Filling Holes in Triangular Meshes Using Digital Images by Curve Unfolding *

Alan Brunton
University of Ottawa, Ottawa, Canada
National Research Council of Canada, Ottawa, Canada
abrunton@site.uottawa.ca

Stefanie Wuhrer
National Research Council of Canada, Ottawa, Ontario, Canada
stefanie.wuhrer@nrc-cnrc.gc.ca

Chang Shu
National Research Council of Canada, Ottawa, Ontario, Canada
chang.shu@nrc-cnrc.gc.ca

Prosenjit Bose
School of Computer Science, Carleton University, Ottawa, Ontario, Canada
jit@scs.carleton.ca

Erik Demaine
Computer Science and Artificial Intelligence Laboratory, Massachusetts Institute of Technology, Cambridge, Massachusetts, USA
edemaine@mit.edu

Received (Day Month Year)
Revised (Day Month Year)
Accepted (Day Month Year)

Communicated by (xxxxxxxxxx)

We propose a novel approach to automatically fill holes in triangulated models. Each hole is filled using a minimum energy surface that is obtained in three steps. First, we unfold the hole boundary onto a plane using energy minimization. Second, we triangulate the unfolded hole using a constrained Delaunay triangulation. Third, we embed the triangular mesh as a minimum energy surface in $\mathbb{R}^3$. When embedding the triangular mesh, any energy function can be used to estimate the missing data. We use a variational multi-view approach to estimate the missing data. The running time of the method depends primarily on the size of the hole boundary and not on the size of the model, thereby making

*Partial results of this work appeared in the 3-D Digital Imaging and Modeling Conference 2007 [A. Brunton, S. Wuhrer, C. Shu, Image-Based Model Completion, 3DIM 2007] and in the IEEE International Conference on Shape Modeling and Applications 2009 [A. Brunton, S. Wuhrer, C. Shu, P. Bose, E. D. Demaine, Filling Holes in Triangular Meshes by Curve Unfolding, SMI 2009].

the method applicable to large models. Our experiments demonstrate the applicability of the algorithm to the problem of filling holes bounded by highly curved boundaries in large models.

Keywords: Hole Filling, Multi-view Stereo, Space Curve Unfolding.

1. Introduction

The wide use of laser range sensors and other types of 3D sensing devices has produced increasingly detailed 3D models. However, holes, due to a variety of reasons, are usually present in models built from range scans. Some holes are caused by the intrinsic limitations of the sensors. Others are the result of object self-occlusion, insufficient view coverage, and shallow grazing angles. Davis et al. [10] give an account of holes and their causes in different situations.

For many computer-aided design applications such as reverse engineering, holes are unacceptable. The reason is that holes may lead to unexpected results in numerical simulations such as finite element simulations. It is therefore crucial to fill these holes.

Many methods have been proposed for hole filling. The majority of them fill holes by interpolating the nearby geometry [10]. This method is only effective when the holes are small. Manual, interactive tools are often used to fix large holes. To keep the surface smoothness the same as the nearby areas, interactive methods were proposed in a way that imitates image touch-up tools such as those found in Adobe Photoshop where users "cut and paste" surface geometry from nearby areas [36].

We propose a novel approach to automatically fill holes in triangulated models. We assume that the holes are bounded by a simply connected loop that is an *unknot*. That is, it can be continuously deformed to a circle without introducing self-intersections. The hole can be large and the boundary loop can be highly curved. We fill the hole with an approximate minimum energy surface. A *mimimum energy surface (MES)* is a surface that minimizes a given energy functional over all surfaces with fixed boundary b. For the examples shown in this paper we use three energy functions: two simple Laplace based energies [39,24], and a stereo-based energy that aims to fill the hole by optimally matching the surface to a given set of input images [33]. However, other energy functionals, for example a discrete Willmore energy [5] or an energy to preserve smooth change in surface normal, could be used instead. The complexity of our algorithm depends primarily on the complexity of the hole boundary instead of on the complexity of the triangular mesh. Note that this is a significant improvement over algorithms that operate on the full mesh because the complexity of the boundary is often small even for large models.

To fill a hole bounded by a loop of boundary edges, the proposed approach proceeds as follows. First, the boundary loop is gradually unfolded to a simple planar polygon. During this unfolding, we constrain the motion such that the loop does not self-intersect. Second, the simple planar polygon is triangulated using a *constrained Delaunay triangulation* algorithm. The resulting triangulation contains all of the edges of the unfolded polygon, does not add any Steiner points, and maximizes the minimum angle over all triangulations that have the first two properties. Third, the triangulated patch is embedded in $\mathbb{R}^3$ using the known boundary positions. As the ordering of the vertices is maintained during the

unfolding, the resulting patch closes the hole. We refine the patch to have the same resolution as the surrounding surface. The interior vertices are positioned to approximate a MES. Note that any energy function can be used to position the interior vertices. In this paper we implement three types of MES. The first two minimize Laplace-based energies: an efficient formulation of Laplacian smoothing as an optimization problem also known as least-squares mesh [39], and discrete fairing using a second-order Laplacian energy [24].

For the third type of MES, we further propose a new image-based method for positioning the interior vertices. Given a set of calibrated input images showing the area of the hole, the initial filling surface is deformed to fit the image data using a surface-oriented multi-view stereo approach. This problem has a variational formulation – deforming a surface such that it minimizes a photoconsistency energy function [16]. It has been solved previously by PDE-based method using either the level-set method [16,33] or mesh deformation [13,19]. In both cases, an initial, closed surface that encloses the object to be modeled is assumed and the algorithm evolves until it converges to a minimal energy solution. We use a global image-matching energy that decouples the energy itself from the surface, while still allowing the energy gradient to be written in terms of the surface vertex positions. However, the problem is an ill-posed one and often the algorithm is trapped in local minima. We make use of the known boundary of the surface and solve a boundary-value problem. This eliminates many spurious local minima because the boundary imposes strong constraints. Furthermore, as we smooth the surface, occlusion and specularity, the two most prominent problems for stereo, are alleviated.

2. Related Work

Geometric methods for filling holes in a mesh model interpolate the hole boundary or extrapolate the surface geometry from the surrounding areas. Two types of representations are used: volumetric representations and triangular meshes.

The volumetric representation discretizes the surface mesh into regular 3D grids or an octree structure either locally or globally. Davis et al. [10] diffuse the geometry from the hole boundary to the interior until the fronts meet. This method handles complex topological configurations such as holes with islands. However, it may change the existing mesh. Podolak and Rusinkiewicz [32] embed the incomplete mesh in an octree and use a graph cut method to decide the connections between pairs of the hole boundaries. It resolves difficult boundary topologies globally. Ju [21] constructs a volume using an octree grid and reconstructs the surface using contours. Volumetric approaches work well for complex holes. However, they are time consuming. Furthermore, the topology of the generated result may be incorrect in case of large holes.

The triangle-based approaches to hole filling work directly on the surface mesh. The advantage of working directly with the surface mesh is that the rest of the surface is unchanged when the holes are filled. Let the mesh contain n vertices and let the mesh boundary contain m vertices. This class of algorithms usually only deal with holes bounded by a simple loop that is unknot. That is, holes with islands can usually not be filled using these algorithms.

Barequet and Sharir [4] give an $O(n + m^3)$ algorithm for triangulating a 3D polygonal boundary which represents the boundary of a hole. When triangulating the hole boundary, a divide and conquer technique is used. No new vertices are added to the mesh. Liepa [28] extends this method to include a surface fairing step. The high complexity of the algorithm limits the use of this method to small meshes. Jun [22] proposes another method based on subdividing the hole into simple regions. Each simple hole is filled with a planar triangulation. This algorithm is not guaranteed to find a result. Li et al. [27] extend the method to achieve higher efficiency and stability. However, the algorithm is not guaranteed to find a result for arbitrary holes.

Dey and Goswami [11] use a Delaunay triangulation-based method, called Tight Cocone, in which tetrahedrons are labeled as in or out. In this method, no extra vertices are added to the mesh. The method is shown to perform well to fill small holes. However, the method is inappropriate to fill large holes because the geometry is extrapolated from the nearby boundary.

Carr et al. [7] use radial basis functions (RBF) to compute an implicit surface covering the hole. One RBF is computed for the full surface. Hence, the complexity of the algorithm is a function of n. That is, in cases where the surface is large and the hole boundary is small, the algorithm is inefficient. To overcome this problem, Branch et al. [6] extend this approach to use local RBF for each hole. Chen et al. [8] use an RBF-based approach to fill holes and recover sharp features in the hole area.

Tekumalla and Cohen [40] propose an approach that fills the hole by repeatedly using moving least squares projection. The approach iteratively adds layers of triangles onto the boundary until the hole is filled. Zhao et al. [43] fill holes in a similar way. After finding an initial triangulation by iteratively adding layers to the boundary, the position of the vertices is optimized by solving a Poisson equation. The goal of the optimization is to achieve smooth normal changes across the mesh.

Pernot et al. [31] propose a non-iterative approach that proceeds in three steps. First, the hole contour is cleaned by removing triangles with large aspect ratio. Second, the hole is filled with a triangulated ellipse deformed to touch the boundary. Third, the interior vertices of the triangulated ellipse are deformed to minimize an approximate curvature variation with the surrounding mesh. The main disadvantage of this method is that degenerate triangles may occur if the shape of the hole is far from elliptical.

Lévy [26] proposes a general technique for surface editing based on global parameterization. The method can fill holes in a surface by parameterizing the surface in the plane, filling the hole in the parameter domain, and placing the added vertex coordinates in three dimensions to approximate a MES. Unlike the previously discussed approaches, this approach can fill holes with arbitrary boundaries. However, the complexity of the algorithm is a function of n. That is, in cases where the surface is large and the hole boundary is small, the algorithm is inefficient. Furthermore, it is hard to ensure that no global overlap occurs during the parameterization of an incomplete mesh. If the parameterization overlaps, then the holes cannot be filled.

In this paper, we propose a novel approach to fill holes in triangular meshes. Our ap-

proach is similar to the approach by Lévy [26] in that we construct a planar parameterization. However, unlike Lévy, we do not parameterize the full mesh in the plane, but only the boundary of the hole. Note that this restricts our algorithm to operate on holes bounded by simple loops. However, if the triangular mesh and the boundaries of the holes are given, our algorithm is independent of the complexity n of the mesh and depends primarily on the complexity m of the hole boundary. If the boundaries of the holes are not given, our algorithm can extract the boundaries in $O(n)$ time. Furthermore, if our algorithm finds a solution, then the parameterization of the boundary is not self-intersecting.

This parameterization step decouples the topology of the filled hole from its geometry. After the parameterization is found, we embed the triangular path filling the hole using a minimal energy surface. One of the energy minimizations demonstrated in this paper is based on multi-view stereo. That is, we demonstrate that 2D images can be used as geometry recording data complementary to the range data. Since we can take digital images with much more flexible viewpoint selections, we can record those areas of the object that are difficult for the range sensors to reach.

A number of authors have used 2D images to enhance the quality of the 3D models generated from range scans. Dias et al. [12] fuse stereo reconstruction with 3D points obtained from range sensors. Abdelhafiz et al. [1] and El-Hakim et al. [14] combine range image with photogrammetric reconstructions. The vision-based approach of Dias et al. needs points from range images to be close to the reconstructed stereo points. Therefore it does not work well with large holes. The photogrammetric approach needs a lot of manual interactions. Two techniques based on shape-from-shading have been proposed. Xu et al. [42] use 2D images captured for textures to fill holes in a shape-from-shading scheme. They learn the surface normal from the existing mesh geometry. Recently, Panchetti et al. [30] propose an extension of the approach by Pernot et al. [31] that uses a shape-from-shading method based on a single photograph of the area covering the hole to position the interior vertices of the triangulated ellipse found by Pernot et al. [31]. However, the single-view approach to reconstruction has intrinsic limitations in handling non-Lambertian surface and difficult lighting conditions. To overcome these problems, we propose the use of multi-view stereo to infer the geometry of the missing surface.

There is an extensive literature for multi-view stereo. Seitz et al. [35] give a recent survey. Our work is closely related to the surface-based approaches such as in [13,15,16]. We use a deformation energy very similar to the one used for the surface deformation step of the recent work by Hiep et al. [19]. Unlike previous multi-view stereo approaches, we explicitly use boundary conditions.

3. Overview

Given a triangular manifold S with n vertices with partially missing data, we aim to fill the holes of S by triangular manifold meshes of minimum energy.

We first identify the boundaries of holes of S. Since S is a manifold, we can find the edges of S bounding a hole as edges that touch exactly one triangle. We fill each hole separately. Filling a hole bounded by m edges with a triangulation that does not have self-

intersections may require an exponential number of Steiner points in m [18]. Furthermore, the problem of deciding whether such a triangulation exists is an NP-complete problem [3]. Hence, we do not require that the mesh avoids self-intersection. However, unlike the approach by Barequet and Sharir [4], we add Steiner points to the mesh to reduce the occurrence of self-intersections.

Our approach proceeds in three steps. First, the boundary loop is gradually unfolded to a simple planar polygon. During this unfolding, the loop does not self-intersect. Second, the simple planar polygon is triangulated using a constrained Delaunay triangulation algorithm. Third, the triangulated patch is embedded in $\mathbb{R}^3$ using the known positions of the boundary vertices, refined to match the resolution of the surrounding mesh, and the interior deformed to approximate a MES. During this step, any suitable energy function can be used.

We demonstrate three examples of suitable energy functions. Two smoothly interpolate the geometry around the hole boundary. We also demonstrate the use of side information with a stereo-matching energy, aiming to embed the triangulated patch to optimally match a given set of calibrated input images. The following sections give a detailed description of these steps.

4. Unfolding the Boundary

We aim to unfold a boundary loop in $\mathbb{R}^3$ into a simple planar polygon with similar curvature as the boundary loop by gradually moving the vertices of the loop without causing any self-intersections of the loop. This is only possible if the original loop is an unknot. Hence, in the following, we assume that an unknot boundary loop is given.

The aim is to unfold the loop to a planar polygon. Minimizing an energy that encourages all sets of four vertices on the loop to be planar is computationally expensive because it takes $O(m^4)$ time to evaluate the energy. Hence, we minimize an energy that encourages all sets of four consecutive vertices on the loop to be planar. This energy can be evaluated in $O(m)$ time. As we unfold the curve gradually, we expect that the resulting planar curve has similar curvature as the boundary loop. In particular, we expect that concavities of the curve are preserved. Maintaining concavities helps to obtain a triangulation that does not self-intersect.

Let $p_0, p_1, \ldots, p_{m-1}$ denote the m vertices of the boundary loop. To unfold the boundary loop, we aim to minimize $E_{PE} = \sum_{i=0}^{m-1} d_{PE}(p_i)$ subject to the constraint $d_{MD} > \epsilon$ for an arbitrary threshold ϵ. We specify these terms below:

$$d_{PE}(p_i) = \angle(n_i, n_{(i+1) mod\ m}), \text{ where}$$

$$n_i = (p_i - p_{(i+1) mod\ m}) \times (p_{(i+2) mod\ m} - p_{(i+1) mod\ m})$$

and $\angle(a, b)$ denotes the angle between the two vectors a and b. An illustration of $d_{PE}(p_i)$ is shown in Figure 1. Here, d_{MD} denotes the minimum distance between any two segments on the boundary loop, which can be computed in $O(m^2)$ time. The minimum distance between two segments can be computed using dot products [25].

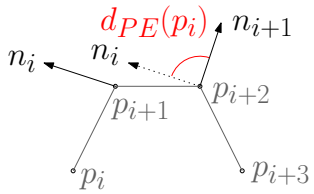


Fig. 1. *Illustration of the distance $d_{PE}(p_i)$. In the illustration, we assume that $i + 3 < m$. Otherwise, all subscripts need to be taken modulo m.*

Note that we can compute the gradient of E_{PE} analytically, which allows to minimize E_{PE} using a gradient descent approach. However, experimental evidence suggests that such an approach is likely to get trapped in a local minimum.

Therefore, we solve the minimization problem using simulated annealing [23], which proceeds by trying random steps. If the new solution achieves lower energy than the previous one, the step is always accepted. Otherwise, the step is accepted according to a probability distribution that depends on the time that elapsed since the algorithm was started. In the beginning of the algorithm, steps that increase the energy are more likely to be accepted than in the later stages of the algorithm. This is analogous to the way liquids cool and crystallize. We use the approach outlined by Press et al. [34] that uses the Boltzmann distribution to decide whether a step is accepted or rejected. This approach accepts a new step with probability $exp(\frac{-(E_{PE}^{t+1}-E_{PE}^{t}}{kT})$, where E_{PE}^{t} is the energy in the last step, E_{PE}^{t+1} is the energy in the current step, $k < 1$ is a constant that describes the rate of cooling, and T is the start temperature. We enforce the constraint $d_{MD} > \epsilon$ by restricting the maximum step size of the algorithm to $\max(\epsilon, d_{MD})$.

As outlined above, a solution that achieves lower energy than the previous one is always accepted. In this case, we choose the next random step close to the current one. Namely, we move each point along a direction within 10 degrees of the previous random direction.

After the simulated annealing step, we force the boundary loop to be planar by projecting it to its best-fitting plane. If it is possible to move the boundary loop to the projection by linear motions of the vertices without introducing self-intersections, we accept the unfolding. Otherwise, we restart the simulated annealing process. If we cannot find a solution after starting the simulated annealing process 100 times, we consider the algorithm inappropriate to fill the hole.

5. Triangulating the Planar Patch

After unfolding the boundary as outlined in the previous section, we aim to triangulate the simple planar polygon. We use the method and available code by Shewchuck [37] to complete this step. The algorithm computes a constrained Delaunay triangulation of the input polygon. A constrained triangulation of a polygon is a triangulation that is constrained to contain each of the edges of the polygon. That is, no Steiner points are added along the edges of the polygon. The constrained Delaunay triangulation of a polygon has the property that it maximizes the minimum angle over all constrained triangulations of the

polygon. During the triangulation, we do not add Steiner points.

6. Embedding the Triangular Mesh in $\mathbb{R}^3$

The previous section constructs a planar triangular mesh. This section outlines how we move this mesh to the boundary of the hole to obtain a watertight model.

First, we embed the mesh in $\mathbb{R}^3$ by moving the vertices of the mesh to their corresponding vertices on the hole boundary. As the order of the vertices along the boundary polygon is maintained during unfolding, this yields a watertight model, and, as we expect concavities to be preserved during the gradual unfolding, we expect to obtain a triangulation that does not self-intersect. For the non-pathological examples shown in our experiments, no self-intersections occur.

The result of this step is similar to the result by Barequet and Sharir [4]. Our approach takes $O(m^2cd)$ time to compute this result, where c is the number of unfolding steps during each SA run and d is the number of SA runs required to find the result, while the approach by Barequet and Sharir takes $O(m^3)$ time. The number of simulated annealing steps counts how often we need to restart the simulated annealing process while the number of unfolding steps counts the number of random steps taken in one simulated annealing step.

The resolution of the filling mesh may be different from the resolution of the surrounding mesh, so we refine the mesh using the approach by Chew [9]. Steiner points are added such that the Delaunay triangulation of the added points is guaranteed to have all the angles between 30° and 120° and where the edge lengths are at most twice as long as the edges of the mesh surrounding the hole. The running time of the algorithm is linear in the number of generated triangles.

Finally, we embed the interior vertices P_{Int} of the mesh, such that P_{Int} minimize an energy function. In this paper, we use three energy functions: a Laplacian energy to obtain a least-squares mesh [39], a discrete fairing energy [24], and a photoconsistency energy based on images of the objects. We first review the two simple Laplace based energies and then turn our attention to the newly proposed stereo-based energy. Note that these energies can be replaced by any desired energy function. For example, different boundary conditions can be used, as could different smoothing energies such as the Willmore energy [5].

6.1. *Laplacian Energy*

To obtain a MES using the Laplacian energy, the newly added interior vertices of the mesh filling the hole are repositioned to minimize the area of the triangular mesh subject to the boundary constraints provided by the positions of the boundary vertices. This can be achieved by repeatedly applying Laplacian smoothing because Laplacian smoothing is equivalent to minimizing the surface area [20].

To improve the efficiency of the algorithm, we formulate Laplacian smoothing as an optimization problem. For each vertex p of the mesh, define

$$U(p) = \frac{1}{|N_1(p)|} \sum_{q \in N_1(p)} q - p,$$

where $N_1(p)$ is the 1-ring neighborhood of p and $|N_1(p)|$ is the cardinality of the set $N_1(p)$. We aim to minimize

$$E_L = \sum_{p \in P_{Int}} \left(U(p)\right)^2,$$

where P_{Int} denotes the set of interior vertices of the mesh.

We can write E_L as a matrix product and compute the gradient of E_L with respect to all of the interior vertices explicitly. Let S denote a $k \times k$ matrix with entries

$$S_{i,j} = \begin{cases} -1 & \text{if } i = j, \\ \frac{1}{|N_1(p)|} & \text{if } (i,j) \in E, \\ 0 & \text{otherwise,} \end{cases}$$

where k denotes the number of vertices in P_{Int} and E denotes the edge set of the triangular patch. Let $X = [p_0 p_1 \dots p_{k-1}]^T$ denote the $k \times 3$ matrix containing the coordinates of P_{INT}. Then, $E_L = tr((SX)(SX)^T)$, where $tr(X)$ denotes the trace of the matrix X. Hence, the gradient of E_L with respect to X is $\nabla E_L = 2S^T SX$. This allows us to solve the optimization problem using a quasi-Newton method [29].

Note that minimizing E_L is equivalent to computing a least-squares mesh [39].

6.2. *Discrete Fairing*

To obtain a MES using discrete fairing, we minimize a second-order Laplacian energy. For each vertex p of the mesh, we compute $U(p)$ as before. Next, compute for each interior vertex of the mesh

$$U^2(p) = \frac{1}{|N_1(p)|} \sum_{q \in N_1(p)} U(q) - U(p).$$

We aim to minimize the energy

$$E_{DF} = \sum_{p \in P_{Int}} \left(U^2(p)\right)^2.$$

As before, we can express E_{DF} in matrix form as $E_{DF} = tr((SSX)(SSX)^T)$ and compute the gradient $\nabla E_{DF} = 2(SS)^T(SS)X$. Note that in this case, the matrices S and X contain information about all the points in P_{INT} and about all neighbors of P_{INT} in the original mesh. Hence, the dimensions of the matrices are greater than k. We minimize this energy using a quasi-Newton method [29].

6.3. *Photoconsistency*

This section describes how we reposition the interior vertices P_{Int} using a multi-view stereo method. The main contribution of this section is using the hole boundary information given by the laser range data to formulate the multi-view stereo problem as a boundary value problem for constrained surface deformation under photoconsistency energy terms.

The input to the deformation algorithm is a watertight surface consisting of the scan S with the holes filled using the initial mesh P as described above and a set of k input images $I_1, \ldots, I_k$ viewing the region in which S is incomplete. We further assume that the calibrations of $I_1, \ldots, I_k$ are known. The surface P is deformed with the goal to minimize the matching cost over all of the images viewing P.

We take our matching cost from Pons et al. [33]. The matching term for images I_i and I_j is computed by reprojecting the matching image I_j into camera i and measuring the dissimilarity of the two. That is, for the surface P

$$\mathcal{M}_{ij}(P) = M|_{\Omega_i \cap \Pi_i(P_j)} \left(I_i, I_j \circ \Pi_j \circ \Pi_{i,P}^{-1} \right)$$

where M is a dissimilarity measure, Ω_i is the domain of I_i, P_j is the portion of P that is visible in I_j, Π_i is the perspective projection calibrated for camera i, and $\Pi_{i,P}^{-1}$ is the reprojection from camera i onto P. The final term, $I_j \circ \Pi_j \circ \Pi_{i,P}^{-1}$, is the image predicted by reprojecting I_j into camera i via the surface P. Henceforth, we will denote the predicted reprojection of the matching image by $\tilde{I}$. We use as dissimilarity measure the sum of the squared differences over a small window surrounding each pixel location. Pons et al. derive the gradient $\nabla \mathcal{M}_{ij}(P)(p)$ of the matching term for a location p on P as

$$\nabla \mathcal{M}_{ij} = -\delta_{P_i \cap P_j}(p) \left[\partial_2 M(p_i) \, DI_j(p_j) \, D\Pi_j(p) \, \frac{d_i}{z_i^3} \right] n$$

where $\partial_2 M(p_i)$ is the derivative of the dissimilarity measure with respect to its second argument, DI_j and $D\Pi_j$ indicate the Jacobian matrices of the respective functions, d_i and z_i are the displacement and depth relative to camera i, and n is the outward surface normal. The Kronecker delta $\delta_{P_i \cap P_j}$ maintains that the gradient is zero in regions not visible from both cameras.

The matching cost of P over all images viewing P is $E_{PC} = \sum_i \sum_j \mathcal{M}_{ij}(P)$, where the indices $1 \leq i, j \leq k$. Hence, $\nabla E_{PC} = \sum_i \sum_j \nabla \mathcal{M}_{ij}(P)(p)$.

During the minimization, we restrict the boundary vertices of P not to move. Since we know the derivative of the cost function in closed form, we can formulate the problem as a boundary value problem and solve it using a quasi-Newton method. The method we use to minimize the is the cost function is the *limited-memory Broyden-Fletcher-Goldfarb-Shanno* (LSBFGS) scheme [29].

We efficiently implemented the calculation of the predicted image, the matching cost, and the matching gradient using GPU programming. In our implementation, we use nearly minimal data transfer between GPU and CPU to implement an efficient interface between computing the matching cost and using a FORTRAN LBFGS solver [44].

To avoid noise in the result, we perform 100 diffusion steps after the algorithm terminates.

7. Experiments

This section presents experiments using the algorithm presented in this paper. The experiments were conducted using an implementation in C++ using OpenMP on an Intel

Duo-Core 2.4GHz with 3GB of RAM and an NVIDIA GeForce 7300LE GPU. We set $k = 0.9, T = 0.5$, and $\epsilon = 10^{-5}$ in all of our experiments. In all of the figures showing holes in the models, back faces are shown in blue.

Although the results vary due to the random component in the unfolding step, we found the variation to be small. The reason is that the energy minimization converges to the same minimum if started from similar starting positions.

First, we demonstrate the applicability of the curve unfolding technique to highly curved boundaries. Second, we demonstrate the advantages of embedding the triangular patch to optimally match a given set of calibrated input images.

7.1. *Unfolding Technique*

This section focuses on the applicability of the unfolding technique for hole filling applications. When embedding the meshes in $\mathbb{R}^3$, we only use the simple Laplace based energies discussed above.

We first apply the algorithm to holes arising from the limitations of range scanners. The first experiment fills the holes present in the scan of a chicken model. The model was scanned using a ShapeGrabber laser range scanner. The scanned model contains 135233 vertices. The algorithm filled eight holes with a total of 666 vertices on the boundaries. The model along with the filled holes is shown in Figure 2. The model is grey and the filled holes are colored. Figures 2 (a) and (b) show the front and back of the chicken with five and three holes respectively. The most complex of the five holes on the front is located underneath the little chicken and shown in detail in Figure 2 (c). Another complex hole on the front of the chicken is located under the eye and shown in detail in Figure 2(d). Figures 2 (e) and (f) show large complex holes at the back base of the chicken model. Our algorithm took the longest to fill the two highly curved holes shown in Figure 2 (f). We can see that the proposed algorithm fills all of the holes present in the scan with approximate MES of similar resolution as the surrounding mesh.

The second experiment fills the holes present in a scanned model of a kitten available at the Aim@Shape data base [a]. The model has genus one and contains 135276 vertices. The algorithm filled 11 holes with a total of 856 vertices on the boundaries. The model along with the filled holes in shown in Figure 3. As before, the model is shown in grey and the filled holes are colored. Note that the proposed algorithm is independent of the genus of the model and can be used to fill holes in models of arbitrary genus.

[a] http://shapes.aimatshape.net/releases.php

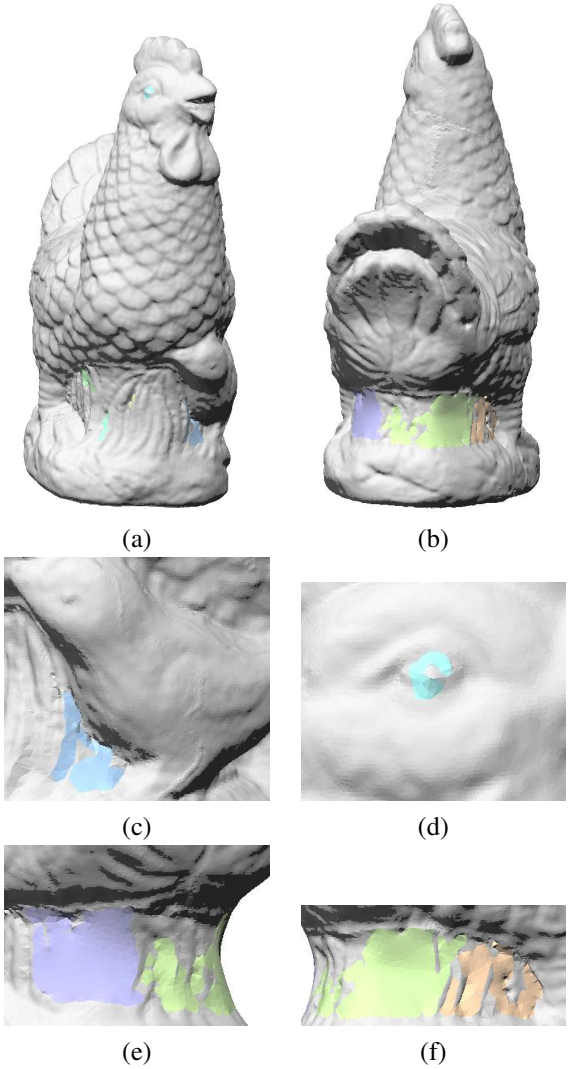


Fig. 2. *Chicken model filled by minimizing E_L. (a): Front view of the chicken with five filled holes. (b): Back view of the chicken with three filled holes. (c): Detail view of the filled hole under the little chicken. (d): Detail view of the filled hole under the eye. (e): Detail view of a filled large hole at the base of the model. (f): Detail views of two filled complex holes at the base of the model.*

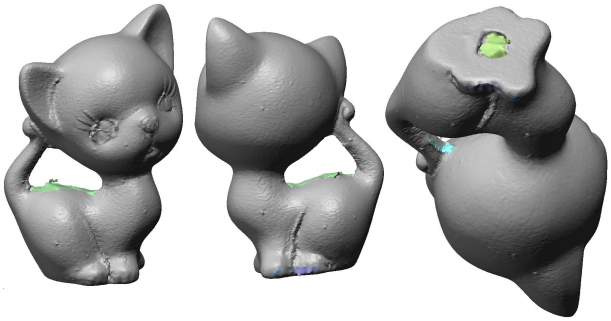
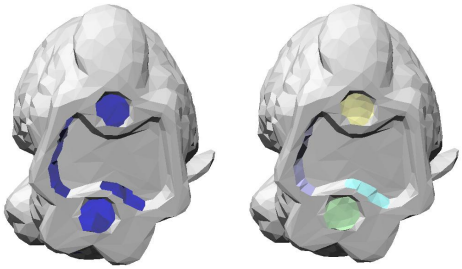


Fig. 3. *Kitten model filled by minimizing* E_{DF}.

The third experiment based on laser range scans fills the well-known holes present in the Stanford bunny [41]. There are 5 holes and they contain 79 vertices total. We fill the holes while minimizing E_L. Four of the holes before and after filling are shown in Figure 4.



(a) (b)

Fig. 4. *Stanford bunny model filled by minimizing* E_L. *(a): Four of the holes in the bunny model. (b): Filled holes.*

We next apply the algorithm to a number of artificial holes. We created holes of large curvature in complete models to show the applicability of our algorithm in this case. The first experiment fills the hole present in the armpit of a human model. The hole boundary has high curvature. The initial hole as well as the result of our algorithm are shown in Figure 5.

The following three experiments fill holes present in the head of a human model. The first hole is shown in Figure 6. The hole is large and the hole boundary is highly curved. Nonetheless, our algorithm finds a visually pleasing solution.

The second hole is shown in Figure 7. The hole is large. The hole boundary is highly curved and highly twisted. Nonetheless, our algorithm finds a visually pleasing solution.

The third hole is shown in Figure 8. The hole is large. The hole boundary is curved in all three dimensions. Nonetheless, our algorithm finds a visually pleasing solution.

Finally, we fill an artificial hole in the head of an alien model from the Princeton Shape Benchmark [38] shown in Figure 9. The hole covers an area where the original model con-

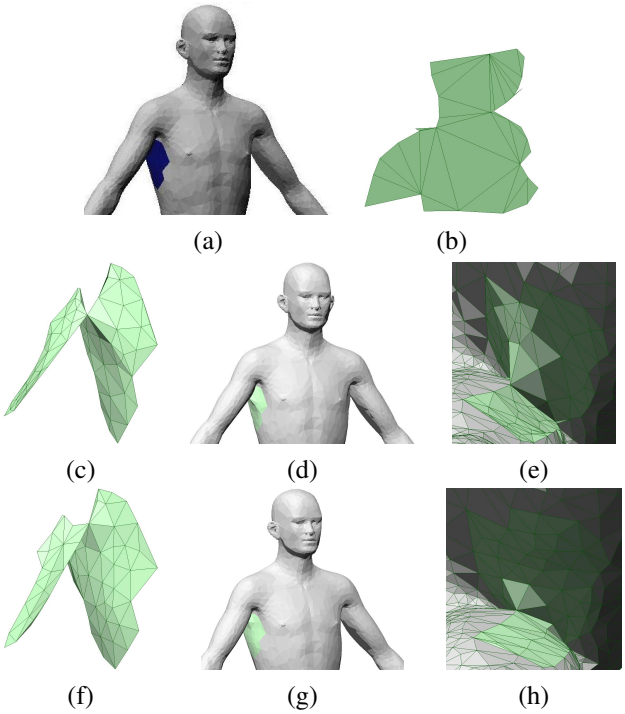


Fig. 5. *Armpit model. (a): Hole in the armpit of a human model. (b): Unfolded mesh. (c)-(e): Final embedded mesh obtained by minimizing E_L. (f)-(h): Final embedded mesh obtained by minimizing E_{DF}.*

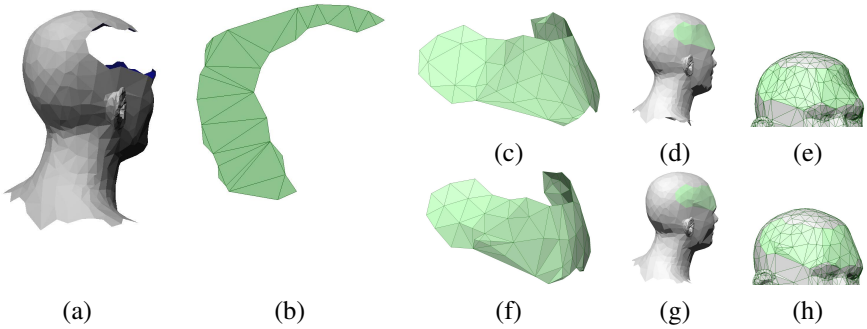


Fig. 6. *Head model 1. (a): Hole in the head of a human model. (b): Unfolded mesh. (c)-(e): Final embedded mesh obtained by minimizing E_L. (f)-(h): Final embedded mesh obtained by minimizing E_{DF}.*

tains a crease. Note that the crease is recovered by the hole filling algorithm if E_{DF} is used.

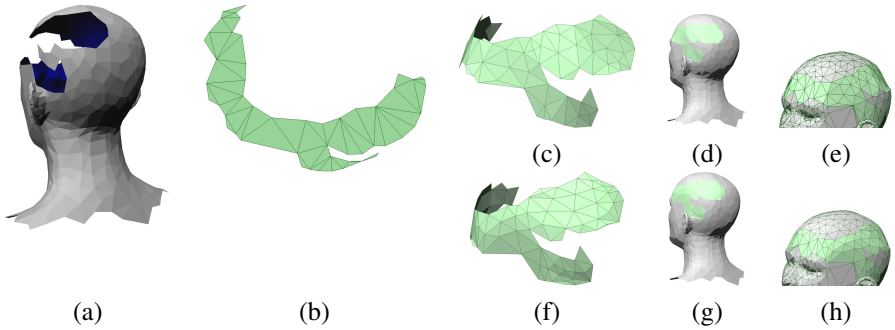


Fig. 7. *Head model 2. (a): Hole in the head of a human model. (b): Unfolded mesh. (c)-(e): Final embedded mesh obtained by minimizing E_L. (f)-(h): Final embedded mesh obtained by minimizing E_{DF}.*

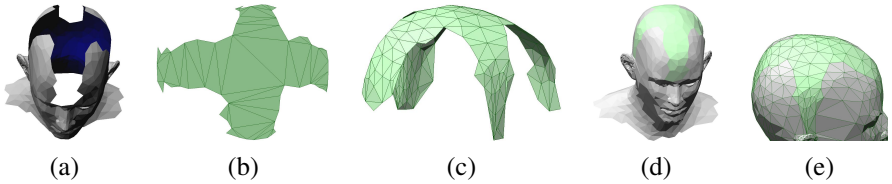


Fig. 8. *Head model 3 filled by minimizing E_{DF}. (a): Hole in the head of a human model. (b): Unfolded mesh. (c): Final embedded mesh. (d): Filled hole. (e): Detail view of the filled hole.*

Fig. 9. *Alien model filled by minimizing E_{DF}.*

7.2. *Stereo-based Embedding Energy*

This section focuses on using the stereo-based embedding energy detailed in Section 6.3 to reconstruct the geometry of the missing surface.

We first apply the algorithm to an artificial hole in the Armadillo data set. The original model is from the AIM@SHAPE repository [b]. The mesh contains 171011 vertices and the artificial hole contains 190 vertices on the boundary. We applied a striped texture to the model and captured six synthetic images of the complete model before cutting the hole. One of the images is shown in Figure 10. The model with filled holes is shown in

[b]http://shapes.aimatshape.net/releases.php

Figure 11. The figure shows the model in grey and the filled holes minimizing E_{PC}, E_L, and E_{DF}, respectively, both in grey and using a color coding. For each point on the filled hole, we compute the signed Euclidean distance to the nearest vertex on the ground truth model. A distance of zero is shown in green. For points outside the ground truth surface, the color is linearly interpolated from green to red as the distance increases. For points inside the ground truth surface, the color is linearly interpolated from green to blue as the distance increases. Table 1 gives statistics of the unsigned error. From both the figure and the table, we can see that the filled hole minimizing E_{PC} is the most accurate when texture is used. When no texture is used when creating the images, E_L outperforms E_{PC}. This shows that meaningful texture information is crucial for this method.

Fig. 10. An input image for the armadillo experiment.

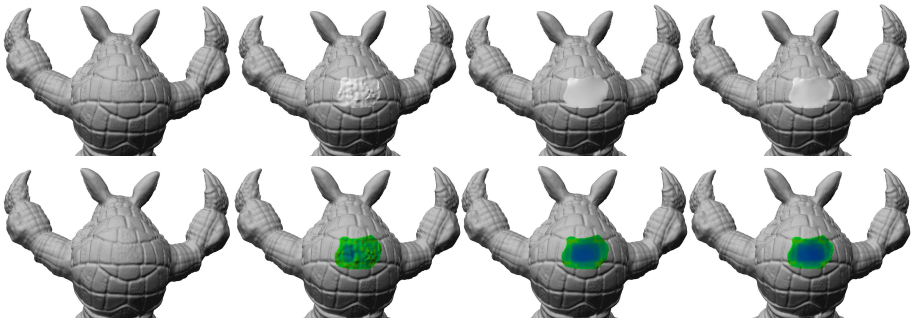


Fig. 11. Armadillo model. Top shows geometry of the models and bottom shows errors using color coding. Ground truth is shown on the left. The three images on the right show the filled hole using E_{PC}(striped texture), E_L, and E_{DF} (from left to right).

Second, we tested the algorithm using the scanned surface of the chicken shown in Figure 2. The model was scanned using a ShapeGrabber laser range scanner. We obtained 11 images of the model using a Canon Powershot A520 4 mega-pixel camera. We calibrated the images by manually selecting features on the model and in the images and computing the projection matrix using the direct linear transform (DLT) algorithm [17] . The result is

Embedding	Min squared error	Max squared error	Mean squared error
E_{PC} with striped texture	0.0	1.51	0.31
E_{PC} without texture	0.0	2.06	0.60
E_L	0.0	1.83	0.52
E_{DF}	0.0	1.99	0.89

Table 1. Error statistics for the armadillo model.

shown in Figure 12. The detail view on the right of Figure 12 shows ridges recovered from the information provided by multi-view stereo.

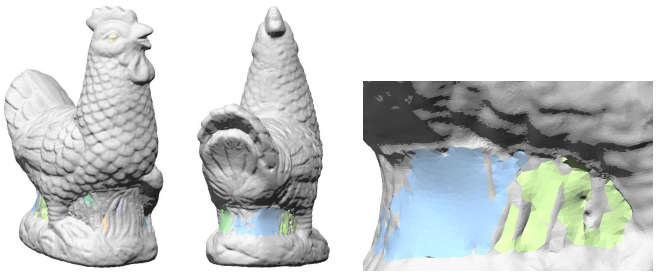


Fig. 12. *Chicken model filled by minimizing E_{PC}.*

7.3. *Running times*

This section gives the running times of the newly presented algorithm. The running times of the experiments are given in Table 2. We average the running time over 10 runs. Note that due to the random component in simulated annealing, the running times of the unfolding step during the 10 runs vary significantly. The running time of the unfolding step depends on the number of times simulated annealing is restarted and on the number of steps required to unfold the boundary. The running time of the embedding step depends on the number of Steiner points added to the triangular mesh that fills the hole. When E_{PC} is minimized, the running time of the embedding step also depends on the number and dimensions of the input images.

The efficiency of the unfolding step may be improved by using a more sophisticated simulated annealing technique than the one described by Press et al. [34] .

7.4. *Comparison to ReMESH*

This section compares the results obtained using our approach to the results obtained using ReMESH [2]. ReMESH implements the algorithm proposed by Liepa [28], which first triangulates the polygonal hole boundary and then performs a surface fairing step. Figure 13

Model	n	h	m	d	U	$T(E_L)$	$T(E_{DF})$	k	$T(E_{PC})$
Armpit	9951	1	35	4.8	3	< 1	< 1	-	-
Head 1	1177	1	33	2.0	1	< 1	< 1	-	-
Head 2	1189	1	53	2.9	2	< 1	< 1	-	-
Head 3	1157	1	63	16.4	2	1	1	-	-
Bunny	1494	4	79	1.0	1	1	1	-	-
Alien	6815	1	26	1.0	1	< 1	< 1	-	-
Armadillo	171011	1	190	1.0	6	20	73	6	86
Chicken	135233	8	666	8.4	344	6	156	11	4648

Table 2. *Running times in seconds. The number of vertices in the model is denoted by n, the number of holes in the model is denoted by h, the number of vertices on the hole boundary is denoted by m, the number of simulated annealing steps is denoted by d, and the number of input images is denoted by k. The running times are denoted as follows: U denotes the total time required to unfold the holes, $T(E_L)$ denotes the embedding time when E_L is minimized, $T(E_{DF})$ denotes the embedding time when E_{DF} is minimized, and $T(E_{PC})$ denotes the embedding time when E_{PC} is minimized. The times for the armadillo model are achieved when the striped texture is used. All unfolding times are averaged over 10 runs. For models with multiple holes, d is averaged over all holes.*

shows the results of using ReMESH to fill the holes in the chicken model. The mesh is shown in grey and the filled holes are shown in green. Note that the results are of similar quality as the ones obtained by our method shown in Figure 2. For most models presented in this paper, ReMESH obtains similar results as our method when E_{DF} is minimized. ReMESH fails for the kitten model. ReMESH is currently faster than our method, since our implementation of the unfolding step is not optimized. However, our method has lower asymptotic complexity.

Furthermore, unlike our method, ReMESH is purely geometry-based and cannot accommodate energy functions that take side information, such as images, into account.

Fig. 13. Holes in chicken scan filled by ReMESH.

8. Conclusion

We presented a novel approach to automatically fill holes in triangulated models. The approach fills the hole using a minimum energy surface that is obtained by unfolding the hole boundary into the plane using an energy minimization approach. The planar curve is then triangulated and embedded to the three-dimensional position of the boundary loop. In this paper, we embed the triangular patch as a minimal surface. We propose a new stereo-based energy that aims to optimally match the triangular patch to a given set of input images. Note that this could be replaced by a prior distribution of the surface's geometry to embed the triangular patch. We leave this for future work.

The energy used to unfold the boundary loop encourages all sets of four consecutive vertices on the loop to be planar. This energy can be evaluated efficiently in $O(m)$ time. We use simulated annealing to minimize this energy. We leave applying more sophisticated SA variants to this problem for future work.

Finally, we summarize some limitations of our approach:

- The approach can only fill holes bounded by a simple unknot boundary. That is, holes with islands and holes that touch in a point cannot be filled using this approach.
- The approach unfolds the hole boundary using simulated annealing. This method is not guaranteed to find a solution and fails for highly twisted curves. This type of curve is unusual in practical applications.

Acknowledgments

The authors would like to thank Dr. Patrick Morin and Dr. Michiel Smid of the School of Computer Science, Carleton University, for the use of the laser range scanner. This research is supported in part by OGS.

References

1. Ahmed Abdelhafiz, Björn Riedel, and Wolfgang Niemeier. Towards a 3D true colored space by the fusion of laser scanner point cloud and digital photos. In *Proceedings of the ISPRS Working Group V/4 Workshop (3D-ARCH 2005)*, 2005.
2. Marco Attene and Bianca Falcidieno. Remesh: An interactive environment to edit and repair triangle meshes. In *SMI '06: Proceedings of the IEEE International Conference on Shape Modeling and Applications 2006*, page 41, 2006.
3. Gill Barequet, Matthew Dickerson, and David Eppstein. On triangulating three-dimensional polygons. In *Symposium on Computational Geometry*, pages 38–47, 1996.
4. Gill Barequet and Micha Sharir. Filling gaps in the boundary of a polyhedron. *Computer Aided Geometric Design*, 12(2):207–229, 1995.
5. Alexander Bobenko and Peter Schröder. Discrete willmore flow. In *In Eurographics Symposium on Geometry Processing*, pages 101–110, 2005.
6. John Branch, Flavio Prieto, and Pierre Boulanger. Automatic hole-filling of triangular meshes using local radial basis function. In *Third International Symposium on 3D Data Processing, Visualization and Transmission (3DPVT'06)*, pages 727–734, 2006.
7. Jonathan C. Carr, Richard K. Beatson, Jon B. Cherrie, Tim J. Mitchell, Richard W. Fright,

Bruce C. Mccallum, and Tim R. Evans. Reconstruction and representation of 3d objects with radial basis functions. In *ACM Transactions on Graphics (SIGGRAPH'01)*, 2001.

8. Chun-Yen Chen, Kuo-Young Cheng, and Hong-Yuan Mark Liao. Fairing of polygon meshes via bayesian discriminant analysis. In *International Conference in Central Europe on Computer Graphics, Visualization and Computer Vision*, 2004.
9. L. Paul Chew. Guaranteed-quality triangular meshes. Technical Report TR 89-983, Computer Science Department, Cornell University, 1989.
10. James Davis, Stephen R. Marschner, Matthew Garr, and Marc Levoy. Filling holes in complex surfaces using volumetric diffusion. In *First International Symposium on 3D Data Processing, Visualization and Transmission (3DPVT'02)*, pages 438–433, 2002.
11. Tamal K. Dey and Samrat Goswami. Tight cocone: A water-tight surface reconstructor. In *ACM Symposium on Solid Modeling and Applications 2003*, pages 127–134, 2003.
12. Paulo Dias, Vìtor Sequeira, Francisco Vaz, and Vo ao Gonçalves. Registration and fusion of intensity and range data for 3D modelling of real world scenes. In *Fourth International Conference on 3-D Digital Imaging and Modeling (3DIM'03)*, 2003.
13. Ye Duan, Liu Yang, Hong Qin, and Dimitris Samaras. Shape reconstruction from 3D and 2D data using PDE-based deformable surfaces. In *ECCV (3)*, pages 238–251, 2004.
14. Sabry F. El-Hakim, J.-Angelo Beraldin, Michel Picard, and Antonio Vettore. Effective 3D modeling of heritage sites. In *Fourth International Conference on 3-D Digital Imaging and Modeling (3DIM'03)*, pages 302–309, 2003.
15. Carlos H. Esteban and Francis Schmitt. A snake approach for high quality image-based 3D object modeling. In *2nd IEEE Workshop on Variational, Geometric and Level Set Methods in Computer Vision*, pages 241–248, Nice, France, 2003.
16. Olivier Faugeras and Renaud Keriven. Variational principles, surface evolution, PDE's, level set methods and the stereo problem. *IEEE Transactions on Image Processing*, 7(3):336–344, 1998.
17. Richard Hartley and Andrew Zisserman. *Multiple View Geometry in Computer Vision*. Cambridge University Press, 2000.
18. Joel Hass, Jack Snoeyink, and William P. Thurston. The size of spanning disks for polygonal curves. *Discrete and Computational Geometry*, 29(1):1–17, 2003.
19. Vu Hoang Hiep, Renaud Keriven, Jean-Philippe Pons, and Patrick Labatut. Towards high-resolution large-scale multi-view stereo. In *IEEE Conference on Pattern Recognition and Computer Vision (CVPR97)*, pages 1430–1437, 2009.
20. Klaus Hildebrandt and Konrad Polthier. Anisotropic filtering of non-linear surface features. *Computer Graphics Forum*, 23:391–400, 2004.
21. Tao Ju. Robust repair of polygonal models. In *ACM Transactions on Graphics (SIGGRAPH'04)*, pages 888–895, New York, NY, USA, 2004. ACM.
22. Yongtae Jun. A piecewise hole filling algorithm in reverse engineering. *Computer-Aided Design*, 37(2):263–270, 2005.
23. Scott Kirpatrick, Dan Gelatt, and Mario Vecchi. Optimization by simulated annealing. *Science*, 220(4598):671–680, 1983.
24. Leif Kobbelt, Swen Campagna, Jens Vorsatz, and Hans-Peter Seidel. Interactive multi-resolution modeling on arbitrary meshes. In *Conference on Computer graphics and interactive techniques (SIGGRAPH98)*, pages 105–114, 1998.
25. Andrew Ladd and Lydia Kavraki. Using motion planning for knot untangling. *The International Journal of Robotics Research*, 23(7–8):797–808, 2004.
26. Bruno Lévy. Dual domain extrapolation. In *ACM Transactions on Graphics (SIGGRAPH'03)*, pages 364–369, 2003.
27. Gen Li, Xiu-Zi Ye, and San-Yuan Zhang. An algorithm for filling complex holes in reverse engineering. *Engineering with Computers*, 24(2):119–125, 2008.
28. Peter Liepa. Filling holes in meshes. In *Eurographics Symposium on Geometry Processing*,

pages 200–205, 2003.
29. Dong C. Liu and Jorge Nocedal. On the limited memory method for large scale optimization. *Mathematical Programming*, 45:503–528, 1989.
30. Minica Panchetti, Jean-Philippe Pernot, and Philippe Véron. Towards recovery of complex shapes in meshes using digital images for reverse engineering applications. *Computer Aided Design*, Accepted for publication.
31. Jean-Philippe Pernot, George-Florin Moraru, and Philippe Véron. Repairing triangle meshes built from scanned point cloud. *Journal of Engineering Design*, 18(5):459–473, 2007.
32. Joshua Podolak and Szymon Rusinkiewicz. Atomic volumes for mesh completion. In *Eurographics Symposium on Geometry Processing*, 2005.
33. Jean-Philippe Pons, Renaud Keriven, and Olivier Faugeras. *Modelling Non-Rigid Dynamic Scenes from Multi-View Image Sequences, in Handbook of Mathematical Models in Computer Vision, N. Paragios, Y. Chen and O. Faugeras, Eds*, 2006.
34. William H. Press, Saul A. Teukolsky, William T. Vetterling, and Brian P. Flannery. *Numerical recipes in C (2nd ed.): the art of scientific computing*. Cambridge University Press, 1992.
35. Steven Seitz, Brian Curless, James Diebel, Daniel Scharstein, and Richard Szeliski. A comparison and evaluation of multi-view stereo reconstruction algorithms. In *Proc. Computer Vision and Pattern Recognition (CVPR 2006)*, 2006.
36. Andrei Sharf, Marc Alexa, and Daniel Cohen-Or. Context-based surface completion. *ACM Transactions on Graphics (SIGGRAPH'04)*, 23(3):878–887, 2004.
37. Jonathan R. Shewchuk. Triangle: Engineering a 2D quality mesh generator and Delaunay triangulator. In *First Workshop on Applied Computational Geometry*, pages 124–133, 1996.
38. Philip Shilane, Patrick Min, Michael Kazhdan, and Thomas Funkhouser. The princeton shape benchmark. In *Proceedings of Shape Modeling International*, 2004.
39. Olga Sorkine and Daniel Cohen-Or. Least-squares meshes. In *International Conference on Shape Modeling and Applications (SMI)*, pages 191–199, 2004.
40. Lavanya Sita Tekumalla and Elaine Cohen. Hole-filling algorithm for triangular meshes. Technical Report UUCS-04-019, School of Computing, University of Utah, 2004.
41. Greg Turk and Marc Levoy. Zippered polygon meshes from range images. In *Proceedings of SIGGRAPH'94*, pages 311–318, 1994.
42. Songhua Xu, Athinodoros Georghiades, Holly Rushmeier, Julie Dorsey, and Leonard McMillan. Image guided geometry inference. In *Third International Symposium on 3D Data Processing, Visualization and Transmission (3DPVT'06)*, 2006.
43. Wei Zhao, Shuming Gao, and Hongwei Lin. A robust hole-filling algorithm for triangular mesh. *The Visual Computer*, 23(12):987–997, 2007.
44. Ciyou Zhu, Richard H. Byrd, Peihuaung Lu, and Jorge Nocedal. Algorithm 778: L-BFGS-B, Fortran subroutines for large-scale bound constrained optimization. *ACM Trans. Mathematical Software*, 23(4):550–560, 1997.