Image Guided Geometry Inference

Songhua Xu[†] Athinodoros Georghiades[†] Holly Rushmeier[†] Julie Dorsey[†] Leonard McMillan[§]

[†]Department of Computer Science
Yale University
New Haven, CT 06520

[§]Department of Computer Science
The University of North Carolina at Chapel Hill
Chapel Hill, NC 27599

Abstract

We introduce a new method for filling holes in geometry obtained from 3D range scanners. Our method makes use of 2D images of the areas where geometric data is missing. The 2D images guide the filling using the relationship between the images and geometry learned from the existing 3D scanned data. Our method builds on existing techniques for using scanned geometry and for estimating shape from shaded images. Rather than creating plausibly filled holes, we attempt to approximate the missing geometry. We present results for scanned data from both triangulation and time-of-flight scanners for various types of materials. To quantitatively validate our proposed method, we also compare the filled areas with ground-truth data.

1 Introduction

Three-dimensional scanners are powerful tools for capturing the complicated geometry of existing physical objects. Objects may be scanned for applications ranging from cultural heritage [5] to industrial design [12]. A problem with many scanners, however, is that they do not capture every region of an object, leaving visually distracting holes in the model [4]. Numerous methods have been proposed to fill holes, including extrapolating boundary data [6, 7], fusing scanner data with shape acquired with traditional computer vision techniques [1, 8], and copying geometry from other parts of the model into the area of the hole [17, 18]. In this paper, we present a new method for filling the holes using 2D images of the missing area. Our method combines previous ideas by expanding shape-from-shading methods to take advantage of relationships that can be learned from the existing scans and aligned images. These relationships are used to fill in missing areas in a manner consistent with their boundary data. Our goal is to approximate the actual object geometry in the missing regions, rather than to just plausibly fill the holes.

In this paper we consider systems that scan shape by actively reflecting light off the surface to form range images. The two major classes of range scanners are triangulation and time-of-flight scanners. Details of the principles of operations and limitations of these scanners are given in [10]. Triangulation scanners are typically used for smaller objects, up to a few meters in length, and can achieve depth accuracy on the order of 0.1mm at 1m standoff distance. Time-of-flight scanners are used for large objects on the order of 10m to 300m in size but, typically, have an accuracy on the order of 5mm at any distance, because of their dependence on precise time measurement. Both classes of scanners miss surface regions, because of the difficulties of positioning both the object and the scanner to obtain clear lines of sight. An object's fragility or location may make it impossible to capture some views, or the number of views may be limited by the time available for acquisition. For triangulation scanners, some concave regions are inaccessible for any scanner position, because clear lines of sight are required for both the emitter and the sensor.

Digital images are frequently captured during object scanning for use as texture maps when rendering. These digital images are often aligned with the captured geometry by calibrating the camera in the scanner's coordinates. Alternatively, images can be aligned to geometry by hand, using a few user-specified correspondences after the scanning is complete. A description of methods for acquiring and integrating images with a 3D scanned model is given in [3]. These *aligned* images, captured for texture mapping, often include views of the hole regions. We present an automated method that makes use of this information for hole filling.

In the following section, we discuss previous work in geometric hole filling. We then describe our automated method for detecting, subdividing, and filling holes using image data related to geometry. We show sample results for scans of objects with different reflectance properties, and also provide quantitative comparisons of the regions we filled to ground-truth data for the objects.

2 Previous Work

Holes in geometric models processed from scanned data are a well known problem. While holes are acceptable when displaying the portion of an object that has been accurately scanned [7], they are not acceptable in any visual application or in rapid prototyping. Standard software for processing 3D scanned data, such as Raindrop Geomagic® and Inus Rapidform^{TM}, include hole filling capabilities. Commercial software typically includes curvature-based hole filling to avoid abnormally flat regions on the model surface. In many practical applications, these curvature-based approaches fail because of unreliable data on the hole boundary, or the complexity of hole shapes [4].

Over the past several years, researchers have sought improved methods for hole filling. One class of such methods is to fit functions to boundary data. Davis et al. [7] defined the surface implicitly by diffusing signed-distance data in a volume enclosing the surface, allowing in turn the extraction of a watertight definition via isosurface extraction. Ju [14] also used a volumetric approach to formulate a general method for producing closed, polygonal models from polygon soups that have many different types of errors, in addition to holes. Carr et al. [6] incorporate hole filling into a general mesh-integration algorithm. They define Radial Basis Functions (RBFs) at each sample point and extract a surface from the resulting set of functions. Areas in holes are filled by the combination of RBFs associated with the points surrounding the hole. While further variations and improvements of these approaches have been made, these methods cannot produce detail in the filled area that is consistent with the rest of the object, and they make no use of the available information from the area to be filled.

Verdera et al. [19] recognized the similarity between filling holes in geometric meshes, and filling holes left in images. In both cases, a flat-fill area is undesirable. Inspired by the analogy between filling holes in 2D images and 3D geometries, other researchers, such as Sharf et al. [18], Blendels et al. [2], Park et al. [17], and Nguyen et al. [16], have developed methods that fill holes by copying data from other parts of the object into the hole. Sharf et al. [18] demonstrated the success of this approach in continuing large-scale object structures, such as ridges, and high-frequency spatial variations that are lost by simple extrapolation functions. Park et al. [17] demonstrated filling in both color and shape data from the object. By taking a hierarchical approach, Bendels et al. [2] illustrated how these methods can copy data at different spatial frequencies from different parts of the model, rather than simply copying existing regions. By flattening the geometry into an image, Nguyen et al. [16] showed how geometry hole filling can be processed in a way similar to 2D image inpainting. This method is particularly useful for filling holes resulting from geometry editing, rather than missing scan data.

An alternative to hole filling is to obtain more data via additional scanning. Fisher [11] performed a case study on a simple test scene to determine the practical number of views needed with a common scanner configuration. For the simple scene, over 100 views were needed. Even without "impossible" or restricted geometries, exhaustive scanning with a single scanner type is not practical. Fusing scanned data with that captured with a different device provides an alternative hole-filling approach. Dias et al. [8] observed that shape from stereo can be used effectively, since images captured for texture maps often cover the hole region in multiple views. This allows for the automatic alignment of the relatively sparse, high-confidence stereo correspondences into the denser scans. One could extend this basic idea to any computer vision technique—any method of estimating shape from images could be used to fill holes in the scanned model. Apart from aligning the recovered geometry to the scanned geometry, such methods do not take advantage of the information available in areas where both geometry and images have been captured.

Recent methods have been introduced that infer shape, or improve shape estimates, by combining data from scanned shape and from captured images captured under multiple lighting conditions. Hertzmann and Seitz [13] developed a method that learns the relationship between captured images and shape by imaging known object shapes of known materials under multiple lighting conditions. Based on the relationships learned, they inferred the shape of an unknown object composed of unknown materials using photometric stereo. Nehab et al. [15] took advantage of sets of 2D images aligned with the scanned geometry to improve the quality of the geometric reconstruction. By combining normals estimated using photometric stereo with the existing scanned data, more accurate geometry was computed.

The method we propose computes new geometry that is consistent with the hole boundary. Similar to infilling methods, we use geometric information from the areas of the object already scanned. To accomplish this, we infer geometry using relationships learned between the images and the existing scanned geometry. Unlike [13] and [15], we cannot rely on the existence of multiple lit images from the same viewpoint. Instead, we learn a mapping from image regions to the scanned geometry. Similar to the approach in [16], the surface is reparameterized by projecting it into the camera view of the guiding image. The normals in the hole regions are then estimated based on training examples from the same geometric model. The holes are filled by inpainting normals selected using the guiding image, while enforcing continuity with the previously estimated normals. The 3D surface is then recovered by integrating the surface normals. In the next section, we describe our automated method in detail, and discuss issues such as how to avoid shading ambiguities and how to deal with non-height-field hole boundaries.

3 Method

Our hole-filling process begins with a merged and assembled 3D model, and an associated set of aligned scan-view images, which we assume cover the hole regions. Our automated method proceeds through a series of steps. First the 3D model is processed to compute estimates of the surface normals. An association of surface normals to image patches is then learned. Next, holes in the 3D model are detected and dilated slightly. The remaining geometry is then projected onto each of the aligned scan-view images, and the projected hole regions are detected. Within these hole regions, surface normals are estimated using the image-intensity-to-normal mapping function that was learned. Finally, a surface is computed by integrating the estimated normals using the hole boundary as constraints.

Scan Pre-Processing. Given a triangle mesh, the vertex normals are estimated by a weighted averaging of the normals of the faces adjacent to each vertex. The mesh is projected into each scan view image. Normals are interpolated for triangles larger than a pixel, while they are averaged with a sample-center-biased weighting, when multiple triangles fall into the same pixel. This is accomplished by rendering at a higher resolution than the aligned scan-view image, followed by a filtering and sub-sampling.

Learning an Image-to-Normal Mapping. A key element of our approach is that a mapping is learned from image regions to surface normal estimates. We use a data-driven learning approach reminiscent of Efros and Leung [9] texture synthesis. A set of training patches is collected by projecting the scanned model, and scan-converting its associated normals onto each of the scan views. Those interpolated normals that are either facing away from the camera or are occluded are thrown away. Illuminated pixels are detected by checking the consistency of the image intensities with the orientation of the surface normals (i.e., the surface normals must be towards the camera). For the remaining pixel locations, all square patches of normals of a given size (currently 5×5 patches are used) are collected, where every pixel has a valid front-facing normal. To improve performance, the images are scaled down by a factor of 4. To reduce memory consumption and also to speed up the algorithm, 80% of the training patches are randomly thrown away.

The remaining training patches with their associated intensities and normals are stored as a file for use in the normal-patch transfer step. The correlation between the set of image intensities in the target and the training set candidates represents the learned relationship from image patches to normals.

Hole Detection. Our method only treats holes that are topological disks, nonetheless, their 3-D shape can be quite complicated. Holes are detected by finding edges that are not shared by 2 triangles. These are considered as boundary edges. Next, the boundary edges are traced to extract the loops defining each hole. Once a hole is detected, the hole boundary is dilated by removing all triangles that have a vertex on a boundary edge. This process can be repeated to remove successive rings around each hole. Dilation reduces the influence of scanning errors, which could lead to inaccurate normal estimates near silhouettes.

Hole Mask Creation. The process of creating a *hole mask* is illustrated in Fig. 1 for a particular hole (a) and scan view (b). The creation of a hole mask begins by projecting detected hole boundaries onto each of the scan-view aligned images , e.g. as in Fig. 1(c). This provides an oriented projected contour of the hole. The surface normals are computed pixel by pixel in the image, and back facing pixels are identified, i.e. the yellow pixels in Fig. 1(c). Occlusion detection is also applied at this step to remove pixels from the hole mask which are occluded by any part of the scanned mesh. If the area of the resulting hole mask is too small, the hole is marked as invalid for that image, and it is not considered in subsequent processing steps of that view.

Hole Subdivision. At this point, each hole is subdivided according to the normal orientations and depth variations along its boundary. The angle spanned by the normal direction of the hole-boundary pixels and the camera direction for each scan view is computed. This angle value is diffused from the boundary pixels inward, Fig. 1(d). After diffusing, the hole mask is subdivided by removing those pixels whose spanning angles are less than a user-defined threshold, that is in Fig. 1(d) the dark red pixels are kept, but the dark yellow ones are eliminated. After each iteration, this threshold is reduced, starting from 120 degrees, and gradually being reduced to 90 degrees. (Note that the spanning angle of a normal facing the camera is 180 degrees.) The resulting image will be used as the target mask image for that particular hole in the current iteration. The hole is also subdivided when the variation of depth exceeds a threshold, to ensure the satisfaction of the height field assumption, that is the blue pixels in Fig. 1(e) are eliminated. The hole subdivision attempts to avoid using portions of hole boundaries whose local depth variation is high, suggesting the possibility of high mesh curvature in that region. The target mask is examined and only the largest connected component is kept, discarding the remaining components, resulting in the green region in Fig. 1(f). If the size of the largest connected component is less than a given threshold, this scan view is

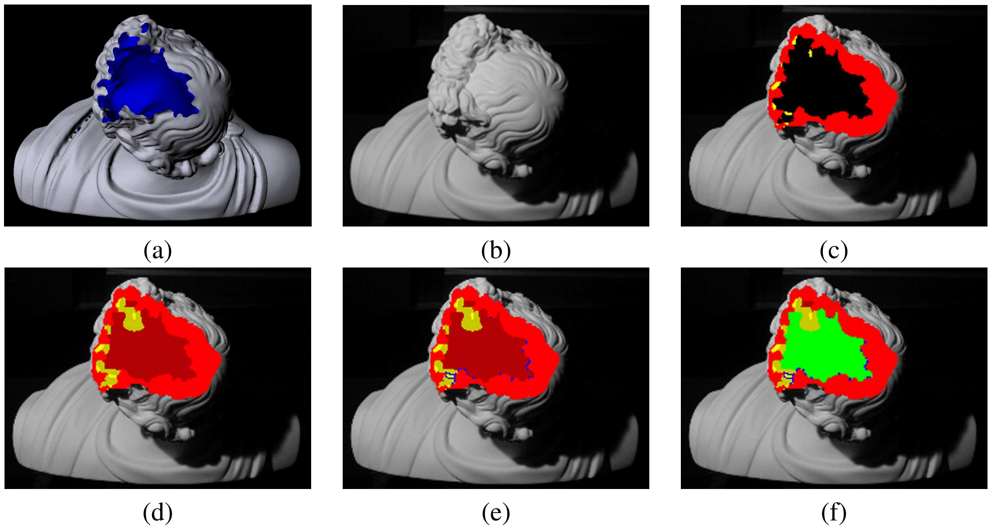


Figure 1. Creating a hole mask. (a) The model with a hole (back facing normals are shown in blue). (b) A digital image in a view to be considered. (c) The boundary of the geometric hole in (a) is projected into the view of the image (b). Surface regions facing toward the camera are shown in red; regions facing away shown in yellow. (d) The normals from the hole boundaries are diffused inward. Regions facing the camera are marked dark red, while those facing away are marked dark yellow. (e) For samples with depths very different from their neighbors, we mark out small neighborhoods that cannot be filled in this view (shown in blue here). (f) The largest connected component in the remaining dark red area is selected for hole filling. The valid hole mask is marked green at this step.

disregarded when filling that hole. Again, the thresholds are decreased in each iteration.

View Selection. In this stage, the best image is selected for filling a given hole. Our system selects the scan-view image where the hole mask is the largest in terms of "diffusely-lit" pixels. The focus on diffusely-lit pixels avoids using images where surface regions are in shadow or highlights, and thus provide little or no information from which to estimate the normal. As the objective function for selecting the best view, the sum of the size of the hole filling mask and the magnitude of the shading variation is used.

Normal Patch Transfer. Given the hole filling mask and the neighboring normal conditions, normals are transferred using the geometry inference algorithm. The objective function used for geometry inference is the sum of consistency with the local intensity distribution and consistency with the local normal distribution. The weight assigned to the intensity distribution is dynamically varied according to the shading variation. A high shading variation leads to a low weight value as it suggests a sudden change of geometry feature locally. The local intensity distribution is directly sampled from the photograph of the selected view and the local normal distribution refers to those normals which are around the target pixel to be synthesized. These normals are either from the original, partially scanned model, or the normals of already synthesized target pixels.

Surface Integration and Clean up. Once the normals of the hole are estimated, they are integrated to generate the hole geometry. At each step the hole filling mesh is zippered with the original mesh of the partially scanned model. After the last iteration, MeshLab is used to merge the intermediate hole filling meshes.

4 Results

We present results for a triangulation scanner with tabletop-scale objects, and a time-of-flight scanner with an architectural-scale object. For the triangulation scanner, we used objects we could scan thoroughly to obtain ground-truth data for comparing our hole filling results. To examine whether we are getting useful improvements by using the color images, we used software provided by other researchers for robust hole-filling methods that do not use image data—Volfill [7] and Polymender [14].

Our tabletop system uses a ShapeGrabber scanner, attached by a metal frame to an Olympus C8080WZ camera calibrated in the scanner coordinate system. A light is also attached to the frame, but we do not measure its position and orientation relative to the camera. We rely only on the information that the light position is the same with respect

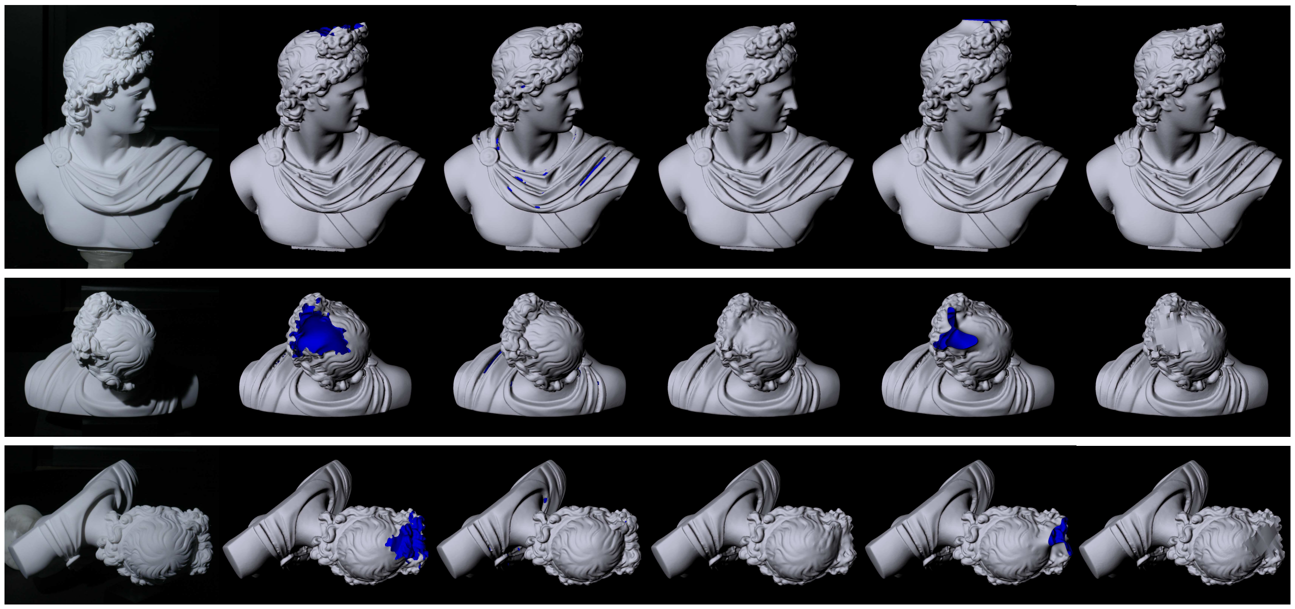
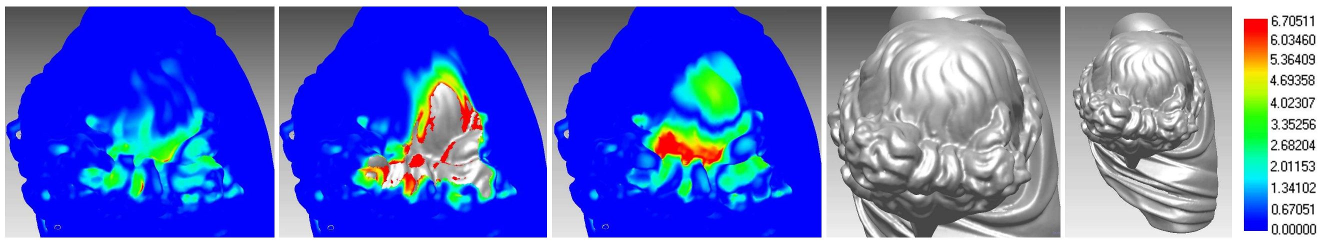

(a) (b) (c) (d) (e) (f)

Figure 2. Hole filling experiment with the Apollo statue. (a) Photograph of the statue; (b) the initial model before hole filling; (c) ground-truth model; (d) our hole filling result; (e) result from Volfill; (f) result from Polymender.



(a) (b) (c) (d) (e) (f)

Figure 3. Analysis of errors in hole-filling results relative to the ground truth for the Apollo model. (a, b, c) The errors for results generated by our algorithm, Volfill and Polymender (in (b), the grey region is where a hole was left by Volfill); (d) the ground-truth geometry model in the same view; (e) the entire geometry model; (f) the color code used to represent the distances between filled and ground-truth models in mm.

to the camera for all scans. We obtained data at a resolution of approximately 1mm. We align and form integrated meshes using the MeshLab tools from the Visual Computing Lab-ISTI-CNR [20]. We created test objects with holes by omitting scanned data before processing the scans into an integrated mesh. By taking this approach, rather than cutting a hole in a fully processed object, we have irregular hole shapes and unreliable edge data that are characteristic in scanned models with holes.

The hole filling software packages we compare to have a number of parameters that affect timing and to a small extent the quantitative results. We have not tuned our code for performance, and cannot guarantee that we have set the parameters for the other methods to obtain optimal performance. We give approximate timings based on several trials, and measures of the model and hole complexity to provide a sense, rather than precise comparison, of performance trade-offs.

Figure 2 shows results for a 26cm tall sculpture of Apollo. This is the same hole as shown in Fig. 1. The hole boundary is not a height field, and is subdivided to be filled using several different images as guides. The model with the hole to be filled contains 143,508 vertices with a surface area of 127,518mm^2, the complete ground-truth model has a surface area of 130,499mm^2. Our method ran approximately 8 hours to produce the filled result; Polymender required less than 20 minutes. We allowed Volfill to run for 48 hours, but it did not converge to a closed solution. The results from our hole filling method and the other methods are compared quantitatively with the ground-truth model in Fig. 3, where the distance between the filled models and the ground truth is color coded on the ground-truth model.

For the Apollo model, the major benefit that can be observed of using the image data to guide the hole filling is that the crease between the front part of the hair and the back has been preserved, rather than being smoothed out or left as a hole. Some, but not all, of the finer detail in the hair is also captured with the image guided approach.

Figure 4 shows the experiment results of filling holes on an 18cm tall statuette of Rembrandt. The object has a spatially varying surface albedo and is also somewhat shiny. The model with holes to be filled contains 73,294 vertices with a surface area of 37,480mm^2. The complete ground-truth model has a surface area of 44,218mm^2. To obtain the results shown, our method required 4 hours, Volfill approximately 5 hours, and again Polymender finished in less than 20 minutes. The results from our hole filling method and the other methods are compared quantitatively in Fig. 5.

For the Rembrandt model, there is no significant difference between the various hole filling methods on the front of the model. On the back however, our image-guided approach better preserved the shape on the upper portion of the back of the statuette. In this region, Volfill overestimated the ridge on the top left, and Polymender flattened out the concave regions. Our method provided inferior, bumpier, results for the small ridge at the bottom of the back. One possible source of this result is that the scanned geometry used to form the training set was bumpier than the true physical surface.

The time-of-flight system used for our tests is a Cyrax 2500. Color images were obtained with an Olympus C8080WZ digital camera. We aligned the color image to the scan geometry by selecting corresponding points in the color images and the scans to compute the camera parameters. The scanned facade has a spatially varying surface albedo. A 12.5m by 15.1m section was scanned at a 0.5cm resolution, and simplified to a model of 83,942 vertices. It was not possible to measure the ground-truth values for the areas of holes. Figure 6 shows the geometry with holes and one of three color images used for hole filling. Figure 6 (c) through (f) show the filled geometry (as well as the texture-mapped versions) for the section shaded red in Fig. 6 (a) using our algorithm and the volumetric hole filling algorithm. For filling the hole shown, our method took approximately 4 hours, while VolFill took about 5 hours. Since Polymender seeks to produce a closed model, we were not able to produce with it a comparable result for this experiment. While neither our method nor Volfill faithfully produced the missing data, using the image information our method estimated a shape that produces substantially less distortion of the texture map.

5 Conclusion

This paper presents a novel method for filling holes in a geometric surface model assuming calibrated images of the hole region are available. It uses a three step approach. First holes of missing geometrical data are identified; next a relationship between local surface normals and image intensities is derived; finally the latter relationship is used to do an image based inference of the normals for the part where the geometry is missing. Our method is automatic, requiring no user intervention. Experimental results on indoor objects clearly verify the effectiveness of our newly proposed method. A preliminary example of outdoor scene shows encouraging results and proves the concept of our new algorithm. In the future, we intend to accelerate our naive implementation of the algorithm and also work on improving the algorithm performance and quality on outdoor cases.

We gratefully acknowledge the authors of the Volfill and Polymender software for making their software publicly available. This work was funded by NSF Grant CCF-0528204.

References

[1] A. Abdelhafiz, B. Riedel, and W. Niemeier. Towards a 3D true colored space by the fusion of laser scanner point cloud and digital photos. *Proceedings of the ISPRS Working Group V/4 Workshop (3D-ARCH 2005)*, 2005.

[2] G. Bendels, R. Schnabel, and R. Klein. Detail-preserving surface inpainting. *The 6th International Symposium on Virtual Reality, Archaeology and Cultural Heritage VAST*, pages 41–48, 2005.

[3] F. Bernardini and H. Rushmeier. The 3D model acquisition pipeline. *Computer Graphics Forum*, 21(2):149–172, 2002.

[4] W. Boehler, M. B. Vicent, K. Hanke, and A. Marbs. Documentation of German Emperor Maximilian I's tomb. *The ISPRS International Archives of Photogrammetry and Remote Sensing Vol. XXXIV and The CIPA International Archives for Documentation of Cultural Heritage Vol. XIX*, pages 474–479, 2003.

[5] M. Callieri, P. Cignoni, F. Ganovelli, G. Impoco, C. Montani, P. Pingi, F. Ponchio, and R. Scopigno. Visualization and 3D data processing in the David restoration. *IEEE Comput. Graph. Appl.*, 24(2):16–21, 2004.

[6] J. C. Carr, R. K. Beatson, J. B. Cherrie, T. J. Mitchell, W. R. Fright, B. C. McCallum, and T. R. Evans. Reconstruction and representation of 3D objects with radial basis functions.

In *SIGGRAPH '01: Proceedings of the 28th annual conference on Computer graphics and interactive techniques*, pages 67–76, New York, NY, USA, 2001. ACM Press.

[7] J. Davis, S. R. Marschner, M. Garr, and M. Levoy. Filling holes in complex surfaces using volumetric diffusion. In *First International Symposium on 3D Data Processing Visualization and Transmission (3DPVT'02)*, pages 428–433, 2002.

[8] P. Dias, V. Sequeira, F. Vaz, and J. G. Gonalves. Registration and fusion of intensity and range data for 3d modelling of real world scenes. *Proceedings of the Fourth International Conference on 3-D Digital Imaging and Modeling (3DIM)*, pages 418–426, 2003.

[9] A. A. Efros and T. K. Leung. Texture synthesis by non-parametric sampling. In *ICCV '99: Proceedings of the International Conference on Computer Vision-Volume 2*, pages 1033–1038, Washington, DC, USA, 1999. IEEE Computer Society.

[10] F. F. Blais. Review of 20 years of range sensor development. *Journal of Electronic Imaging*, 13(1):231–240, January 2004.

[11] R. B. Fisher. Solving architectural modeling problems using knowledge. *Proceedings of the Fourth International Conference on 3-D Digital Imaging and Modeling (3DIM)*, pages 139–146, 2003.

[12] T. Funkhouser, M. Kazhdan, P. Shilane, P. Min, W. Kiefer, A. Tal, S. Rusinkiewicz, and D. Dobkin. Modeling by example. *ACM Trans. Graph.*, 23(3):652–663, 2004.

[13] A. Hertzmann and S. M. Seitz. Example-based photometric stereo: Shape reconstruction with general, varying brdfs. *IEEE Trans. Pattern Anal. Mach. Intell.*, 27(8):1254–1264, 2005.

[14] T. Ju. Robust repair of polygonal models. *ACM Trans. Graph.*, 23(3):888–895, 2004.

[15] D. Nehab, S. Rusinkiewicz, J. Davis, and R. Ramamoorthi. Efficiently combining positions and normals for precise 3D geometry. *ACM Trans. Graph.*, 24(3):536–543, 2005.

[16] M. X. Nguyen, X. Yuan, and B. Chen. Geometry completion and detail generation by texture synthesis. *Proceeding of Pacific Graphics (PG'05) Macao, China. Oct 12- 14, 2005. (Special issue of The Visual Computer, Springer-Verlag.)*, 2005.

[17] S. Park, X. Guo, H. Shin, and H. Qin. Shape and appearance repair for incomplete point surfaces. In *ICCV '05: Proceedings of the Tenth IEEE International Conference on Computer Vision (ICCV'05) Volume 2*, pages 1260–1267, Washington, DC, USA, 2005. IEEE Computer Society.

[18] A. Sharf, M. Alexa, and D. Cohen-Or. Context-based surface completion. *ACM Trans. Graph.*, 23(3):878–887, 2004.

[19] J. Verdera, V. Caselles, M. Bertalmio, and G. Sapiro. Inpainting surface holes. In *ICIP*, pages 903–906, 2003.

[20] Visual Computing Lab ISTI-CNR. Meshlab. http://meshlab.sourceforge.net/.

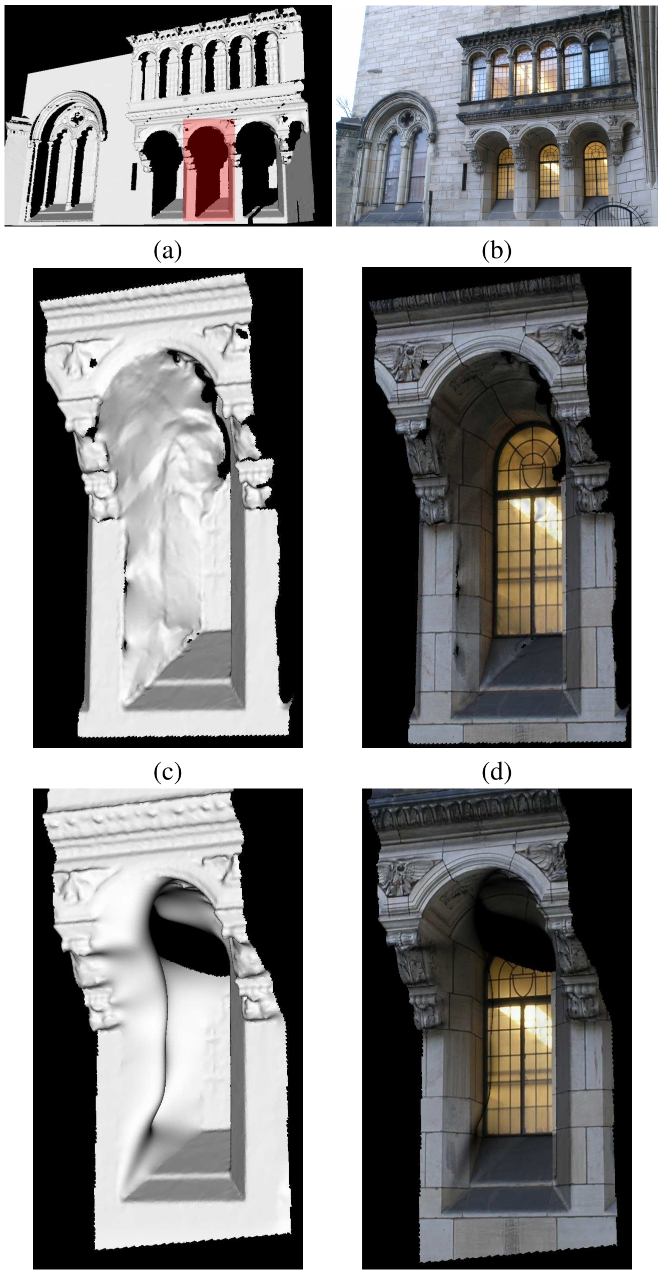


Figure 6. An example of hole filling on data obtained with a time-of-flight scanner and digital camera. (a) The facade geometry model captured through scanning. In this example, only the hole filling result for the region within the red box is shown. (b) One of three photographs used for hole filling. (c) The geometry model resulting from our hole filling algorithm; (d) its texture mapped version. (e) The geometry model generated by Volfill; (f) its texture mapped version.

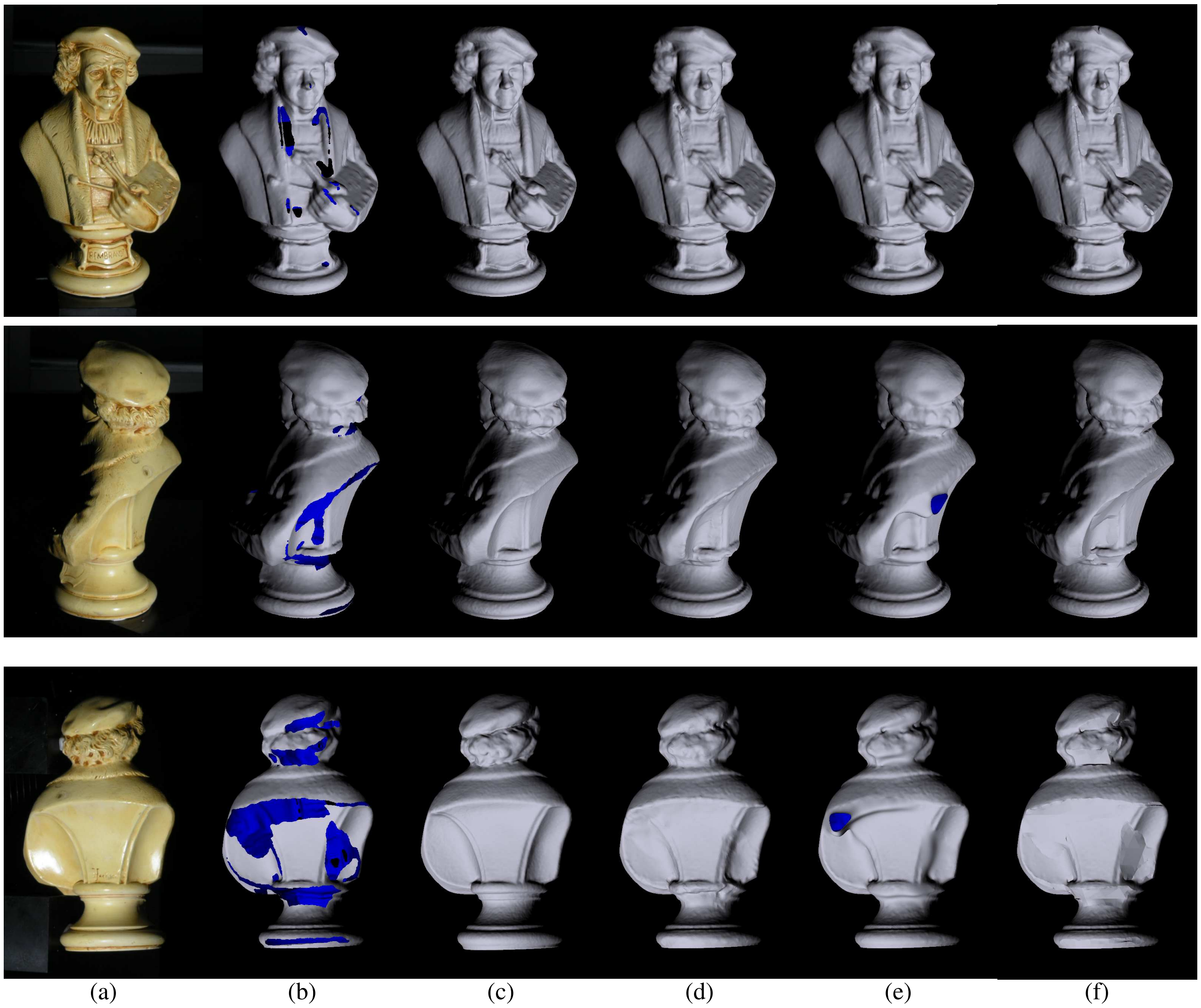


Figure 4. Hole filling experiment with the Rembrandt statuette. (a) Photograph of the statuette; (b) the initial model before hole filling; (c) ground truth model; (d) our hole filling result; (e) result from Volfill; (f) result from Polymender.

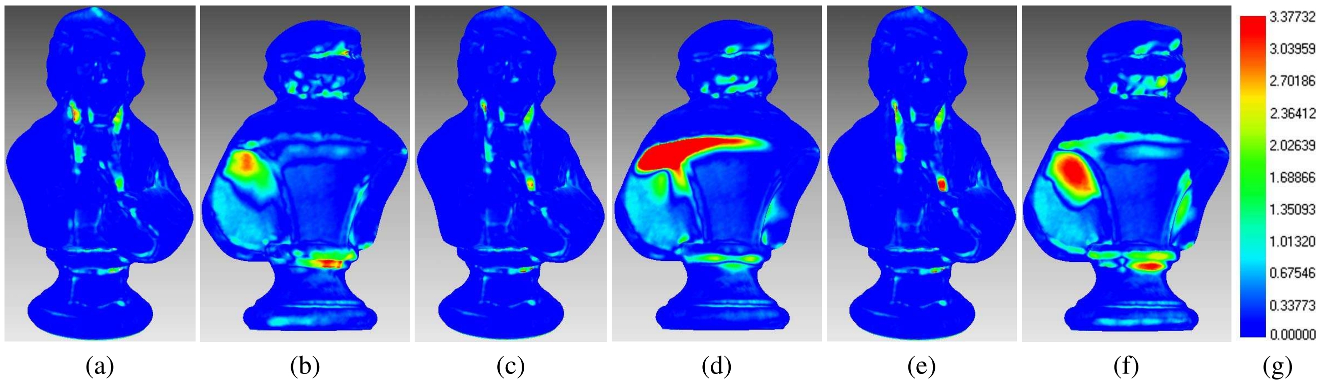


Figure 5. Analysis of errors in the hole filling results relative to the ground truth for the Rembrandt model. (a, c, e) The errors for results generated, respectively, by our algorithm, Volfill, and Polymender for the front of the model; (b, d, f) the errors generated for the back of the model in the same order; (g) the color code used to represent the distances between filled and ground-truth models in mm.