

# A Hole-Filling Strategy for Reconstruction of Smooth Surfaces in Range Images

JIANNING WANG<sup>1</sup>    MANUEL M. OLIVEIRA<sup>2</sup>

<sup>1</sup>SUNY at Stony Brook – Department of Computer Science, Stony Brook, NY, 11794-4400, USA  
jianning@cs.sunysb.edu

<sup>2</sup>UFRGS – Instituto de Informática - Caixa Postal 15064, 91501-970, Porto Alegre, RS, Brazil  
oliveira@inf.ufrgs.br

**Abstract.** Creating models of real scenes is a complex task for which the use of traditional modelling 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 recover both geometry and shading information, providing a good alternative when the properties to be reconstructed are locally smooth. The reconstruction process is mostly automatic and the sampling rate in the reconstructed areas follows the given samples. We demonstrate the use of the algorithm on both real and synthetic data sets to obtain complete geometry and reasonable shading.

## 1 Introduction

Creating accurate models of real environments is a non-trivial task for which traditional modelling techniques are inappropriate. In these situations, the use of laser rangefinders [5] 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 [15, 16]. Unfortunately, surface properties (*i.e.*, low or specular reflectance), occlusions and accessibility limitations cause the scanner to miss some surfaces, leading to incomplete reconstruction of the scene and introducing holes in the resulting models. Creating high quality mesh representations for objects in the scene based on such incomplete information remains a challenge [24]. Due to the costs and difficulties involved in scanning real environments, it is quite desirable to have automatic or semi-automatic tools for helping users to improve the quality of incomplete data sets.

The problem of filling holes in range data can essentially be divided into two sub-problems: identifying the holes, and finding appropriate parameterizations that allow the reconstruction of the missing parts using the available data. Unfortunately, none of these problems are trivial since holes arising during the scanning of geometrically rich objects, such as detailed sculptures, can be quite complex [9]. However, in many other situations of practical interest, holes occurring in range images can be topologically considerably simpler. This is the case of many holes found when scanning interior environments, where most surfaces tend to be smooth and planar areas are abundant (for example,

imagine a home or an office environment). For these situations, simpler algorithms for identifying holes and for parameterizing their vicinities can be specified that avoid the problems usually associated with the more general cases.

This paper presents an algorithm for automatically identifying and filling holes in point clouds in regions associated with smooth surfaces. Although our algorithm is targeted towards filling holes in smooth surfaces and, therefore, does not provide a general solution to the hole filling problem, it is conceptually very simple and its implementation is straightforward. Essentially, it takes a point cloud as input and produces an intermediate representation consisting of a triangle mesh, which is analyzed for the existence of boundary edges (edges that belong to a single triangle). The occurrence of a hole implies the existence of a cycle defined by boundary edges. Thus, once a boundary edge is found, the algorithm traces the entire boundary. A ring of points around the boundary, called its *vicinity*, is used to provide the context for an interpolation procedure. The vicinity is then used to fit a surface using a *moving least squares* (MLS) approach. Since MLS interpolates the original data, an important feature of our algorithm is to guarantee that the reconstructed patches smoothly blend into the original surface. Moreover, the reconstructed surface preserves the original sampling rate of the given samples. As the new primitives are distinguished from original points, they can be processed further. Since the algorithm works after surface reconstruction, it can be used with any reconstruction technique and its processing is limited to the vicinities of holes. We demonstrate the effectiveness of our approach on both real and synthetic data sets and show how

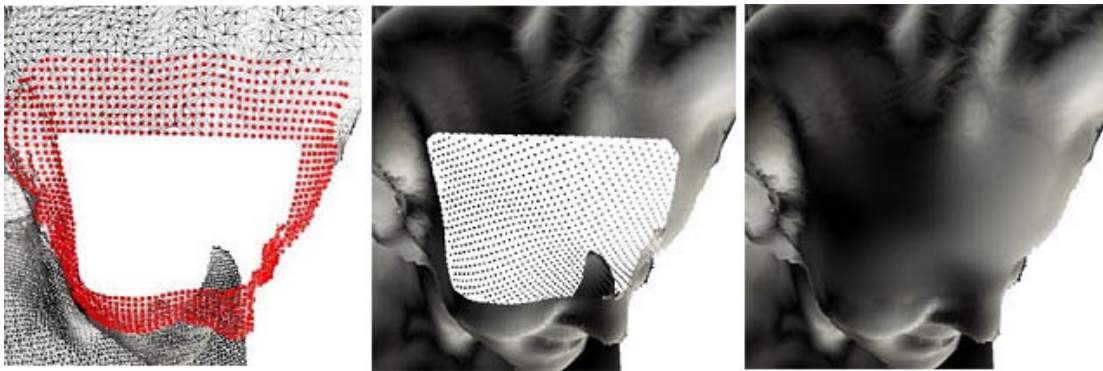


Figure 1: A statue with a hole on its head. Triangle mesh with the hole and vicinity identified (left). Interpolated points used to fill the hole (center). Reconstructed model (right).

it can significantly improve the overall quality of the models.

Figure 1 illustrates the use of the algorithm for the case of a synthetic model of a statue containing a relatively big hole. On the left, one sees the triangle mesh reconstructed from the point cloud. The highlighted points represent the vicinity of the hole. The figure on the center shows the re-sampled points on the surface and, to its right, the reconstructed model.

The remaining of the paper is organized as follows: Section 2 discusses some previous and related work. Section 3 presents a short review of the moving least squares method. The details of the algorithm are described in Section 4. Section 5 discusses some results and Section 6 summarizes the paper and points at directions for further exploration.

## 2 Previous and Related Work

Hole filling is an important problem in scene reconstruction and this work benefits from previous efforts in areas such as registration of range images [6, 17, 20] and surface reconstruction from point clouds [2, 3, 11, 12].

Curless and Levoy [8] used a hole filling technique to interpolate non-sampled surfaces in concave regions of objects. In this case, the added surfaces were intended to produce "watertight" models for reproduction using rapid prototyping techniques, and have little or no impact on the appearance of the objects. In our work, we are concerned about the reconstruction of holes that would, if not fixed, result in major rendering artifacts as the models are used in graphics applications.

Carr et al. [7] use polyharmonic radial basis functions (RBF) to fit an implicit representation to a set of sampled points. It consists of creating a signed distance function, fitting an RBF to the resulting distance function, and extracting iso-surfaces from the the fitted RBF. The existence of

an implicit representation supports the smooth reconstruction of missing areas. This technique is quite general and produces very impressive results, but the entire set of points is treated as a single implicit function. In order to create the signed distance function, the system needs to know what portion of the space corresponds to the exterior of the surface and what portion corresponds to its interior, which may not always be readily available.

Davis et al. [9] use a volumetric diffusion approach, analogous to inpainting techniques [4, 18], to fill holes in range scans. The process consists of converting a surface into a voxel-based representation with a clamped signed distance function. The diffusion algorithm consists of alternate steps of blurring and compositing, after which the final surface is extracted using Marching Cubes [14].

Alexa et al. [1] used 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. The reconstructed surfaces are used to down-sample or up-sample the set of points. Their method, however, does not attempt to fill holes on surfaces.



Figure 2: (Left) Chair extracted from the range image shown in Figure 8. Notice the existence of big holes due to occlusion. (Right) Symmetry-based Reconstruction (after [21]).

Wang and Oliveira [21] proposed a pipeline to improve the reconstruction of scenes represented as sets of range images. The pipeline consists of a segmentation step fol-

lowed by the reconstruction of missing geometric and textual information for individual objects. The reconstruction of missing geometric data exploits the fact that real (indoor) scenes usually contain many planar and symmetric surfaces. Thus, a 3D Hough transform is used to identify large planar regions, whose corresponding samples are replaced by texture-mapped polygons and removed from the point cloud. In the remaining dataset, individual objects are segmented as clusters of spatially close points, using an incremental surface reconstruction algorithm [11]. Inside each cluster, the point cloud is analyzed searching for approximately symmetric patterns using a variation of the 3D Hough transform [21]. As such patterns are identified, the algorithm automatically proceeds with reconstruction by mirroring data from one part of the model to another. Figure 2 shows a chair extracted from a real data set. The image on the left corresponds to the original samples. Notice the existence of large holes. The image to its right shows the model recovered using the symmetry-based reconstruction algorithm described in [21]. It provides a significant improvement with respect to the original model, but some holes are still visible. Such holes essentially result from the lack of data in both sides of the model and from limitations of surface reconstruction algorithms [2, 3, 11, 12] to work effectively on areas with variable sampling density. The algorithm presented in this paper is intended to fill in these remaining holes.

### 3 Moving Least Squares

Moving least squares (MLS) provide a class of complete solutions to the problem of fitting smooth functions to scattered data [13]. Compared with other interpolation techniques, MLS has the advantage that the reconstructed surface passes through the original data points. This guarantees that not only the original samples will remain unchanged, but also that the reconstructed patches used to fill the holes will smoothly blend into the original surface. This can be accomplished with the use of a relatively small number of samples in the vicinity of the holes. For these reasons, MLS has been selected as the interpolation technique for our hole filling strategy. Next, we provide a quick review of the MLS method.

Let  $s$  be a height function defined over a two-dimensional subspace ( $s \mid= U \subset \mathbb{R}^2 \rightarrow \mathbb{R}$ ) and let  $p_i \in U \subset \mathbb{R}^2$  be a point in the domain of  $s$ .  $f_i$  is a height measurement associated with  $p_i$ . The error of the fitting  $s$  through this set of points is given by

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

where  $N$  is the total number of points and  $w_i$  is the weight associated with point  $p_i$ . The solution is obtained by mini-

mizing the error  $E(s)$ . In practice, simple polynomial functions, such as the one shown in Equation 2, are usually used to represent the surface to be fitted [13].

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

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

$$a = (BWB^T)^{-1} BWf \quad (3)$$

where  $B$  is the following matrix and  $W$  is a matrix with diagonal equal to  $w_i$

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

The weighted least squares solution above can only represent low order surfaces, thus resulting in poor approximations in some cases. To reflect the fact that samples near the resampling position should have more influence than far away samples, the error function should take into account weight factors  $w_i$ , which ‘‘moves’’ with the evaluation point:

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

For this case, the following weight function is suggested [13]:

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

Here,  $d_i(p)$  is the distance between the new sampling position  $p$  and the  $i$ th original sample  $p_i$  (in the vicinity). This weight function becomes infinity when evaluated right at an input point, thus interpolating the input. To avoid numerical problems, evaluation right at input points are handled individually. The parameter  $\alpha$  controls the influence of vicinity features on the region to be re-sampled. Since the weight functions are dependent of the resampling positions, new coefficients for Equation 2 need to be computed individually for every re-sampled point as:

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

where  $W(p)$  is a diagonal matrix of the weights 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. Therefore it can fit higher order surfaces.

## 4 The Hole Filling Algorithm

In order to fill holes, new points need to be added into the unsampled regions. First, the algorithm identifies holes and their vicinities and then orthographically projects each vicinity onto a plane, thus reducing the problem of reconstructing holes in 3D to the simpler interpolation problem. The basic version of the algorithm is presented in Algorithm 1, and its details are explained next.

- Starting with a point cloud, create a triangle mesh
- Repeat
  - Automatically find a boundary and its vicinity
  - Compute a projection plane for the hole vicinity
  - Determine the resampling positions on the plane
  - Fit the surface and resample points
- Until no holes exist

**Algorithm 1.** The hole filling algorithm

### 4.1 Finding the Holes

A number of algorithms can be used to create triangular meshes from point clouds [2, 3, 11, 12]. We currently use an incremental surface reconstruction algorithm [11] for this purpose. The decision of using this particular algorithm was based on its relatively simple implementation and support for preserving the original sampling density in the reconstructed holes.

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 5. From the example of the cylinder, it becomes clear that not all holes should be filled. In order to guarantee that proper reconstruction is obtained in all cases, user assistance is required.

Ideally, new points used to fill holes should preserve the original sampling density at the vicinity of the hole. Once a hole is identified, the following equation provides a heuristic spacing the resampling apart inside the hole.

$$stepsize = \sqrt{\frac{area}{3n}} \quad (8)$$

Here,  $n$  is the number of points on the boundary of the hole and area is the sum of the areas of 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  $\beta$ .

### 4.2 Computing the “Tangent” Plane

The next step is to orthographically project all vicinity points onto the approximate tangent plane at the vicinity, and find the resampling positions necessary to fill in the hole. The orientation of the plane is computed as follows. First, the

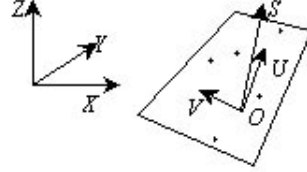


Figure 3: The  $UV$  projection plane

average  $O = (\bar{x}, \bar{y}, \bar{z})$  of all vicinity points is computed as the origin of a new coordinate system associated to the plane. *Singular Value Decomposition* (SVD) [19] is used to decompose the matrix  $M$  obtained by subtracting  $O$  from all points in the vicinity (Equation 9). The eigenvector corresponding to the eigenvalue of  $M$  with smallest absolute value is used as the normal of the plane ( $S$  axis). The other two eigenvectors are taken as the  $U$  and  $V$  axis (Figure 3). After projection into the  $UV$  plane, every vicinity point will have  $(u, v)$  coordinates and a height  $s$  computed as its distance from the  $UV$  plane. These coordinates are used to fit the surface using MLS.

$$M = \begin{bmatrix} x_1 - \bar{x} & y_1 - \bar{y} & z_1 - \bar{z} \\ x_2 - \bar{x} & y_2 - \bar{y} & z_2 - \bar{z} \\ \cdot & \cdot & \cdot \\ x_{N-1} - \bar{x} & y_{N-1} - \bar{y} & z_{N-1} - \bar{z} \\ x_N - \bar{x} & y_N - \bar{y} & z_N - \bar{z} \end{bmatrix} \quad (9)$$

### 4.3 Determining the Resampling Positions

It is important that the reconstructed patches preserve the sampling density of the original set of points at the vicinity of the holes. Two criteria are used for determining the resampling positions:

- The projections of new points should be inside the projection of the hole;
- The minimum distance between any new point and all the others (both new and vicinity points) should be bigger than a threshold.

While the first criterion seems obvious, the second one is used to guarantee good remeshing results, since some reconstruction techniques require input points to be spaced as evenly as possible. In order to find 2D new sampling positions ( $u$  and  $v$  coordinates), the vicinity mesh is orthographically projected onto the  $UV$  plane, defining a mask. This situation is illustrated in Figure 4(b) 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 a scan line order using the step size compute using Equation 8. 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  $0.5 \times \text{stepsize}$ , such a point is not used as a resampling point.

#### 4.4 Fitting the Surface

For every point added, 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 color ( $R, G, B$  channels) of the new points are treated as three separate height functions and computed using MLS. After the new points have been introduced, the final step is to re-mesh the complete model. Figure 4 illustrates the intermediate steps of the algorithm for the simple case of a planar surface.

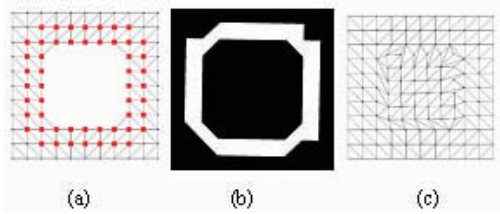


Figure 4: Intermediate results of the algorithm. (a) The triangle mesh with vicinity points highlighted. (b) Mask image. (c) Reconstructed mesh.

## 5 Results

We have implemented the hole-filling algorithm described using C++. An incremental surface construction algorithm [11] was used to create the initial triangle meshes and to re-mesh the model after new points were added. In the current implementation, we used  $\alpha = \frac{1}{16}$  (Equation 6). Figure 5 is meant to be a didactic example and illustrates the algorithm in action. The left picture shows a triangle mesh model with a hole on the surface. A number of points around the hole are highlighted, showing the vicinity. In the middle, new points inside the hole are computed using MLS. The right image shows the reconstructed mesh.

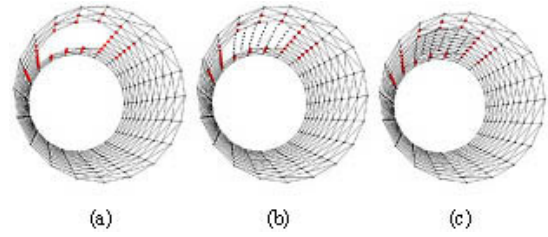


Figure 5: Cylinder. (a) Triangle mesh with the hole and vicinity identified. (b) New points added (c) Reconstructed mesh.

The algorithm was tested on two groups of data sets. The first group consists of range images taken from synthetic 3D models; the second, consists of range data obtained from a real scene.

The "lamp shade" data set (Figure 6) was obtained from 4 range images (from a synthetic model) that were merged and simplified. After merging, the remaining samples in the point cloud were spaced without any apparent pattern, thus simulating the case of an arbitrary point set. Figure 6(a) shows the mesh obtained from this point set. A hole is identified and its vicinity points are highlighted. In Figure 6(b), new points are added inside the hole. Figure 6(c) shows the reconstructed model. Figure 7 shows the reconstructed lamp model. The hole in the shade was filled using the algorithm described in this paper, whereas the lamp body reconstructed exploiting its symmetry [21].

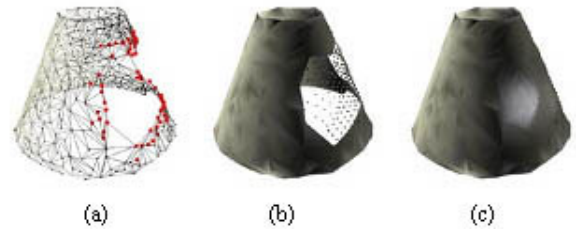


Figure 6: Lamp shade with a hole. (a) Triangle mesh with a hole and vicinity identified. (b) New points added to inside the hole. (c) Reconstructed model.

Figure 1 (Satyr 1) depicts a hole in the head of a synthetic statue. Figure 10 (Satyr 2) shows a hole in the chest and Figure 11 (Angel 1) shows a hole on one the angel's wings. The underlying surfaces for those holes are relatively smooth. In these cases, plausible reconstructions are obtained. In these figures, the vicinity points are highlighted to show the identified hole. The points added inside the hole and the reconstructed complete models are shown as well. In Figure 10 and 11, the actual models are shown on the right for comparison.

The last data set used to test the algorithm consists of

Data Set	# Original Pts	# Vicinity Pts	# New Pts	# Orig. Triangles.	# New Triangles	MLS time (sec.)
Cylinder	205	16	283	356	497	0.05
Shade	581	62	753	1090	1464	0.3
Satyr 1	12,853	1,506	16,162	21,757	29,357	72.85
Satyr 2	12,853	999	13,940	23,731	26,603	11.67
Angel 1	10,189	347	11,264	18,737	20,921	4.16
Chair	64,159	1,873	65,114	126,588	136,002	10.93

Table 1: Statistics and Running time for different data sets on a 2.0 GHz Pentium 4 PC with 512MB of memory

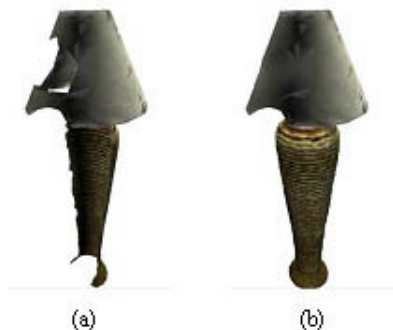


Figure 7: Lamp. (a) Incomplete model. (b) Reconstructed model. The body is reconstructed by symmetry. The shades are reconstructed using the algorithm in this paper.

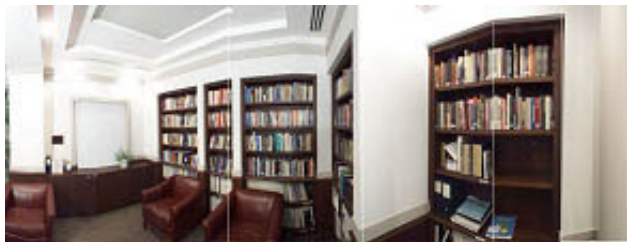


Figure 8: Real data set: Sitterson Hall reading room. Panorama obtained by concatenating 10 864x240 range images.

a partial model of the UNC reading room shown in Figure 8. In order to illustrate the effectiveness of our algorithm, we used the techniques described in [21] to segment a chair and applied the algorithm to it. The remaining of the scene (walls, shelves, etc.) was also reconstructed using the pipeline presented in [21] and used to create a background for the scene. The original scene was edited by replacing the floor texture and positioning one of the chairs in order to emphasize the existence of holes in it. Views of the edited scene are shown in Figure 9. Figure 9(a) shows the original samples of the chair rendered as a triangle mesh. Notice the existence of a big hole. Figure 9(b) shows a chair reconstructed exploiting its symmetry [21]. Figure 9(c) de-

picts the reconstruction obtained by applying the MLS algorithm described in the paper to the result shown in Figure 9(b). Notice that the holes have been eliminated. The original incomplete chair model has 41,511 points and 82,065 triangles. Those numbers grow to 64,159 for points and 126,588 for triangles after reconstruction using symmetry information. The complete chair model, reconstructed with the hole-filling algorithm, has 65,114 points and 136,002 triangles. Table 1 lists the statistics of test data sets and the running time for applying the moving-least-squares-based reconstruction.

Automatically finding hole boundaries is an  $O(n^2)$  procedure. The search for boundary edges is linear in the number of the original points. Once such an edge is found, tracing the boundary requires, in the worst case, following  $n-1$  edges. The running time of the MLS fitting depends on the number of new points  $k$  to be added and the size of vicinity  $m$ , thus  $O(mk)$ .

## 6 Conclusion and Future Work

Reflectance properties, occlusions and accessibility limitations can cause scanners to miss some surfaces, leading 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. Although the topology of holes found when scanning some real-world objects can be quite complex, in some situations of practical interest this topology can be significantly simpler. This paper described an algorithm for automatically identifying and filling holes on locally smooth surfaces. It consists of finding boundary edges and tracing them to identify holes. Once a hole has been identified, its vicinity is computed and projected onto a plane. On the plane, a height field function is defined and a moving least squares solution is used to reconstruct the surface.

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, the color interpo-

lation strategy cannot be used to reconstruct arbitrary textures. In this case, the use of texture synthesis such as the one described in [10, 22, 23] is more appropriate.

The algorithm cannot recover missing areas if such areas do not have closed boundaries. This situation is illustrated in Figure 7 where the shade of the lamp has a dent, caused by occlusion. Another limitation of the algorithm is that in order to obtain a proper parametrization of the hole vicinity, the projected boundary should not contain folds in the direction of the projection onto the plane. We are looking for more general parameterizations to deal with holes of arbitrary shapes.

## References

- [1] M. Alexa, J. Behr, D. Cohen-Or, S. Fleishman, D. Levin and C. Silva. Point Set Surfaces. *IEEE Visualization 2001*. pp. 21-28.
- [2] N. Amenta, M. Bern and M. Kamvyselis. A new Voronoi-based surface reconstruction algorithm. *SIGGRAPH'98*, pp.415-421, 1998.
- [3] C. Bajaj, F. Bernardini and G. Xu. Automatic Reconstruction of Surfaces and Scalar Fields from 3D Scans. *SIGGRAPH'95*, pp.109-118, 1995.
- [4] M. Bertalmio, G. Shapiro, V. Caselles and C. Ballester. Image Inpainting. *SIGGRAPH'00*, pp.417-424, 2000.
- [5] P. Besl. Advances in Machine Vision. *Advances in Machine Vision*, Springer Verlag, Chapter 1 - Active Optical Range Sensors. pp. 1-63. 1989.
- [6] P. Besl and N. McKay. A method for registration of 3D shapes. *IEEE Trans. on PAMI*, **14**(2): pp. 239-256, 1992.
- [7] J. Carr, R. Beatson, J. Cherrie, T. Mitchell, W. Fright and B. McCallum. Reconstruction and Representation of 3D Objects with Radial Basis Functions. *SIGGRAPH'01 Proc.*, pp. 67-76, 2001.
- [8] B. Curless and M. Levoy. A Volumetric Method for Building Complex Models from Range Images. *SIGGRAPH'96 Proc.*, pp. 303-312, 1996.
- [9] J. Davis, S. Marschner, M. Garr and M. Levoy. Filling Holes in Complex Surfaces using Volumetric Diffusion. *Proc. First International Symposium on 3D Data Processing, Visualization, Transmission*, 2002.
- [10] A. Efros and W. Freeman. Image Quilting for Texture Synthesis and Transfer. *SIGGRAPH'01*, pp: 341-348.
- [11] M. Gopi and S. Krishnan. A Fast and Efficient Projection Based Approach for Surface Reconstruction. *Intern. Journal of High Performance Computer Graphics, Multimedia and Visualisation* **1**(1):1-12, 2000.
- [12] H. Hoppe, T. DeRose, T. Duchamp, J.A. McDonald and W. Stuetzle. Surface reconstruction from unorganized points. *SIGGRAPH'92 Proc.* pp. 71-78, 1992.
- [13] P. Lancaster and K. Salkauskas. Curve and Surface Fitting, An Introduction. Academic Press. 1986.
- [14] W. Lorensen and H. Cline. Marching cubes: A high resolution 3D surface construction algorithm *Proc. SIGGRAPH'87*. pp. 163-169, 1987.
- [15] D. McAllister, L. Nyland, V. Popescu, A. Lastra and C. McCue. Real Time Rendering of Real World Environments. *Rendering Techniques'99 Proc.*, pp. 145-160, 1999.
- [16] L. Nyland, et al. The Impact of Dense Range Data on Computer Graphics. *Proceedings of Multi-View Modeling and Analysis Workshop*, pp. 3-10, 1999.
- [17] L. Nyland, A. Lastra, D. McAllister, V. Popescu and C. McCue. Capturing, Processing and Rendering Real-World Scenes. In *Videometrics and Optical Methods for 3D Shape Measurement, Electronic Imaging 2001, Photonics West*, SPIE Vol. **4309**, 2001.
- [18] M. Oliveira, B. Bowen, R. McKenna and Y. Chang. Fast Digital Image Inpainting. *International Conference on Visualization, Imaging and Image Processing (VIIP 2001)*, Marbella, Spain. pp.261-2665, 2001.
- [19] W. Press, S. Teukolsky, W. Vetterling and B. Flannery. Numerical Recipes in C: The Art of Scientific Computing. 2nd Ed., *Cambridge University Press*, 1992.
- [20] K. Pulli. Multiview Registration for Large Data Sets. *3DIM'99*, pp. 160-168, 1999.
- [21] J. Wang and M. Oliveira. Improved Scene Reconstruction from Range Images. *Proc. EUROGRAPHICS'2002*, pp. 521-530, 2002.
- [22] L.Y. Wei and M. Levoy. Texture Synthesis over Arbitrary Manifold Surfaces. *SIGGRAPH'01*, pp.355-360, 2001.
- [23] L. Ying, A. Hertzmann, H. Biermann and D. Zorin. Texture and Shape Synthesis on Surfaces. *Eurographics Rendering Workshop, 2001*. pp. 301-312, 2001.
- [24] Y. Yu, A. Ferencz and J. Malik. Extracting Objects from Range and Radiance Images. *IEEE Transactions on Visualization and Computer Graphics*, **7**(4), pp. 351-364, 2001.

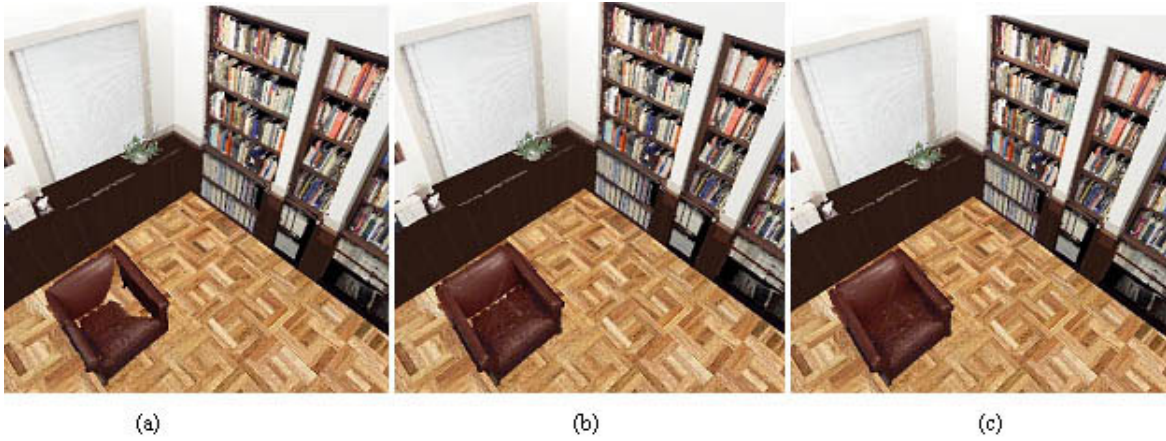


Figure 9: Edited UNC reading room. Two of the chairs were removed and the floor texture was replaced to emphasize the existence of holes. (a) Triangle mesh for the chair created using the original samples present in Figure 8. Notice the big hole. (b) Chair reconstruction exploiting symmetry, after [16]. Some holes are still visible. (c) Chair reconstructed applying the described algorithm to the result shown in (b). Notice that the holes have been eliminated.

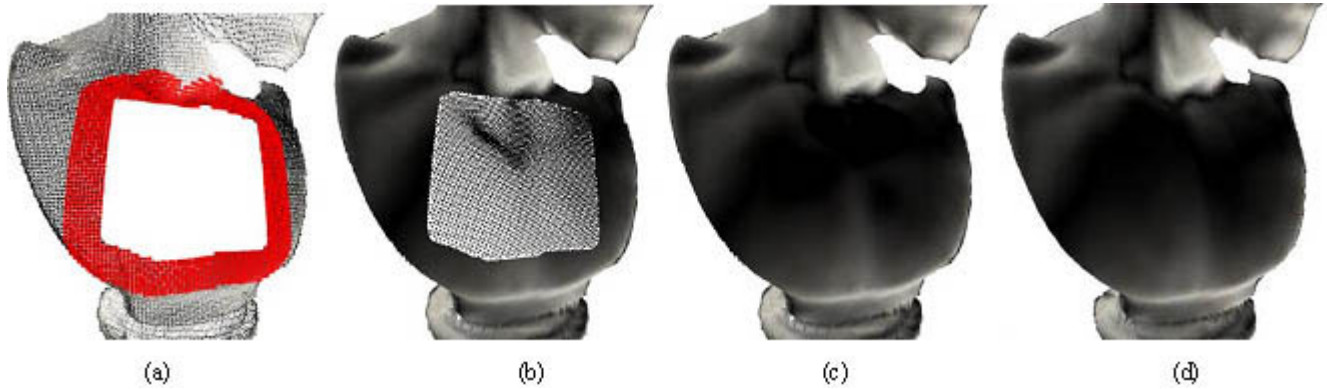


Figure 10: Satyr with a hole in his chest. (a) Triangle mesh with the hole and vicinity identified. (b) Points added inside the hole. (c) Reconstructed model. (d) The actual model for contrast.

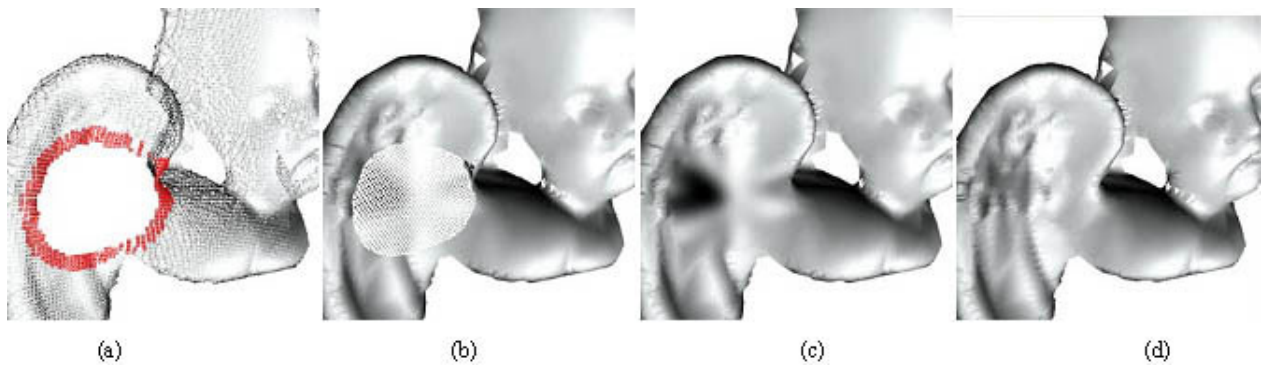


Figure 11: Angel with a hole on one of his wings. (a) Triangle mesh with the hole and vicinity identified. (b) Points added inside the hole. (c) Reconstructed model. (d) The actual model for contrast.