Filling Holes on Locally Smooth Surfaces Reconstructed from Point Clouds

Jianning Wang [a], Manuel M. Oliveira [b]

[a]*Center for Visual Computing and Computer Science Department - Stony Brook University*
Stony Brook, NY, 11794-4400, USA

[b]*Instituto de Informática - Universidade Federal do Rio Grande do Sul - UFRGS*
Av. Bento Gonçalves 9500, CEP 91501-970, Porto Alegre - RS, Brazil

Abstract

Creating models of real scenes is a complex task for which the use of traditional modeling techniques is inappropriate. For this task, laser rangefinders are frequently used to sample the scene from several viewpoints, with the resulting range images integrated into a final model. In practice, due to surface reflectance properties, occlusions and accessibility limitations, certain areas of the scenes are usually not sampled, leading to holes and introducing undesirable artifacts in the resulting models. We present an algorithm for filling holes on surfaces reconstructed from point clouds. The algorithm is based on moving least squares and can interpolate both geometry and shading information. The reconstruction process is mostly automatic and the sampling rate of the given samples is preserved in the reconstructed areas. We demonstrate the use of the algorithm on both real and synthetic datasets to obtain complete geometry and plausible shading.

Key words: Hole filling, surface reconstruction, moving least squares, 3D scanners.

1. Introduction

Creating accurate models of real objects and environments is a non-trivial task for which the use of traditional modeling techniques is inappropriate. In these situations, the use of laser rangefinders [6] seems attractive due to its relative independence of the sampled geometry and short acquisition time. The combined use of range and color images is very promising and has been demonstrated to produce an unprecedented degree of photorealism [22,25]. Unfortunately, surface properties (*e.g.*, low or specular reflectance), occlusions and accessibility limitations cause scanners to miss some surface areas, leading to incomplete reconstruction and introducing holes in the resulting models. This makes hole filling an important component of object and scene reconstruction. Its importance can be better appreciated when considering that, often times, it will not be possible to re-scan the original scene for acquiring extra samples. This may happen because the

Email addresses: `jianning@cs.sunysb.edu`, `oliveira@inf.ufrgs.br` (Manuel M. Oliveira).

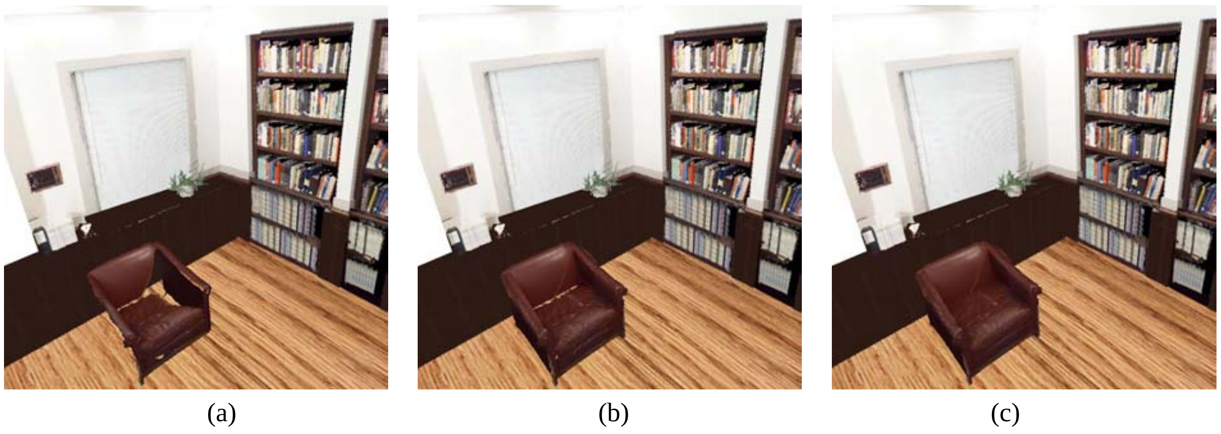

Fig. 1. Partial model of the UNC reading room, a real environment digitized with a 3D laser scanner. The floor texture has been replaced to emphasize the holes in the chair model. (a) Chair model reconstructed from the original samples only. Notice the big holes and missing outer surface on its left. (b) Model obtained from (a) using the symmetry-based techniques described in [40]. Despite the clear improvement, many small holes are still visible. (c) Complete model obtained applying our hole-filling algorithm to the model shown in (b).

scene might have changed or due to cost limitations. In these situations, one should try to obtain the best possible reconstruction using only the available samples. Creating high quality mesh representations for objects in the scene based on such incomplete information remains a challenge [45].

The problem of filling holes in range data can be divided into two sub-problems: (i) identifying the holes, and (ii) finding appropriate parameterizations that allow the reconstruction of the missing parts using the available data. Unfortunately, none of these problems are trivial because holes arising from the scanning of geometrically complex surfaces (*e.g.*, twisted, self-occluding surfaces) can be quite intricate [11]. However, in many situations of practical interest, holes occurring in range images present simple topologies. This is the case, for example, of many holes found when scanning interior environments and most objects. For these cases, the quality of the reconstructed 3D meshes can be significantly improved with the use of relatively simple algorithms.

This paper presents a revised and extended version of the work originally described in [41]. These extensions include an in-depth discussion comparing the proposed technique to state-of-the-art approaches used for hole filling [11,18,20,34], and several new surface-reconstruction examples with their associated statistics. Like [41], it discusses an approach for automatically identifying and filling holes in locally smooth regions of surfaces sampled as point clouds. The proposed algorithm can be applied to manifolds and surfaces with boundaries. In the later case, user assistance is required to avoid the inherent ambiguity of deciding which holes are artifacts of undersampled regions and which ones may define true surface boundaries. The technique does not provide a general solution to the hole-filling problem. In particular, it does not handle holes with arbitrary topologies or on highly twisted geometry. Nevertheless, it can be applied to a large range of practical situations, is conceptually very simple and its implementation is straightforward. Essentially, the algorithm takes a point cloud as input and produces an intermediate representation consisting of a triangle mesh, which is then analyzed for the existence of boundary edges (edges belonging to a single triangle). The occurrence of a hole implies the existence of a cycle defined by boundary edges. Once a boundary edge is found, the algorithm traces the entire boundary. A ring of points around the boundary, called the *boundary vicinity*, is then used to interpolate the hole using a *moving least squares* (MLS) procedure.

Our algorithm presents several desirable features:

– Hole filling is performed after meshing and, therefore, the algorithm can use any surface reconstruction technique that produces a triangle mesh;

– Since MLS interpolates the original samples, the algorithm guarantees that the reconstructed patches smoothly blend into the original mesh;

– It can be used to fill holes on both closed surfaces as well as on surfaces with boundaries;

– The reconstructed patches preserve the original sampling rate of their vicinities;

– As the new primitives are distinguished from the original points, they can be processed further;

– The processing is limited to the vicinities of holes.

We demonstrate the effectiveness of our approach on both real and synthetic datasets and show that it can significantly improve the overall quality of the models. We also show that for locally smooth surfaces, the described technique produces better results than the state-of-the-art hole-filling techniques [11,18]. In particular: (i) the reconstructed patches blend with the original model in a smoother way, (ii) it preserves the original mesh, and (iii) the resulting models contain a smaller number of vertices and triangles than the ones produced by other techniques [11,18] (see Section 6).

Figure 1(a) shows a partial model of the UNC reading room, an interior environment scanned with a laser rangefinder. Color, except for the floor texture, was obtained from photographs taken with a digital camera [22]. The clear floor texture was chosen in order to highlight the existence of holes in the chair model. Figure 1(b) shows the chair model after it has been processed using the techniques described in [40]. Despite the considerable improvement, many small holes are still visible in the chair. Figure 1(c) shows the same scene after our hole-filling algorithm has been applied to the chair model of Figure 1(b). Notice that the holes have been removed.

The remaining of the paper is organized as follows: Section 2 discusses some related work. Section 3 provides a brief review of the moving least squares method. The details of the hole-filling algorithm are described in Section 4. Section 5 presents some results obtained with the use of the proposed algorithm. A detailed discussion and comparison of our approach with other hole-filling techniques is presented in Section 6. Finally, Section 7 summarizes the paper and discusses some directions for future exploration.

2. Related Work

Hole filling is crucial for creating high quality mesh representations from scanned data. The chair model shown in Figure 7 was reconstructed from samples acquired during the scanning of real room-size environment. It is illustrative of a common situation: often times it is not possible or practical to cover the entire surface of an object even if several scanning positions are used. Moreover, the sampling density tends to vary across the scanned surface.

The problem of surface reconstruction from scattered data has been extensively studied in the past several years and many surface reconstruction algorithms have been proposed. These algorithms can be generally classified as *computational geometry, algebraic* and *implicit methods*. Computational geometry methods use mechanisms such as Delaunay triangulation [2,13] and region growing [4,7,15,23], and tend to leave holes in undersampled regions. Bernardini's ball-pivoting approach [4] can fill the resulting holes with successive executions of the algorithm using increasingly larger ball radii. The algorithm has no provision for selectively filling only certain holes and the maximum radius size needs to be specified.

Algebraic methods (*e.g.*, [35,36]) recover a surface by fitting a smooth function to the set of input points. Such methods produce hole-free models but cannot be used to reconstruct complex geometry or surfaces with boundaries.

The most popular approach for surface reconstruction is based on the use of implicit functions [8,10–12,16,24,26,32,37,43]. Implicit function methods, with the exception of [16], also produce hole-free models, but cannot be used to directly reconstruct surfaces with boundaries. In case the surface contains boundaries, these methods tend to produce incorrect results because implicit functions will, depending on the geometry at the boundary's neighborhood, tend to interpolate gaps or indefinitely extend the surface across its boundaries. Notice that, in practice, "artificial boundaries" may be introduced in the scanned dataset as a result of unsampled regions. This is, for example, the case of the armchair model shown in Figure 7(a), where the bottom of the chair and its leftmost part were not visible to the scanner. While the approach introduced by Hoppe et al. [16] can preserve boundaries, it assumes a constant sampling density across the entire surface and will leave holes in undersampled regions.

Curless's and Levoy's VRIP algorithm [10] uses an implicit method for surface reconstruction and hole filling. The approach consists of computing signed distance functions from a set of aligned meshes obtained from range scans. These functions are then blended and the final surface is extracted using the marching cubes algorithm [21]. In order to perform hole filling, the algorithm requires information about the line of sight of the scanner, and tends to perform poorly if the available information does not appropriately cover the entire volume enclosing the object. Like other implicit methods, it cannot be used to reconstruct surfaces with boundaries. According to Davis et al. [11], this method may reconstruct surfaces that look less plausible than a smooth interpolation of the observed surfaces.

Davis et al. [11] use a volumetric diffusion approach, analogous to inpainting techniques [5,27] to fill holes in range scans. The technique is targeted at the reconstruction of densely sampled closed surfaces. The process consists of converting a surface into a voxel-based representation with a clamped signed distance function. The diffusion algorithm consists of alternating steps of blurring and compositing, after which the final surface is extracted using marching cubes [21]. Like in our approach, Davis's et al. technique performs hole filling after surface reconstruction and the processing is constrained to the neighborhood of the holes. Unlike our approach, this algorithm is based on the use of an implicit function and can handle holes with more complex topology and twisted geometry. However, it cannot be applied to surfaces with boundaries and does not preserve the vertices of the original mesh. Moreover, the reconstructed patch is not guaranteed to smoothly blend with the rest of the surface, and, like in [10], may look little plausible.

Alexa et al. [1] use point sets to represent shapes and employ an approach similar to ours in the sense that they also locally project points onto planes and fit surfaces through those points. These fitted surfaces are used to down-sample or up-sample the original set of points, in order to guarantee that the resulting point rendering of the underlying surface has proper image space resolution. Their method, however, does not attempt to reconstruct a mesh representation or fill holes on surfaces.

Verdera et al. [38] use an approach similar to the one described by Davis et al. [11]. They turn a mesh into an implicit representation, then use a PDE system to inpaint the missing regions. Savchenko and Kojekine [33] deploy a space mapping approach to extend existing surface boundaries and fill the gaps using RBFs. Clarenz et al. [9] minimize the Willmore energy to ensure continuity of the normal field using PDEs.

Liepa [20] presented an algorithm targeted toward filling holes in oriented connected manifold meshes. It fills holes by first identifying hole boundaries, triangulating the holes, and finally smoothing the resulting patches. The triangulation is based on an $O(n^3)$ algorithm that produces a minimum area triangulation, thus requiring the resulting patches to be smoothed during the last stage of the algorithm.

More recently, Ju [18] introduced a volumetric algorithm that takes a polygonal mesh and creates a closed surface. It starts by scan converting the mesh into voxels, where holes are filled by patching boundary cycles using minimal surfaces. This technique is appropriate for filling very small holes or larger holes in locally flat surfaces. Like other volumetric approaches where surfaces are extracted using contouring techniques, the resulting mesh does not preserve the original vertices.

Sharf et al. [34] reconstruct fine details in missing regions on geometric complex surfaces represented by point clouds, which our approach cannot produce. It uses a coarse to fine approach based on an octree to copy and paste samples from a set of example regions (at the same level of detail) to the missing ones. The technique can produce some impressive results, but in the absence of appropriate examples, or for noisy or poorly sampled surfaces, the algorithm tends to produce poor results.

3. Moving Least Squares

Moving least squares (MLS) provides a class of complete solutions to the problem of fitting smooth functions to scattered data [19]. This is performed by interpolating the original data points, which may not be desirable in case the dataset is noisy. Our algorithm, however, uses MLS only to fill holes. Therefore, surface reconstruction can be performed with any reconstruction technique and may even include a low-pass filtering step to minimize the effects of noise. By using MLS to guide the hole filling process, our algorithm guarantees that the reconstructed patches smoothly blend into the reconstructed mesh. This is accomplished with the use of a relatively small number of samples in the vicinity of the hole boundaries. The entire algorithm is described in Section 4. The remaining of this section provides a quick review of MLS interpolation. For a more in-depth discussion of the subject, we refer the reader to [19].

Let s be a height function defined over a two-dimensional subspace ($s \models U \subset \Re^2 \rightarrow \Re$) and let $p_i \in U$ be a point in the domain of s. f_i is a height measurement associated with p_i. The fitness of s to a set of values f_1 to f_N can be measured by the error

$$E(s) = \sum_{i=1}^{N} w_i (s(p_i) - f_i)^2 \tag{1}$$

where N is the total number of points and w_i is the weight associated with point p_i. The best fit to the given set of values is obtained by minimizing the error $E(s)$. In practice, s is usually approximated by simple polynomial functions, such as the one shown in Equation 2 [19].

$$s(u,v) = a_0 + a_1 u + a_2 v + a_3 u^2 + a_4 v^2 + a_5 uv \tag{2}$$

In this case, the a_i coefficients that minimize the error are obtained by solving

$$a = \left(BWB^T\right)^{-1} BWf \tag{3}$$

where B is the matrix shown in Equation 4 and W is an n by n diagonal matrix with diagonal elements equal to w_i.

$$B = \begin{bmatrix} 1 & \cdots & 1 \\ u_1 & \cdots & u_n \\ v_1 & \cdots & v_n \\ u_1^2 & \cdots & u_n^2 \\ v_1^2 & \cdots & v_n^2 \\ u_1 v_1 & \cdots & u_n v_n \end{bmatrix} \tag{4}$$

Such a weighted least squares solution can only represent low order surfaces, often resulting in poor approximations. To reflect the fact that samples near the resampling positions should have more influence than far away samples, the error function should take into account weight factors w_i, which vary with the evaluation point:

$$E_p(s) = \sum_{i=1}^{N} w_i(p)(s(p_i) - f_i)^2 \tag{5}$$

For this case, a good choice of weight function is given by[19]:

$$w_i(p) = \frac{e^{-\alpha d_i^2(p)}}{d_i^2(p)} \tag{6}$$

Here, $d_i(p)$ is the distance from the new sampling position p to the ith original sample p_i (in the vicinity). When evaluated right at an input point, this weight function becomes infinity, thus interpolating the point. To avoid numerical problems, evaluation right at input points are handled individually. The parameter α controls the influence of vicinity features on the region to be resampled. As the weight functions depend on the resampling positions, new coefficients a_0 to a_5 for Equation 2 need to be computed for every resampled point as:

$$a(p) = \left(BW(p)B^T\right)^{-1} BW(p)f \tag{7}$$

where the elements of the diagonal matrix $W(p)$ are computed using Equation 6. Compared to the weighted least squares method, which creates a quadric surface for the entire hole, moving least squares compute one quadric surface for each evaluation point and blends them all. Therefore it can fit higher order surfaces.

4. The Hole Filling Algorithm

In order to fill holes, new points need to be added to the unsampled regions. To accomplish this, the algorithm first identifies hole boundaries and their vicinities. For each hole, it fits a plane through the vicinity points and, for each such a point, computes its distance to this plane as well as its projection onto the plane. The set of distances define a height field around the hole which is then used for surface fitting. This way, the problem of reconstructing holes in 3D is reduced to a simpler interpolation problem. Once a surface has been fitted to the height field using MLS, new points for filling the hole can be obtained by resampling the fitted surface. The basic version of the algorithm is presented in Algorithm 1, and its details are explained next.

4.1. *Finding Holes*

In order to identify holes, we start by creating a triangle mesh from the input point cloud. A number of algorithms exist for this purpose [2–4,13,15,16]. For the results shown in the paper, we have used the incremental surface reconstruction algorithm described in [15], which was chosen due to its simple implementation.

A hole consists of a loop of boundary edges. A *boundary edge* is defined as an edge belonging to a single triangle, as opposed to *shared edges*, which are shared by two triangles. By tracking boundary edges, holes can be identified automatically. Note, however, that there are two kinds of distinct boundaries: internal and external ones. An *internal* boundary delimits a hole on a surface. An *external* boundary, in turn, delimits either a patch ("island") inside a hole, or the limits of a surface, such as in the case of the end portions of the cylindrical surface shown in Figure 2. From the example of the cylinder, it becomes clear that not all holes should be necessarily filled and that user assistance is required in order to guarantee proper reconstruction.

Algorithm 1. The Hole-Filling Algorithm

```
1: Create a triangle mesh from the input point cloud;
2: repeat
3:     Automatically find a hole boundary and its vicinity;
4:     Compute a reference plane for the hole vicinity;
5:     Compute the distances of the vicinity points to the plane;
6:     Fit a surface through this height field using MLS;
7:     Fill the hole by resampling the fitted surface
8: until no holes exist
```

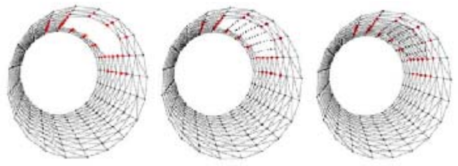
Fig. 2. Cylinder. (a) Triangle mesh with the hole and its boundary vicinity identified. (b) New points added (c) Reconstructed mesh.

4.2. *Computing the Reference Plane*

Once a hole has been identified, the next step is to use a ring of points around the boundary of the hole to provide a context for its interpolation. A height field is obtained by computing the distances from these points to a reference plane, which is the best fit plane to the set of points in the vicinity of the hole. The plane's position and orientation are computed as follows: first, the average $O = (\overline{x},\overline{y},\overline{z})$ of all vicinity points is computed as the origin of a new coordinate system associated to the plane. A matrix M is obtained by subtracting O from all points in the vicinity (Equation 8). Then, *Singular Value Decomposition* (SVD) [31] is used to compute the eigenvectors and eigenvalues of $M^T M$. The two eigenvectors with the largest absolute values span the reference plane and correspond to the U and V axes shown in Figure 3. The third eigenvector represents the plane normal (S axis).

$$M = \begin{bmatrix} x_1-\overline{x} & y_1-\overline{y} & z_1-\overline{z} \\ x_2-\overline{x} & y_2-\overline{y} & z_2-\overline{z} \\ . & . & . \\ x_{N-1}-\overline{x} & y_{N-1}-\overline{y} & z_{N-1}-\overline{z} \\ x_N-\overline{x} & y_N-\overline{y} & z_N-\overline{z} \end{bmatrix} \quad (8)$$

4.3. *Determining the Resampling Positions*

Each vicinity point is orthographically projected into the reference plane (UV plane), producing a pair of coordinates (u,v) and a height s computed as its distance to the reference plane. These values are used to fit the surface using MLS.

It is important that the set of points used to resample the hole have the same sampling density as the vicinity points. Two criteria are used for determining the resampling positions:

– The projections of new points should fall on the projection of the hole on the reference plane;

– The minimum distance from any new point to any other one (either new or vicinity point) should be bigger than a threshold.

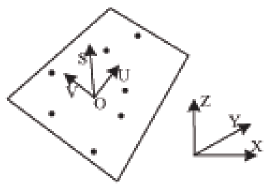

Fig. 3. The UV projection plane

While the first criterion seems self-evident, the second one is used to guarantee good remeshing results, since some reconstruction techniques require the input points to be spaced as evenly as possible [15,16]. The vicinity mesh is orthographically projected onto the UV plane, defining a mask. This situation is illustrated in Figure 4 for the case of a hole topologically equivalent to a disk. In case the hole contains "islands", they should also be projected and will be part of a disconnected mask. The mask image is traversed in scan line order using the step size compute using Equation 9. New sampling positions are then set over a regular grid inside the hole in the UV plane. If the distance between a point and the vicinity mask is less than $0.5 \times stepsize$, such a point is not used as a resampling point.

$$stepsize = \sqrt{\frac{area}{3n}} \tag{9}$$

Equation 9 provides a heuristic for spacing the resample positions inside a hole. n is the number of points on the boundary of the hole and *area* is the sum of the areas of all triangles connected with these points. The vicinity of the hole is then defined as the set of points whose distances from the boundary of the hole is less than $\beta \times stepsize$. The "thickness" of the vicinity ring is controlled by the parameter β.

4.4. *Fitting the Surface*

For each new point created to fill holes, a solution of Equation 7 provides the coefficients of Equation 2 necessary for determining $s(u,v)$. After that, the transformation from the UVS coordinate system to the XYZ coordinate system is straightforward. Similarly, the colors (R,G,B channels) associated to the points in the vicinity of a hole are treated as three separate height functions, from which the colors of the new points are resampled. This is achieved by replacing the height value of each vicinity point with the value of one of its associated color channels at a time. The resulting three height functions representing the three color channels are reconstructed using MLS and are resampled using a procedure similar to the one used to recover the missing geometric information. After the new points have been introduced, the final step is to remesh the complete model. Figure 4 illustrates the intermediate steps of the algorithm for the simple case of a planar surface.

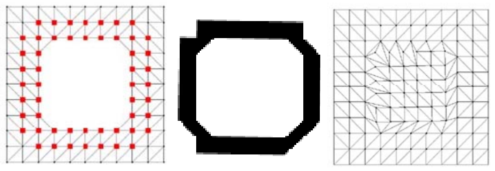

Fig. 4. Intermediate results of the algorithm. (left) A triangle mesh with the projection of the vicinity points highlighted. (center) Mask image for the projection of the vicinity points onto the reference plane. (right) Reconstructed mesh.

5. Results

We have implemented the described algorithm and used it to fill holes in models of both real and synthetic objects. In all cases, the input to the algorithm consisted of unorganized point clouds. For real objects, we used datasets acquired with laser scanners. The point clouds for synthetic objects were obtained by rendering 3D models and saving the contents of the corresponding color and depth buffers. Such information was later used to reproject colored points in 3D. We used the surface reconstruction algorithm described in [15] to create the initial triangle meshes and to remesh the model after new points were added. For the examples shown in the paper, we used $\alpha = \frac{1}{16}$ (Equation 6). The experiments were performed on a 2.0 GHz Pentium 4 PC with 512 MB of memory. Table 1 presents some statistics and the running times for applying our algorithm to the examples shown in the paper.

In order to illustrate the effectiveness of our algorithm, we used the techniques described in [40] to segment and reconstruct a chair model from range images acquired from a real environment (the UNC reading room). We then applied the proposed algorithm to the resulting model. Figure 7(a) shows the original samples of the chair rendered as a triangle mesh. Notice the existence of a big hole. Figure 7(b) shows the reconstruction of the same chair model after the use of the symmetry-based techniques described in [40]. Many small holes are still left due to the nonexistence of data in either side of the symmetry plane and to the inability of the surface reconstruction algorithm used [15] to work in areas containing local variations of sampling density. Figure 7(c) shows the resulting chair model after applying our hole filling algorithm to the model shown in (b). Notice that the holes have been eliminated. The original incomplete chair model has 41,511 points and 82,065 triangles. Those numbers change to 64,159 points and 126,588 triangles after reconstruction using symmetry information [40]. The complete chair model (geometry and color), reconstructed with the hole-filling algorithm, has 65,114 vertices and 136,002 triangles, and was obtained in 10.93 seconds.

Figure 1 shows the three stages of the reconstruction of the chair model in a real environment. The remaining of the scene (walls, shelves, etc.) was also reconstructed using the pipeline presented in [40]. The original reading room scene was edited by replacing the floor texture with a clear one and by repositioning the chair in order to emphasize the existence of holes in the model (Figures 1(a) and (b)).

Figure 8(a) shows the Stanford bunny, which is known for containing a few holes in its bottom. The dataset consists of 35,947 points and 1,246 of these points were used as vicinity points for filling all the holes. Figure 8(b) shows the resulting model after hole filling. It contains 36,478 points and was obtained in 6.43 seconds. The previous two examples illustrate that our algorithm can be applied to both surfaces with boundaries as well as closed surfaces.

The bust and angel models shown in Figures 9 and Figure 10, respectively, are synthetic models used illustrate the steps of the algorithm. In Figure 9(a), one sees the triangle mesh reconstructed from the point cloud. The highlighted points represent the vicinity of the hole, used as input for the MLS interpolation. Figure 9(b) shows the points resampled from the interpolated surface. The final model is shown in Figure 9(c). The larger reconstruction time for this model is as explained by the bigger number of resampled points (3,309), computed as the number of final point minus the number of original points (see Table 1).

Figure 10(a) shows an angel model with a hole in one its wings. The vicinity points are highlighted to show the identified hole. Figure 10(b) shows the points resampled from the patch that fills the hole. Figure 10(c) presents the reconstructed model, while Figure 10(d) shows the original model for comparison. Notice that although not the same as the original surface, the reconstructed patch is a plausible one.

Figure 11 illustrates the local nature of our algorithm. Figure 11(b) shows the Happy Buddha model after a few patches (highlighted in Figure 11(a)) have been removed, partially taking away some surface details. The locations where the patches were removed from were chosen to cover both low and high-frequency surface areas. The surfaces visible through the holes correspond to the back of the statue. The resulting model, after the five patches have been removed, contains 143,298 samples . The total number of vicinity

Table 1
Statistics and running time for different datasets

Data Set	# Orig. Points	# Vici. Points	# Final Points	# Orig. Triangles	# Final Triangles	MLS time (seconds)
Armchair	64,159	1,873	65,114	126,588	136,002	10.93
Bunny	35,947	1,246	36,478	69,451	70,733	6.43
Bust	12,853	1,506	16,162	21,757	29,357	72.85
Angel	10,189	347	11,264	18,737	20,921	4.16
Buddha	143,298	1,154	144,701	290,936	293,707	10.32

points for the five holes is 1,154 samples and the number of resampled (new) points is 1,075. The time required to perform hole filling was 10.32 seconds. Figure 11(c) shows the result produced by our hole filling algorithm. Notice that such a reconstruction is quite plausible. Figure 11(d) displays the original model for comparison.

The cost of searching for boundary edges is linear in the number e of edges of the mesh. Once such an edge is found, tracing the boundary requires, in the worst case, following $e-1$ edges. Since each boundary edge has exactly two adjacent boundary edges (which can be identified by their shared vertices), the cost of automatically finding hole boundaries is $O(e)$.

The cost of the MLS fitting depends on the number k of new points to be added and on the size of vicinity, m, thus $O(mk)$. Therefore, the cost of the hole-filling algorithm is $O(e+mk)$. In practice, however, we observe that the running time of the algorithm is mostly influenced by the size of the vicinity and by the number of resampled points used to fill the holes, as can be seen in Table 1 (for instance, compare the running times for the Bust and Buddha models).

If a point cloud is provided as input, the cost of creating a mesh from the point cloud, $O(n\log n)$ on the number of points [15], needs to be added to the total cost of the algorithm.

6. Discussion and Comparisons

The approaches presented by Davis et al. [11] and by Ju [18] use contouring algorithms to reconstruct polygonal models from intermediate voxel-based representations. Thus, unlike in our approach, the resulting models do not preserve the original meshes. Ju's approach [18] fills holes by reconstructing minimal surfaces, which tend to blend poorly with the original meshes as the sizes of the holes increase. Essentially, hole-filling techniques based on minimal surfaces only perform well on very small holes or when the surface is locally flat. Liepa's technique [20] uses an $O(n^3)$ algorithm for filling holes with minimal-surface patches. These are later processed to produce smoother surfaces. The approach presented by Sharf et al. [34] is intended for filling holes on highly-complex geometric regions and, therefore, is target toward a different domain than ours. Table 2 summarizes the major features of all these algorithms, including input type, properties of the interpolated patches, and cost.

Figure 5 shows two smooth surfaces with boundaries obtained after removing caps of different sizes from the surface of a sphere. The surface on the left contains 3,001 vertices and 5,940 triangles, while the one on the right contains 2,401 vertices and 4,740 triangles. These surfaces were used to compare how smoothly the patches reconstructed by the different techniques blend with the original meshes. A sphere was chosen as a reference shape because of its smoothness and symmetry, and because one has a clear idea about what to expect from an exact reconstruction.

We reconstructed the surfaces shown in Figure 5 using the techniques by Davis et al. [11], Ju [18], and our MLS approach. Davis's et al. and Ju's techniques were chosen because they are representatives of the state-of-the-art in hole filling and their source codes are available on the web [30,39]. Table 3 shows the parameters used by the different algorithms. In order to use Davis's et al. code, the triangle meshes were first converted to the PLY format [28] and then converted to their actual input format using the Ply2Vri program [29]. The resolution of the voxel space was defined experimentally by setting the voxel size to 0.16 in the Ply2Vri program (73 $\approx$ 10.0/0.16 + 10 fringe voxels). The third dimension of the voxel space was chosen so that the resulting surface could fit inside the volume. For Ju's program [30], the volume was subdivided in 64^3 voxels (level 6 of the octree).

Figure 6 shows the reconstructed results produced by the three algorithms for the surfaces shown in Figure 5. Figures 6 (a), (c) and (e) were reconstructed from the surface shown in Figure 5 (left), whereas Figures 6 (b), (d) and (f) are reconstructions of the surface shown in Figure 5 (right). The results shown in the first row of Figure 6 were produced by the algorithm of Davis et at [11]. Notice the existence of some protrusions on the reconstructed patches, which become more evident as the size of the hole increases. The abrupt cut on the largest highlight visible in Figure 6(b) indicates that: (i) the reconstructed patch does not smoothly blend into the original mesh and (ii) the original mesh was not preserved. The results produced by Ju's technique [18] are shown in Figures 6(c) and (d). Notice the minimal surfaces reconstructed by the technique, showing that it can only be used for filling small holes.

Figures 6(e) and (f) show the results produced by the MLS technique using 180 samples in the vicinity of each hole. The results are quite smooth, with the reconstructed patches naturally blending themselves into the original meshes. In Figure 6(e), the sphere was very satisfactorily recovered. In the case of the bigger hole, the algorithm reconstructed an egg-like shape, which is a plausible solution for the ill-posed problem of surface interpolation. The extended highlight indicates the smooth blending of the reconstructed patch with the original mesh. Table 4 shows the numbers of vertices and triangles produced by the three algorithms. The percentage values were computed with respect to the corresponding numbers in the input models. The meshes obtained with the algorithm by Davis et al. [11] present the largest numbers of primitives, followed by the meshes produced by Ju's algorithm [18]. This is due to the fact that these meshes are extracted using contouring techniques [17,21]. As a result, the number of resulting polygons depends on the resolution of the voxel space used, and not on the number of original vertices. Notice the significant increase on the numbers of vertices and triangles in the models produced by these techniques, even though a relatively low-resolution voxel space has been used (see Table 3) and the space is mostly empty. In these techniques, the entire model, and not only the interpolating patch, is extracted at voxel-space resolution.

An important feature of our algorithm is that the shape of the resulting patches can be reasonably predicted by the vicinities of the holes. For example, as we project the vertices belonging to the vicinity of the hole shown in Figure 5 (right) onto the reference plane (the cutting plane for the missing cap), the distances between any two pairs of projected (U,V) coordinates will be smaller than the difference between their corresponding heights (the S coordinates). As a result, $abs(\frac{\partial S}{\partial U}) >$ 1.0 and $abs(\frac{\partial S}{\partial V}) > 1.0$, producing the resulting egg-like shape.

Table 2
Comparison among four hole-filling algorithms. n is the number of points in the point cloud, e is the number of edges in the polygonal model, and v is the number of voxels used for spatial subdivision.

Algorithm	Input	Properties	Cost
MLS	points	smooth transitions	$O(n \log n)$
MLS	mesh	smooth transitions	$O(e+mk)$
Davis et al.	partial meshes	non-smooth transitions	$O(v)$
Ju	polygonal mesh	minimal surfaces	$O(v)$
Liepa	oriented connected manifold meshes	minimal surfaces later smoothed	$O(e^3)$
Sharf et al.	points	surface features copied from vicinity	$O(v)$

Table 3
Parameter used to reconstruct the models shown in Figure 5. The original sphere has radius equal to 5.0 units and the side of a voxel was set to 0.16 units.

Algorithm / Model	Figure 5(left)	Figure 5(right)
Davis et al.	$73 \times 73 \times 104$ volume	$73 \times 73 \times 135$ volume
Ju	$64 \times 64 \times 64$ volume	$64 \times 64 \times 64$ volume
MLS	180 points	180 points

Our algorithm might fail if the vicinity region presents folds or twists, which will not define a one-to-one mapping when projected onto the reference plane. In this case, that part of the hole might not be filled properly. When the size of a hole increases, the possibility of facing such a situation tends to grow. This problem could be addressed by, instead of a plane, using a curved domain for projecting the vicinity points onto.

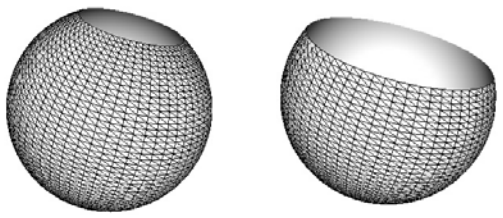

Fig. 5. Spheres with removed caps used to compare the how smoothly the reconstructed patches produced by different algorithms integrate themselves with the existing mesh. The surface on the left contains 3,001 vertices and 5,940 triangles, while the one on the right contains 2,401 vertices, 4,740 triangles.

Table 4
Number of vertices and triangles in the models produced by each technique. The percentages are computed with respect to the corresponding values in the input models.

Algorithm / Model	Figure 5(left)	%	Figure 5(right)	%
Input	3,001 vert	100.0	2,401 vert	100.0
	5,940 tris	100.0	4,740 tris	100.0
Davis et al.	10,323 vert	343.9	10,454 vert	435.4
	20,904 tris	351.9	20,904 tris	441.0
Ju	6,858 vert	228.5	5,978 vert	248.9
	13,712 tris	230.8	11,952 tris	252.1
MLS	3,542 vert	118.0	3,672 vert	152.9
	7,080 tris	119.2	7,098 tris	149.7

7. Conclusions and Future Work

Reflectance properties, occlusions and accessibility limitations can cause scanners to miss some surfaces, lending to incomplete reconstruction of scenes and undesirable holes in the resulting models. Due to the costs and difficulties associated with scanning real environments, the existence of automatic or semi-automatic tools for helping users to improve the quality of the acquired models is very desirable.

We have presented a simple and efficient algorithm for automatically identifying and filling holes on locally smooth surfaces represented as sets of unorganized points. Our approach can fill holes in both manifold

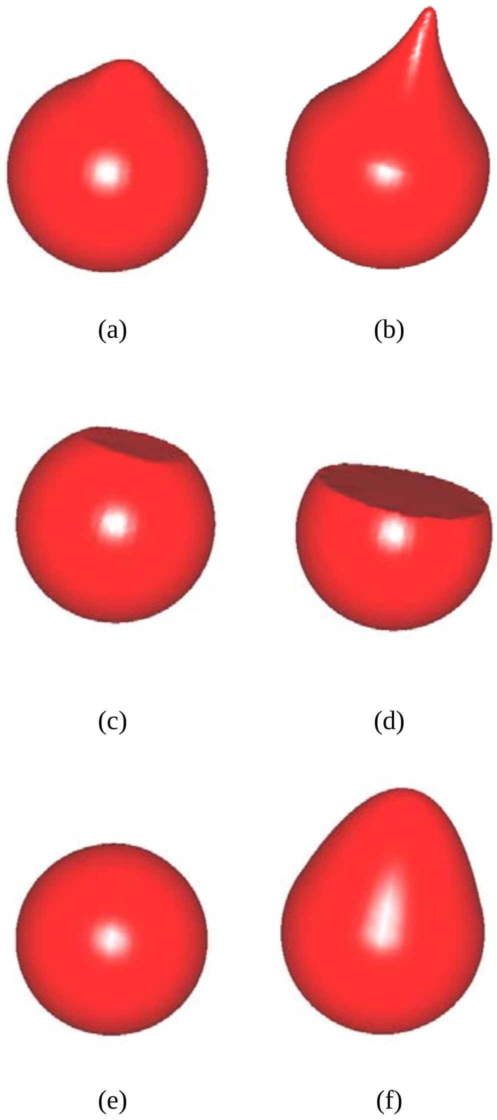

Fig. 6. Reconstruction of the surfaces shown in Figure 5 using different algorithms. First row: results produced by the algorithm of Davis et al. [11]. Middle row: results produced by the algorithm of Ju [18]. Bottom row: results produced by our MLS algorithm.

and surfaces with boundaries. In the case of surfaces with boundaries, user assistance is required to resolve the inherent ambiguity associated with surface reconstruction. The algorithm consists of finding boundary edges and tracing them to identify holes. Once a hole has been identified, its vicinity is used to interpolate the missing portion of the surface using moving least squares.

Like any interpolation approach, the proposed algorithm works well if the areas inside and around the hole are locally smooth. Similarly, although the color interpolation scheme works well for smooth shading, it cannot be used to reconstruct arbitrary textures. In this case, the use of texture synthesis such as the one described in [14,42,44] is more appropriate.

In order to obtain a proper parameterization of the hole vicinity, the projected boundary should not contain twists and folds in the direction of the projection onto the plane. We are currently looking into more general parameterizations as well as new approaches to deal with holes of arbitrary shapes and topologies.

Acknowledgments

The partial model of the UNC reading room was kindly provided by the UNC IBR Group. The bunny and Happy Buddha models are a courtesy of the Stanford 3D scanning repository. We would like to thank Tao Ju and J. Davis for making their codes available, and the anonymous referees for their insightful comments.

References

[1] M. Alexa, J. Behr, D. Cohen-Or, S. Fleishman, D. Levin, C. Silva. Point Set Surfaces. *Proc. IEEE Visualization'01*, pp. 21-28, 2001.

[2] N. Amenta, M. Bern, M. Kamvysselis. A New Voronoi-based Surface Reconstruction Algorithm. *Proc. SIGGRAPH'98*, pp. 415-421, 1998.

[3] C. Bajaj, F. Bernardini, G. Xu. Automatic Reconstruction of Surfaces and Scalar Fields from 3D Scans. *Proc. SIGGRAPH'95*, pp. 109-118, 1995.

[4] F. Bernardini, J. Mittleman, H. Rushmeier, C. Silva, G. Taublin. The Ball-Pivoting Algorithm for Surface Reconstruction. *IEEE Trans. on Visualization and Computer Graphics*. **5**(4), pp. 349-359, 1999.

[5] M. Bertalmio, G. Shapiro, V. Caselles, C. Ballester. Image Inpainting. *Proc. SIGGRAPH'00*, pp. 417-424, 2000.

[6] P. Besl. Advances in Machine Vision. *Advances in Machine Vision*, Springer Verlag, Chapter 1 - Active Optical Range Sensors. pp. 1-63. 1989.

[7] J.-D. Boissonnat. Geometric Structures for Three-Dimensional Shape Representation. *ACM Transactions on Graphics*, **3**(4), pp. 266-286, 1984.

[8] J.C. Carr, R. Beatson, J. Cherrie, T. Mitchel, W. Fright, B. McCallum, T. Evans. Reconstruction and Representation of 3D Objects with Radial Basis Functions. *Proc. SIGGRAPH'01*, pp. 67-76, 2001.

[9] U. Clarenz, U. Diewald, G. Dziuk, M. Rumpf, R. Dusu. A Finite Element Method for Surface Restoration With Smooth Boundary Conditions. *Computer Aided Geometric Design*, **21**(5), pp. 427-445, 2004.

[10] B. Curless, M. Levoy. A Volumetric Method for Building Complex Models from Range Images. *Proc. SIGGRAPH'96*, pp. 303-312, 1996.

[11] J. Davis, S. Marschner, M. Garr, M. Levoy. Filling Holes in Complex Surfaces using Volumetric Diffusion. *Proc. First International Symposium on 3D Data Processing, Visualization, Transmission*, pp. 428-438, 2002.

[12] H. Dinh, G. Turk, G. Slabaugh. Reconstructing Surfaces by Volumetric Regularization using Radial Basis Functions. *IEEE Trans. on Pattern Analysis and Machine Intelligence*, **24**(10), pp. 1358-1371, 2002.

[13] H. Edelsbrunner, E.P. Muche. Three-dimensional alpha shapes. *ACM Trans. on Graphics*, **13**, pp. 43-72, 1994.

[14] A. Efros, W. Freeman. Image Quilting for Texture Synthesis and Transfer. *Proc. SIGGRAPH'01*, pp. 341-348, 2001.

[15] M. Gopi, S. Krishnan. A Fast and Efficient Projection Based Approach for Surface Reconstruction. *International Journal of High Performance Computer Graphics, Multimedia and Visualisation* **1**(1), pp. 1-12, 2000.

[16] H. Hoppe, T. DeRose, T. Duchamp, J.A. McDonald, W. Stuetzle. Surface Reconstruction from Unorganized Points. *Proc. SIGGRAPH'92*, pp. 71-78, 1992.

[17] T. Ju, F. Losasso, S. Schaefer, J. Warren. Dual Contouring on Hermite Data. *Proc. SIGGRAPH'02*, pp. 339-346, 2002.

[18] T. Ju. Robust Repair of Polygonal Models. *Proc. SIGGRAPH'04*, pp. 888-895, 2004.

[19] P. Lancaster and K. Salkauskas. Curve and Surface Fitting, An Introduction. *Academic Press*, 1986.

[20] P. Liepa. Filling Holes in Meshes. *Symposium on Geometry Processing*, pp. 200-205, 2003.

[21] W. Lorensen, H. Cline. Marching cubes: A High Resolution 3D Surface Construction Algorithm. *Proc. SIGGRAPH'87*, pp. 163-169, 1987.

[22] D. McAllister, L. Nyland, V. Popescu, A. Lastra, C. McCue. Real Time Rendering of Real World Environments. *Proc. Rendering Techniques'99*, pp. 145-160, 1999.

[23] R. Mencl. A Graph-based Approach to Surface Reconstruction. *Proc. EUROGRAPHICS'95*, **14**(3), pp. 445-456, 1995.

[24] B. Morse, T. Yoo, P. Rheingans, D. Chen, K. Subramanian. Interpolating Implicit Surfaces from Scattered Surface Data Using Compactly Supported Radial Basis Functions. *Shape Modeling International*, pp. 89-98, 2001.

[25] L. Nyland, M. Cutts, H. Fuchs, D. McAllister, V. Popescu, C. McCue, A. Lastra, P. Rademacher, M. Oliveira, G. Bishop, G. Meenakshisundaram. The Impact of Dense Range Data on Computer Graphics. *Proc. Multi-View Modeling and Analysis Workshop*, pp. 3-10, 1999.

[26] Y. Ohtake, A. Belyaev, M. Alexa, G. Turk, H. Seidel. Multi-level Partition of Unity Implicits. *Proc. SIGGRAPH'03*, pp. 463-470, 2003.

[27] M. Oliveira, B. Bowen, R. McKenna, Y. Chang. Fast Digital Image Inpainting. *International Conference on Visualization, Imaging and Image Processing (VIIP 2001)*, pp.261-2665, 2001.

[28] Ply. http://www.cc.gatech.edu/projects/large_models/ply.html.

[29] Ply2Vri. http://grail.cs.washington.edu/software-data/ply2vri/.

[30] PolyMender. http://www.cs.rice.edu/∼jutao/code/polymender.htm.

[31] W. Press, S. Teukolsky, W. Vetterling, B. Flannery. Numerical Recipes in C: The Art of Scientific Computing. 2nd Ed., *Cambridge University Press*, 1992.

[32] V. Savchenko, A. Pasko, O. Okunev, T. Kunii. Function Representation of Solids Reconstructed from Scattered Surface Points and Contours. *Computer Graphics Forum*, **14**(4), pp. 181-188, 1995.

[33] V. Savchenko, N. Kojekine. An Approach to Blend Surfaces. *Proc. CGI'02*, pp. 139-150, 2002.

[34] A. Sharf, M. Alexa, D. Cohen-Or. Context-based Surface Completion. *Proc. SIGGRAPH'04*, pp. 878-887, 2004.

[35] S. Scarloff, A. Pentland. Generalized Implicit Functions for Computer Graphics. *Proc. SIGGRAPH'91*, pp. 247-250, 1991.

[36] D. Terzopoulos, D. Metaxas. Dynamic 3D Models with Local and Global Deformations: Deformable Superquadrics. *IEEE Trans. on Pattern Analysis and Machine Intelligence*, **13**(7), pp. 703-714, 1991.

[37] G. Turk, J. O'Brien. Variational Implicit Surfaces. Technical Report GIT-GVU-99-15, Georgia Institute of Technology, 1999.

[38] J. Verdera, V. Caselles, M. Bertalmio, G. Sapiro. Inpainting Surface Holes. *Proc. ICIP'03*, pp. 903-906, 2003.

[39] Volfill: a Hole Filler Based on Volumetric Diffusion. http://graphics.stanford.edu/software/volfill/.

[40] J. Wang, M. Oliveira. Improved Scene Reconstruction from Range Images. *Proc. EUROGRAPHICS'02*, pp. 521-530, 2002.

[41] J. Wang, M. Oliveira. A Hole-Filling Strategy for Reconstruction of Smooth Surfaces in Range Images. *Proc. the XVI Brazalian Symposium on Computer Graphics and Image Processing*, pp. 11-18, 2003.

[42] L.Y. Wei, M. Levoy. Texture Synthesis over Arbitrary Manifold Surfaces. *Proc. SIGGRAPH'01*, pp. 355-360, 2001.

[43] H. Xie, J. Wang, J. Hua, H. Qin, A. Kaufman. Piecewise C_1 Continuous Surface Reconstruction of Noisy Point Clouds via Local Implicit Quadric Regression. *Proc. IEEE Visualization 2003*, pp. 91-98, 2003.

[44] L. Ying, A. Hertzmann, H. Biermann, D. Zorin. Texture and Shape Synthesis on Surfaces. *Eurographics Rending Workshop*, pp. 301-312, 2001.

[45] Y. Yu, A. Ferencz, J. Malik. Extracting Objects from Range and Radiance Images. *IEEE Transactions on Visualization and Computer Graphics*, **7**(4), pp. 351-364, 2001.

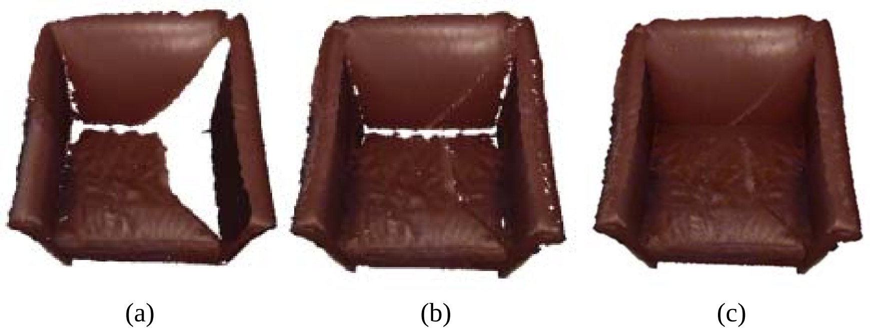

Fig. 7. UNC reading room armchair. (a) Model reconstructed as a triangle mesh using only the original samples. Notice the large missing areas.(b) Model obtained after processing the samples in (a) with the symmetry-based techniques described in [40]. (c) Final model without holes obtained by applying our hole filling algorithm to the model shown in (b).

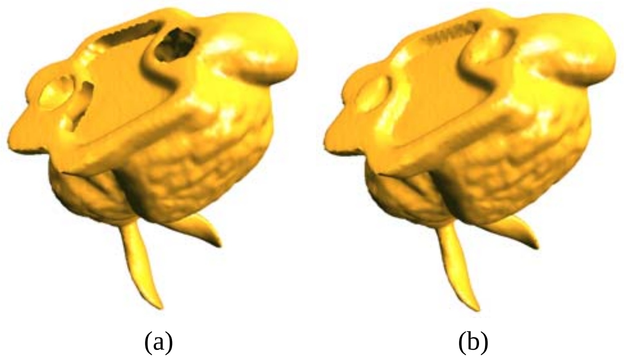

Fig. 8. The Stanford bunny. (a) The original model contains some holes. (b) Bunny model after hole filling performed with our algorithm.

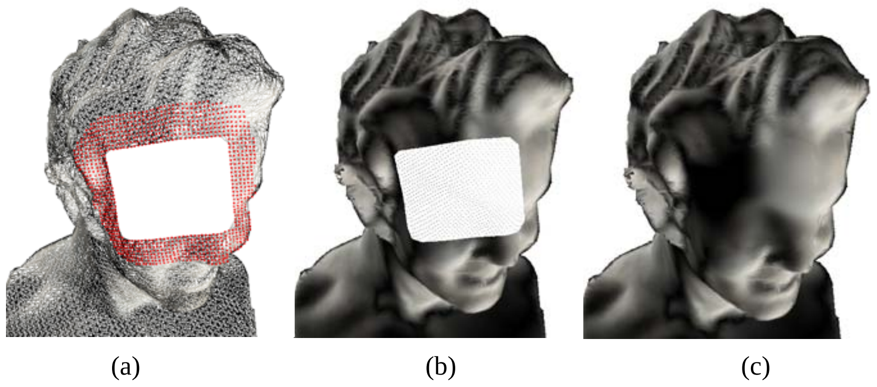

Fig. 9. Bust with a hole in the head used to illustrate the steps of the algorithm. (a) Triangle mesh with the hole and vicinity identified. (b) Points resampled from the interpolated patch. (c) Final model after hole filling.

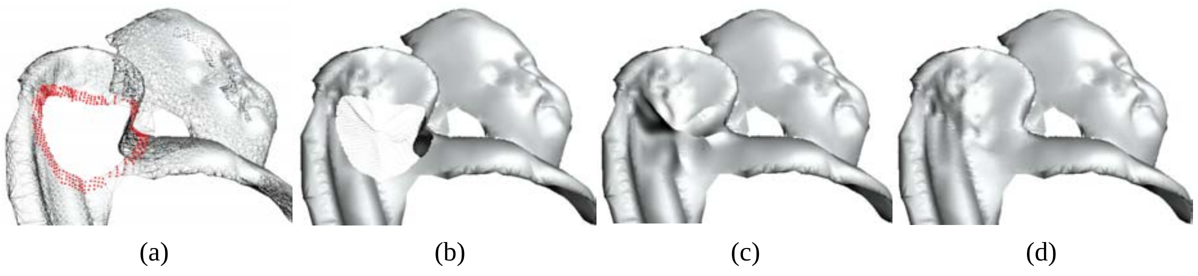

Fig. 10. Angel with a hole on one wing. (a) Triangle mesh highlighting the vicinity of the hole. (b) Points added inside the hole. (c) Reconstructed model after hole filling. (d) The actual model for comparison.

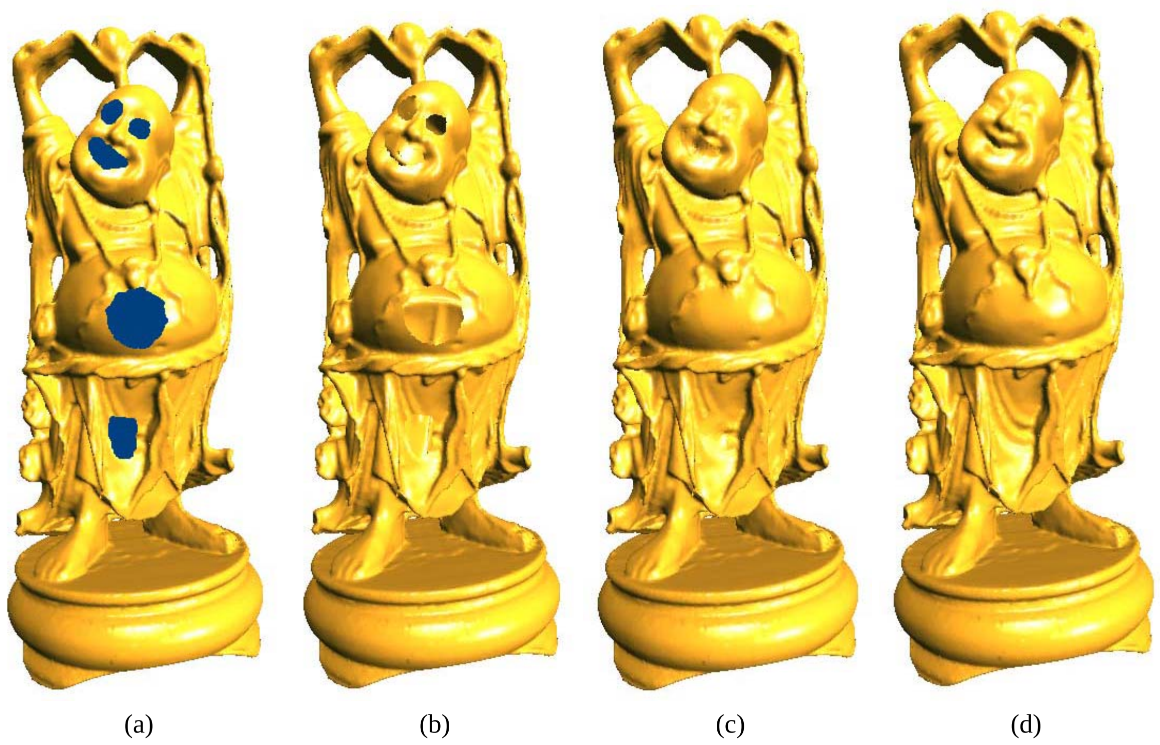

Fig. 11. Happy Buddha. (a) Dark regions indicate holes in the model shown in (b). (c) Model after hole filling performed by our algorithm. (d) Original model shown for comparison.