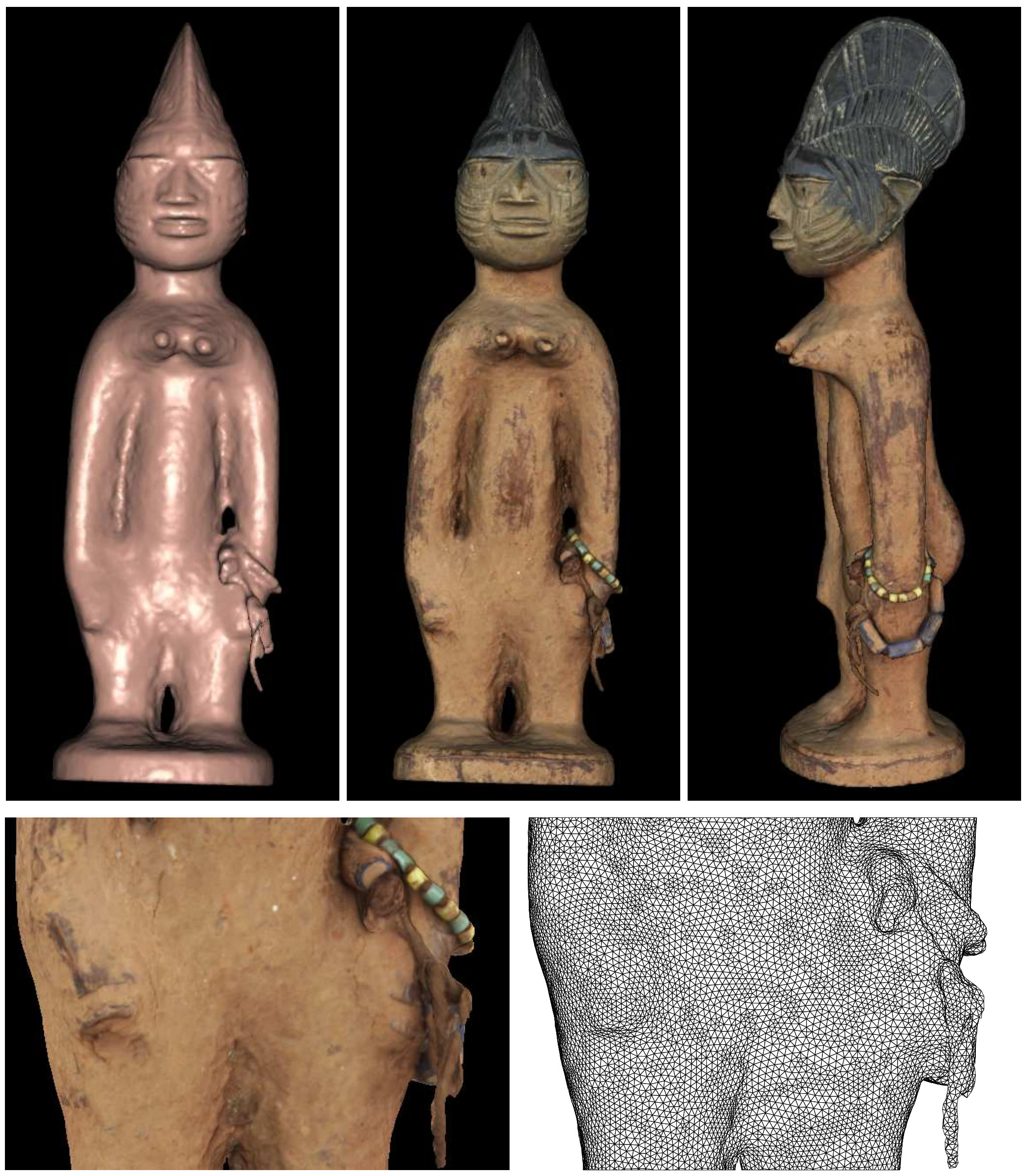

Fig. 11. African model after convergence (57639 vertices). Top left: Gouraud shading. Top middle: same view with texture mapping. Top right: lateral view of the textured model. Bottom left: detail of the textured model. Bottom right: same detail in wireframe.

seen without highlights by some other cameras. This allows the voting approach to overcome the local failure of the correlation criterion and to recover the right surface (see Fig. 8).

Other examples are shown in Fig. 15 and Fig. 16. We can appreciate the quality of the tunic folds reconstruction in the Hygia model (Fig. 15). We observe in Fig. 15 right how the final surface minimizes the GVF force. It shows that the method is powerful even for the reconstruction of small concavities.

In Fig. 17 we present a reconstruction of the same Twins object than in Fig. 8 but

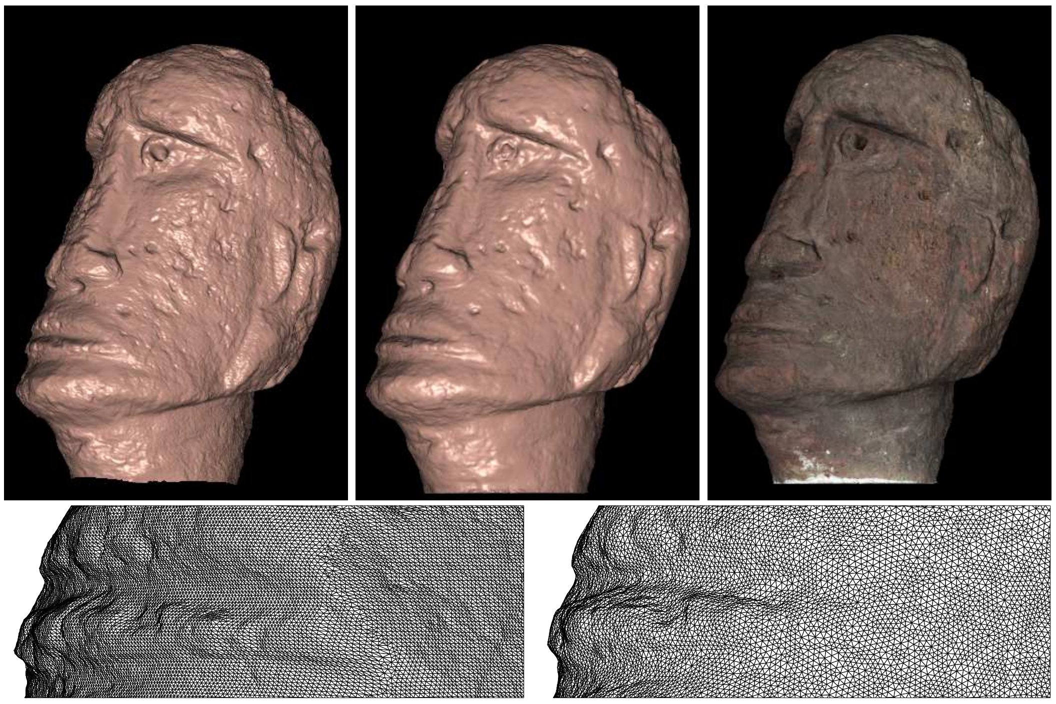
Fig. 12. Comparison between the proposed passive method and a laser active method. Top left: laser model of 385355 vertices obtained with a Minolta VIVID 910 3D scanner. Top middle: proposed method after snake convergence (233262 vertices). Top right: textured mesh after convergence. Bottom left: mouth detail of the laser model. Bottom right: same detail of the snake model.

using only 12 equally spaced images instead of 36. Correlation is computed using only the 2 nearest cameras (± 30 deg) and, although artifacts are visible due to the small number of silhouettes, the algorithm performs quite well computing the texture force, which offers a good 3D reconstruction.

In Fig. 18 we present the finest reconstructions we have currently achieved with the proposed method. We have used 36 images of 4000x4000 pixels and correlations have been stored using an octree of 11 levels. We can observe the great quality of the reconstructed models and the details obtained.

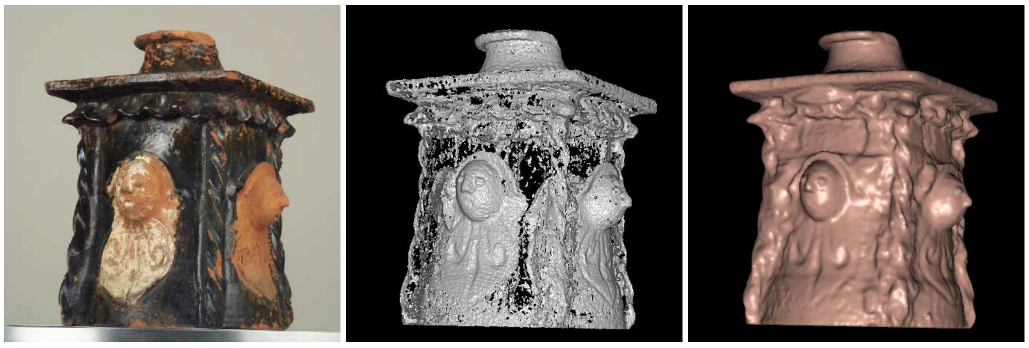
Fig. 13. Example of bad image correlations due to the dark and bright material. Left: one of the original images. Middle: volume rendering of the correlation voting volume. Right: snake mesh after convergence.

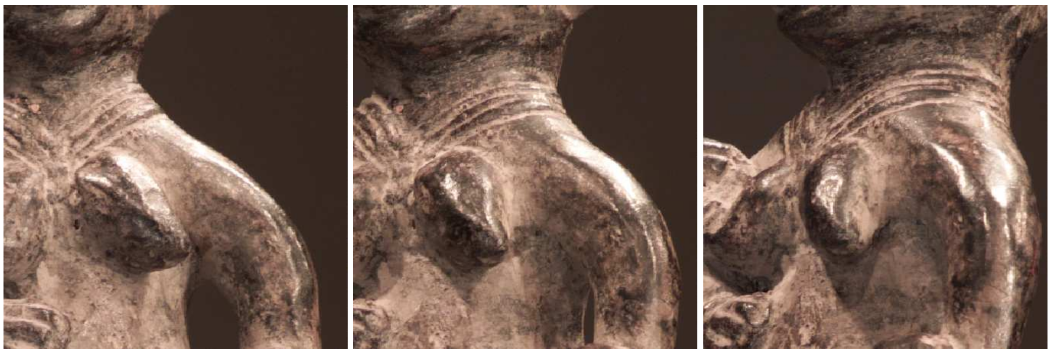

Fig. 14. Detail of the original Twins sequence. We can observe the existence of strong highlights on the surface. Since the highlights are not fixed from one image to another, the correlation voting approach allows recovering the right surface, as shown in Fig. 8.

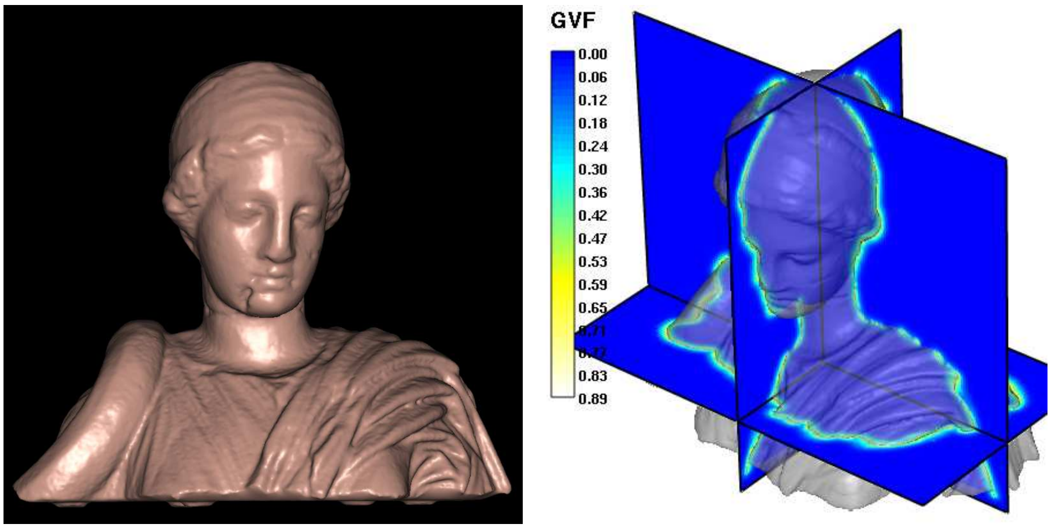

Fig. 15. Hygia model after convergence (159534 vertices). Left: Gouraud shading. Right: shaded model with transparency and 3 slices of the GVF octree volume.

10 Conclusion and future work

We have presented a new approach to 3D object reconstruction based on the fusion of texture and silhouette information. Our two main contributions are the definition and the fusion of the silhouette force into the snake framework, and the full system approach where different known techniques are used and improved in order to obtain high quality results.

The two main limitations of the algorithm are also its two main sources of robustness: the volume voting approach and the topology constant snake approach. The voting approach allows good reconstructions in the presence of highlights, but it also limits the maximum resolution of the 3D model. A way to overcome this limitation could be introducing the final model into another snake evolution where the texture energy computation would take into account the current shape (visibility and tangent plane or quadric based cross-correlation). Since the initial model is al-

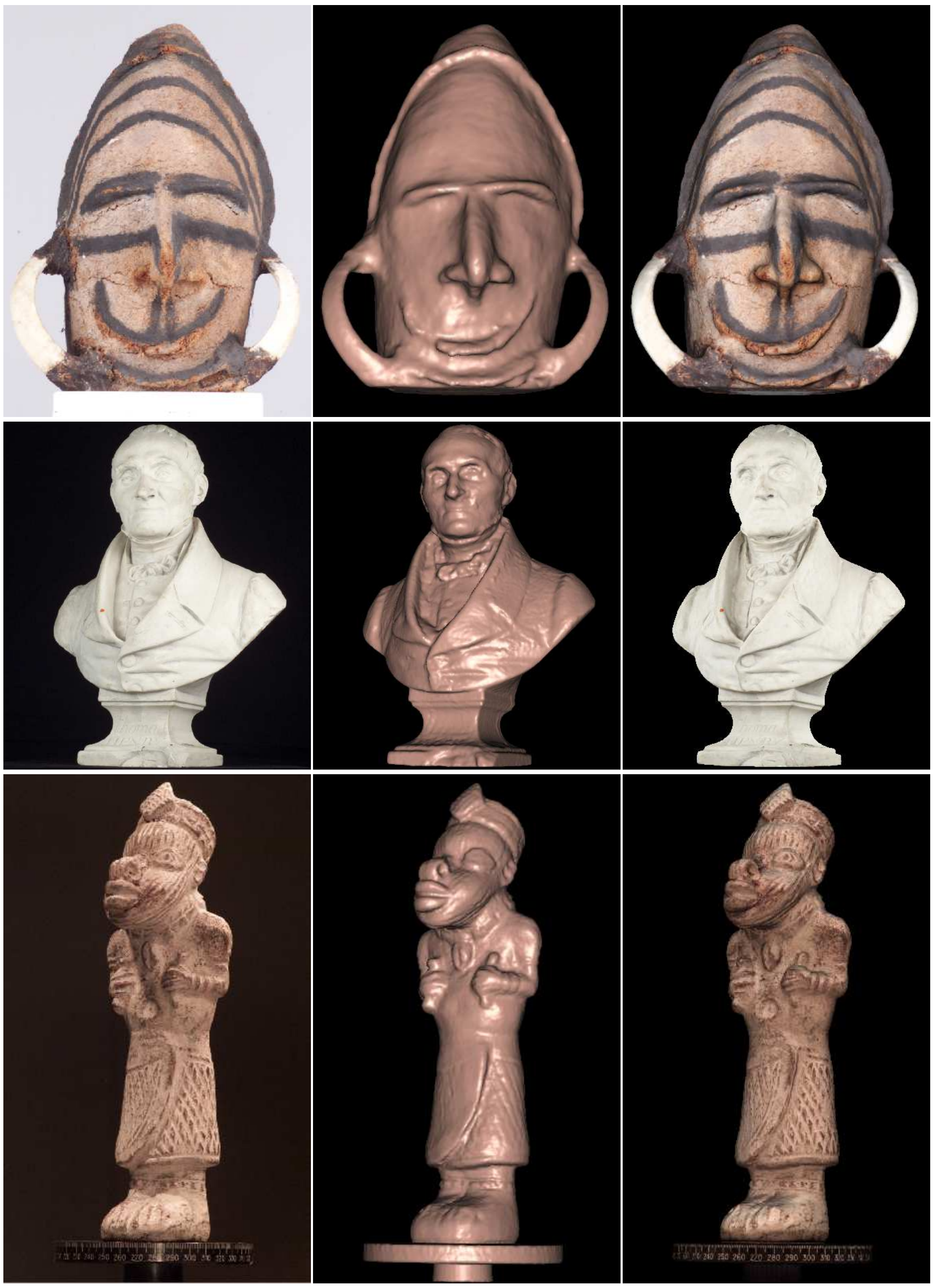

Fig. 16. Reconstructions using our proposed approach. Left: one original image used in the reconstruction. Middle: Gouraud shading reconstructed models (45843, 83628 and 114496 vertices respectively). Right: textured models.

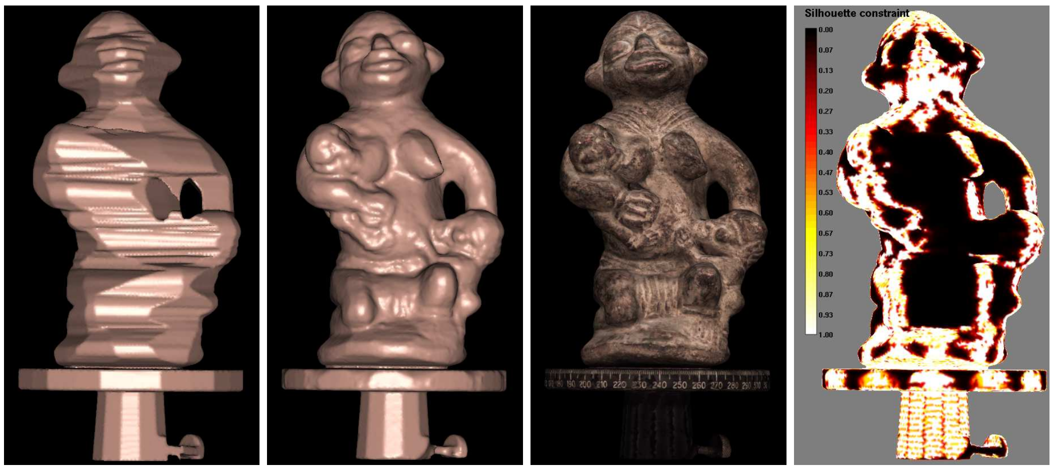

Fig. 17. Twins model reconstruction using only 12 equally spaced cameras. From left to right: visual hull initialization, final model, texture mapping and concavity recovery.

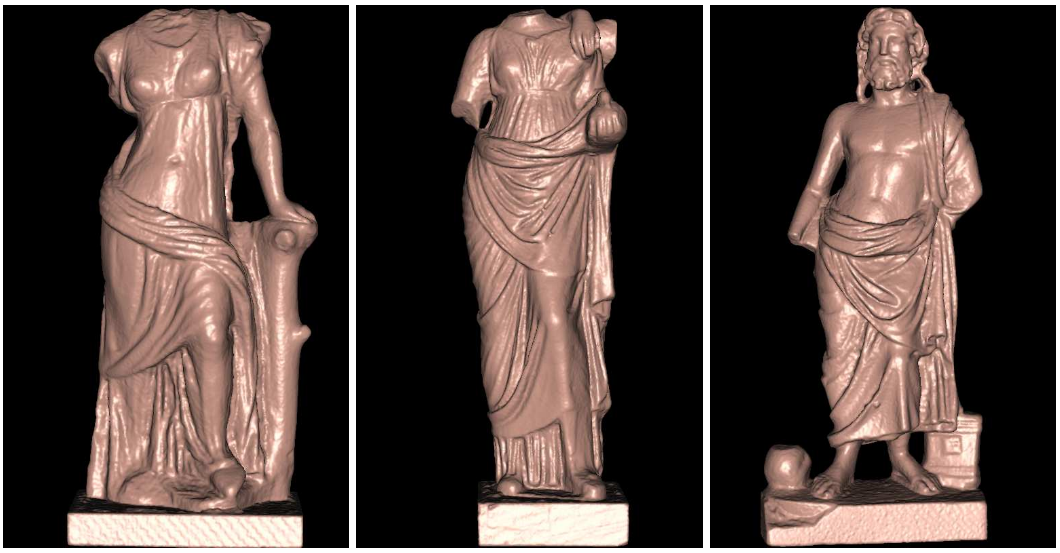

Fig. 18. Roman statues reconstructed from 36 images of 16 Mpixels. The octree used to store the correlation hits has 11 levels of depth. From left to right, models have respectively 223308, 211714 and 204220 vertices.

ready very close to the real surface, only some iterations would suffice to converge. The second drawback is the topology constant evolution. It allows a guaranteed topology of the final model but it is also a limitation for some kind of objects where the topology cannot be captured by the visual hull. A feasible solution would be to detect self collisions of the snake [43], and to launch a local level-set based method in order to recover the correct topology. Further work includes: i) the self calibration of the image sequence using both the silhouettes and traditional methods, ii) an improved strategy to detect local convergence of the snake in order to freeze optimized regions and to accelerate the evolution in the empty concavitity regions, iii) the possible use of the surface curvatures to allow a multi-resolution evolution of the mesh, iv) some more advanced work in the generation of the texture.

Acknowledgements

This work has been partly supported by the SCULPTEUR European project IST-2001-35372. We thank the Thomas Henry museum at Cherbourg for the image sequences corresponding to Figures 12, 13 and 16.

URL: www.tsi.enst.fr/3dmodels/

References

[1] S. Chen, L. Williams, View interpolation for image synthesis, in: SIGGRAPH '93, 1993, pp. 279–288.

[2] L. McMillan, G. Bishop, Plenoptic modeling: An image-based rendering system, in: SIGGRAPH '95, 1995, pp. 39–46.

[3] P. E. Debevec, C. J. Taylor, J. Malik, Modeling and rendering architecture from photographs: A hybrid geometry and image-based approach, in: SIGGRAPH '96, 1996, pp. 11–20.

[4] W. Matusik, C. Buehler, R. Raskar, S. Gortler, L. McMillan, Image-based visual hulls, SIGGRAPH 2000 (2000) 369–374.

[5] G. Slabaugh, R. Schafer, M. Hans, Image-based photo hulls, in: 3DPVT '02, 2002, pp. 704–862.

[6] M. Li, H. Schirmacher, M. Magnor, H. Seidel, Combining stereo and visual hull information for on-line reconstruction and rendering of dynamic scenes, in: Proceedings of IEEE 2002 Workshop on Multimedia and Signal Processing, 2002, pp. 9–12.

[7] F. Schmitt, B. Barsky, W. Du, An adaptative subdivision method for surface-fitting from sampled data, in: SIGGRAPH '86, 1986, pp. 179–188.

[8] B. Curless, M. Levoy, A volumetric method for building complex models from range images, in: SIGGRAPH '96, 1996, pp. 303–312.

[9] M. Levoy, K. Pulli, B. Curless, S. Rusinkiewicz, D. Koller, L. Pereira, M. Ginzton, S. Anderson, J. Davis, J. Ginsberg, J. Shade, D. Fulk, The digital michelangelo project: 3d scanning of large statues, in: SIGGRAPH 2000, 2000, pp. 131–144.

[10] G. Slabaugh, W. B. Culbertson, T. Malzbender, R. Shafer, A survey of methods for volumetric scene reconstruction from photographs, in: International Workshop on Volume Graphics 2001, 2001.

[11] B. G. Baumgart, Geometric modelling for computer vision, Ph.D. thesis, Standford University (1974).

[12] M. Potmesil, Generating octree models of 3d objects from their silhouettes in a sequence of images, CVGIP 40 (1987) 1–29.

[13] R. Vaillant, O. Faugeras, Using extremal boundaries for 3d object modelling, IEEE Trans. Pattern Analysis and Machine Intelligence 14 (2) (1992) 157–173.

[14] W. Niem, J. Wingbermuhle, Automatic reconstruction of 3d objects using a mobile monoscopic camera, in: Int. Conf. on Recent Advances in 3D Imaging and Modeling, 1997, pp. 173–181.

[15] Y. Matsumoto, H. Terasaki, K. Sugimoto, T. Arakawa, A portable three-dimensional digitizer, in: Int. Conf. on Recent Advances in 3D Imaging and Modeling, 1997, pp. 197–205.

[16] S. Sullivan, J. Ponce, Automatic model construction, pose estimation, and object recognition from photographs using triangular splines, IEEE Trans. Pattern Analysis and Machine Intelligence 20 (10) (1998) 1091–1096.

[17] Y. Matsumoto, K. Fujimura, T. Kitamura, Shape-from-silhouette/stereo and its application to 3-d digitizer, in: Proceedings of Discrete Geometry for Computing Imagery, 1999, pp. 177–190.

[18] S. Seitz, C. Dyer, Photorealistic scene reconstruction by voxel coloring, International Journal of Computer Vision 38 (3) (2000) 197–216.

[19] L. Zhang, S. M. Seitz, Image-based multiresolution shape recovery by surface deformation, in: Proc. of SPIE: Videometrics and Optical Methods for 3D Shape Measurement, 2001, pp. 51–61.

[20] A. Yezzi, G. Slabaugh, R. Cipolla, R. Schafer, A surface evolution approach of probabilistic space carving, in: 3DPVT '02, 2002, pp. 618–621.

[21] R. Keriven, O. Faugeras, Variational principles, surface evolution, pdes, level set methods, and the stereo problem, IEEE Transactions on Image Processing 7 (3) (1998) 336–344.

[22] A. Sarti, S. Tubaro, Image based multiresolution implicit object modeling, EURASIP Journal on Applied Signal Processing 2002 (10) (2002) 1053–1066.

[23] G. Cross, A. Zisserman, Surface reconstruction from multiple views using apparent contours and surface texture, in: A. Leonardis, F. Solina, R. Bajcsy (Eds.), NATO Advanced Research Workshop on Confluence of Computer Vision and Computer Graphics, Ljubljana, Slovenia, 2000, pp. 25–47.

[24] J. Isidoro, S. Sclaroff, Stochastic refinement of the visual hull to satisfy photometric and silhouette consistency constraints, in: Proc. ICCV, 2003, pp. 1335 –1342.

[25] S. Soatto, A. J. Yezzi, H. Jin, Tales of shape and radiance in multi-view stereo, in: Proc. ICCV, 2003, pp. 974 –981.

[26] P. Fua, Y. Leclerc, Object-centered surface reconstruction: Combining multi-image stereo and shading, International Journal of Computer Vision 16 (1995) 35–56.

[27] S. Nobuhara, T. Matsuyama, Dynamic 3d shape from multi-viewpoint images using deformable mesh models, in: Proc. of 3rd International Symposium on Image and Signal Processing and Analysis, 2003, pp. 192–197.

[28] M. Kass, A. Witkin, D. Terzopoulos, Snakes: Active contour models, International Journal of Computer Vision 1 (1988) 321–332.

[29] J. Sethian, Level Set Methods: Evolving Interfaces in Geometry, Fluid Mechanics, Computer Vision and Materials Sciences, Cambridge University Press, 1996.

[30] D. Adalsteinsson, J. Sethian, A fast level set method for propagating interfaces, Journal of Computational Physics 118 (1995) 269–277.

[31] X. Han, C. Xu, J. L. Prince, A topology preserving level set method for geometric deformable models, IEEE Transactions on PAMI 25 (2003) 755–768.

[32] A. Laurentini, The visual hull concept for silhouette based image understanding, IEEE Trans. on PAMI 16 (2) (1994) 150–162.

[33] W. E. Lorensen, H. E. Cline, Marching cubes: A high resolution 3d surface construction algorithm, in: Proceedings of SIGGRAPH '87, Vol. 21, 1987, pp. 163–169.

[34] G. Medioni, M.-S. Lee, C.-K. Tang, A Computational Framework for Segmentation and Grouping, Elsevier, 2000.

[35] A. Broadhurst, T. Drummond, R. Cipolla, A probabilistic framework for the Space Carving algorithm, in: Proc. 8th ICCV, IEEE Computer Society Press, Vancouver, Canada, 2001, pp. 388–393.

[36] C. Hernández, F. Schmitt, Multi-stereo 3d object reconstruction, in: 3DPVT '02, 2002, pp. 159–166.

[37] C. Xu, J. L. Prince, Snakes, shapes, and gradient vector flow, IEEE Transactions on Image Processing (1998) 359–369.

[38] B. Fornberg, Generation of finite difference formulas on arbitrarily spaced grids, Mathematics of Computation 51 (1988) 699–706.

[39] L. Kobbelt, $\sqrt{3}$-subdivision, in: SIGGRAPH 2000, 2000, pp. 103–112.

[40] F. Schmitt, Y. Yemez, 3d color object reconstruction from 2d image sequences, in: IEEE Interational Conference on Image Processing, Vol. 3, 1999, pp. 65–69.

[41] H. Lensch, W. Heidrich, H. P. Seidel, A silhouette-based algorithm for texture registration and stitching, Journal of Graphical Models (2001) 245–262.

[42] J. M. Lavest, M. Viala, M. Dhome, Do we really need an accurate calibration pattern to achieve a reliable camera calibration?, in: Proc. ECCV, Vol. 1, 1998, pp. 158–174, germany.

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