Visual Comput (2006) 22: 168–180
DOI 10.1007/s00371-006-0374-y

ORIGINAL ARTICLE

Seyoun Park
Xiaohu Guo
Hayong Shin
Hong Qin

Surface completion for shape and appearance

Published online: 3 February 2006

S. Park (✉)
VMS Lab Dept of Industrial Engineering, Korea Advanced Institute of Science and Technology, Daejoen, South Korea
parksy@vmslab.kaist.ac.kr

X. Guo
Center of Visual Computing, Dept. of Computer Science, State University of New York at Stony Brook, USA
xguo@cs.sunysb.edu

H. Shin
Dept. of Industrial Engineering, Korea Advanced Institute of Science and Technology, Daejoen, South Korea
hyshin@kaist.ac.kr

H. Qin
Center of Visual Computing, Dept. of Computer Science, State University of New York at Stony Brook, USA
qin@cs.sunysb.edu

Abstract In this paper, we present a new surface content completion system that can effectively repair both shape and appearance from scanned, incomplete point set inputs. First, geometric holes can be robustly identified from noisy and defective data sets without the need for any normal or orientation information. The geometry and texture information of the holes can then be determined either automatically from the models' context, or interactively from users' selection. We use local parameterizations to align patches in order to extract their curvature-driven digital signature. After identifying the patch that most resembles each hole region, the geometry and texture information can be completed by warping the candidate region and gluing it onto the hole area. The displacement vector field for the exact alignment process is computed by solving a Poisson equation with boundary conditions. Our experiments show that the unified framework, founded upon the techniques of deformable models, local parameterization, and PDE modeling, can provide a robust and elegant solution for content completion of defective, complex point surfaces.

Keywords Hole filling · Active contour method · Poisson equation

1 Introduction

The ever-increasing popularity of data acquisition devices has made surface completion a critical step in the entire reverse engineering pipeline. Considering the different digitizing techniques that are currently available, many optical devices often produce defective data samples that are subject to a local absence of data. This incompleteness of scanned data inputs is mainly due to occlusions, low reflectance, or scanner placement constraints, etc. In certain digital information restoration applications (especially in archeology), the scanned object (e.g., antiques) itself may be incomplete and defective because some missing parts have been ruined during the long historic time span. In addition, some local surface editing processes may cause big holes unintentionally. In these cases, both shape and appearance (texture) information needs to be repaired in order to facilitate the downstream processing of the digital content. The surface completion task is a nontrivial process, which may often produce unsatisfactory results, and may require tedious human effort as well. In particular, if the hole is relatively bigger than a neighboring feature size, or there is a large missing part in the real object (e.g.,

ruined sculptures), surface completion would be difficult to achieve using only the boundary information of the holes.

Ideally, a set of desirable properties for a general-purpose surface content completion technique would include:

- Ability to deal with raw scanned point sets, without certain specific information (such as surface normals) or requirements (e.g., data accuracy or density).
- Robustness for defective and incomplete point sets.
- Automatic hole detection under the assumption that the surface is a closed manifold. If the surface is not a closed manifold, however, the hole-finding process should be semi-automatic under user guidance.
- Automatic repair for both the shape and appearance of the defective point sets.
- The holes should be filled in a way that is minimally distinguishable from their surrounding regions, both in shape geometry and appearance modeling. If the scanned model has enough context information, it should be automatically filled in the way that conforms with the natural properties of the model [24]. Otherwise it should be guided by the user.

In this paper, we present an automatic and interactive system which can repair both *shape* and *appearance* of defective point sets. The main idea of this system, inspired by [24], is to directly use the context information of the model itself, or other models' geometric and texture information to complete the defective point surfaces, via automatic cut-and-paste. We take as input a set of point samples (possibly noisy) that is assumed to be a closed manifold without any normal or orientation information. The point cloud is first pre-processed by a deformable-model-based region partition approach to determine the orientations of all the point samples, remove outliers and noise, and infer the in-and-out relationship, all in the same stage [31]. Towards this goal, we use an octree discretization of the original point cloud data to build its distance field. The holes in the original point set can be robustly detected using the method of active deformable models by seeking all the saddle points of the unsigned distance field, which will uniquely identify all the missing parts of the scanned data inputs. The automatic hole detection algorithm is more robust than the previous methods that were based on analyzing the point distribution in each octree cell. Then, we extend the key idea of context-based surface completion [24] (when the scanned model contains enough context information) to conduct model repair for both geometry and appearance.

Rather than analyzing the shape similarities based on the signed distance fields through volumetric embedding in [24], we propose to analyze both shape and appearance similarities solely based on the curvature and color information of the surface patches. Specifically, we devise a curvature-centered "digital signature" extracted from the surface geometry, and use it to identify one patch in the context of the scanned model that is most similar to the hole region. The curvature-driven shape signature is an intrinsic surface geometric quantity, and it can provide better performance for geometric shape comparison. Finally, we solve surface partial differential equations (Poisson equations) to acquire the "warped" shape and appearance of the hole region. In contrast to other volumetric PDE-based approaches, our surface PDE method is much more efficient and robust.

To the best of our knowledge, our work is the first attempt to repair both geometry and texture information in a single unified framework. This can be only achieved by using local parameterization, which can facilitate patch similarity comparison, patch alignment, and patch warping via a 2D Poisson solver. We perform local parameterization over the surface patches in each octree cell up to a certain layer under the condition that each octree cell contains at most one surface patch that is homeomorphic to a topological disc. After local parameterization, the patches can be brought into correspondence, which makes similarity comparison easier, and geometric and textural information can be warped by solving the Poisson equations over the surface patches. For complicated surface patches (or even higher genus local patches), however, local parameterization becomes infeasible. For some special cases where the repaired model does not have enough context information, or the automatic patch selection can not give a globally meaningful solution (such as the missing hand of the Santa Claus model in Fig. 1), our surface completion system allows user interaction in the whole surface completion process.

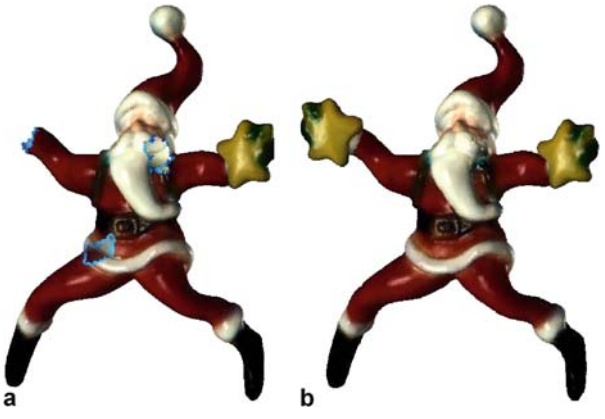

Fig. 1a,b. Shape and appearance repair of the Santa Claus model under user manipulation: **a** automatic hole detection from the defective input point set; **b** both shape and appearance of the holes are filled in a way that is minimally distinguishable from the model's context

2 Related work

Surface completion of the scanned point sets needs to be naturally integrated into a surface reconstruction algorithm. The original point samples can be interpolated using alpha shapes [2, 9], crusts [1], or balls [3]. These methods guarantee the interpolation quality under certain sampling conditions, e.g., in crusts the distance between two points should be less than the local feature size. [11] used the signed distance field to reconstruct the surface from point clouds. [18] used moving least-squares for interpolating, or smoothing scattered-data. [5] used globally supported radial basis functions (RBFs) to fit data points by solving a large dense linear system. [21] proposed multi-level partition of unity implicits for constructing a piecewise implicit surface from large-scale point clouds.

Another class of surface reconstruction methods for point clouds is called the active contour method (or deformable models) [14, 30, 31, 33], which has been exploited extensively in computer vision. Among their many advantages, deformable models are very robust in dealing with noisy data sets. In this paper, we also utilize the active contour method for robustly locating holes.

Surface completion can be also conducted by solving certain partial differential equations. [7] addressed the problem of hole filling via isotropic volumetric diffusion, which can handle geometrically and topologically complicated holes. [16] proposed a variation by using a quadric approximation of the signed distance function. [27] used implicit surfaces to fill the holes via geometric partial differential equations derived from image inpainting algorithms. [6] repaired the surfaces as an optimization process by minimizing the integral of the square mean curvature.

Most hole filling methods are volumetric methods that obtain the surface implicitly as the boundary between inside and outside volumetric regions. [13] constructed an inside/outside volume using an octree grid, and repaired the polygonal meshes by contouring the half-edge loops surrounding the holes. [20] decomposed the space into atomic volumes, and utilized the graph cuts to determine the individual volumes to be inside or outside. Volumetric methods are inappropriate for appearance repair since the texture is an intrinsic property of the surface that cannot be embedded in 3D volume. In this paper we use local parameterization to capture the textural information.

While all of the above methods create a smooth patch to cover the hole region, context-based approaches repair the holes according to their context information. [25] warps a given shape model towards the missing region of the given surface using control points, followed by a fairing step along the boundary of the hole. [24] can automatically choose a local patch that is geometrically similar to the hole region. [15] filled the holes using a mapping between the incomplete mesh and a template model. The completed models are guaranteed to have the same topology as the template. [19] retrieved context models from a database of 3D shapes, and warped the retrieved models to conform with the incomplete scanned data. Our approach in this paper is conceptually similar to [24] for automatic context-based hole filling. Nonetheless, different from the volumetric embedding approach taken in [24], our method is solely based on the intrinsic properties of the surface (curvatures and colors), in which case not only geometry, but also appearance information can be repaired automatically based on context information by only solving a two-manifold PDE system.

3 System overview

Our surface completion system is mainly composed of the following three separate parts:

1. Hole detector: given a set of noisy, defective point samples as input, this part utilizes a deformable-model-based method to determine the orientation of each point sample and remove noise. In the meantime, the holes in the original point cloud can be automatically detected (Sect. 4) in this stage.
2. Surface patch selector: this part finds the most similar surface patch to the detected hole based on the octree discretization. For automatic selection, a curvature-centered "digital signature" is computed for each local patch (Sect. 5.2), which can be subsequently employed for both geometry and appearance comparison in a quantitative way. If users need to select surface patches directly, Boolean operations are provided for a more accurate selection task.
3. Poisson solver: in this part, the best candidate surface patch and its corresponding boundary are aligned with the hole region using the colored ICP (iterative closest point) method (Sect. 6.1). Finally, its geometry and color information are warped one after the other to the hole region by solving Poisson equations (Sect. 6.2).

Figure 2 shows the general pipeline of our surface completion system. To achieve efficiency, three kinds of data structures are constructed as a pre-process. First, we embed an input point set in an octree structure. Also, the hierarchical volumetric grid structure is used to construct the distance field in Sect. 4. Finally, we utilize a kd-tree for the fast retrieval of neighboring points in our system.

4 Hole detection

We use the active contour method as a pre-processing stage for the original point set surface. Given a set of noisy, defective point samples without any normal or orientation information, the active contour method of [31] can be utilized to facilitate determining the orientation of

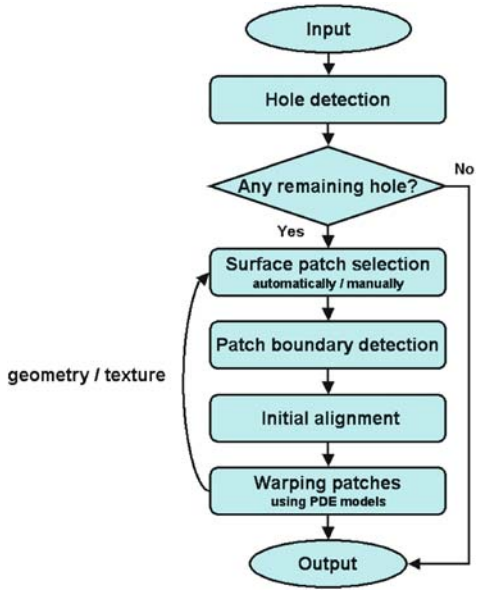

Fig. 2. The general framework and data pipeline of our surface completion system

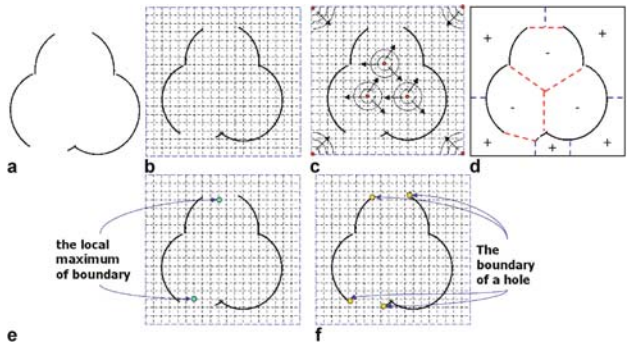

Fig. 3. Active contour method for automatic hole detection

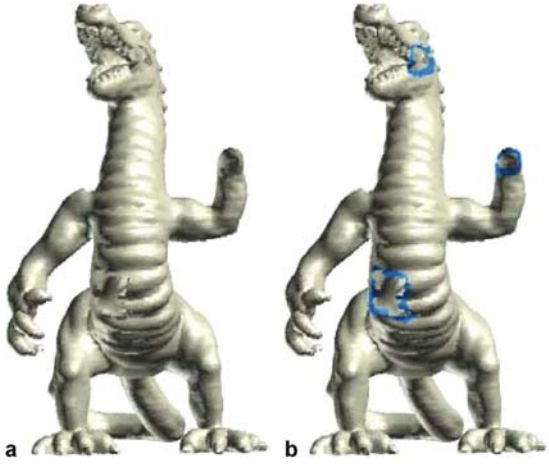

Fig. 4a,b. Automatic hole detection for the incomplete dragon model: **a** the original point set with three complicated holes; **b** the holes (in blue) are detected automatically

each point sample, removing outliers and noise. We will show in this paper that the active contour method can also be utilized for automatic hole detection. In this section, we will briefly introduce the active contour method and explain how to utilize it to find the holes. For more details on its technical settings and applications to orientation determination and removal of outliers, the reader is referred to [31].

If the input point set, Γ, is assumed to be a closed manifold, we can naturally divide the volumetric space into inside and outside regions. The active contour method is a commonly used approach to determine the inside or outside of a surface. To avoid leakage through holes, most deformable models resort to the minimization of certain strain energies. [31] takes a different approach by launching two active contours growing at both sides of the hole. The active contours travel at the same speed and keep the same distance to the surface, and they will finally collide at the center of the hole. For any spatial region visually bounded by a set of surface samples, there exists at least one space point inside the surface that has a local maximal unsigned distance field. Therefore, it is sufficient to launch an active contour seed at each local maximum point besides the one in the outer bounding space. Figure 3 shows the idea of the active contour method. We launch an active contour at each local maximum of the unsigned distance field, as well as in the outer bounding space (Fig. 3 (c)). The active contours grow and shrink toward the surface and try to keep the same distance to the surface, and at the end the whole surface is sandwiched between these active contours (Fig. 3 (d)). Please note that the hole region is sandwiched by an inner and an outer contour, and the center of the hole is a saddle point of the unsigned distance field (Fig. 3 (e)).

In our implementation, the continuous unsigned distance field is discretized onto a volumetric grid. Each grid point is associated with a distance value, which is defined as its block distance to the nearest nonempty cell. Each local maximum grid point initiates an active contour. All points on the active contours are sorted with a heap by their distance to the surface, and the furthest point grows first. This can guarantee that all parts of the active contours grow with nearly the same distance to the surface. When the active contours collide, we can check for the distance value associated with each grid point. If the distance value is the local maximum, this grid point can be considered the center of a hole. The performance of this volumetric approach can be improved by introducing a hierarchical version of this algorithm (see [31] for more detail), which allows us to obtain a $O(n^2 \log n)$ speed, when n is the diameter of the volumetric grid. Also, we calculate normal vectors for the latter steps using principal component analysis (PCA) with neighbor points.

When a hole Υ is detected, we can identify its boundary $\partial\Upsilon$ by first collecting all the surface points in Γ residing in the neighborhood of the boundary grid points. We can check over each point $\boldsymbol{p} \in \Gamma$, and project the neighboring points of $\boldsymbol{p}$ onto its tangent plane. If the angle of the open fan area (the region where no neighboring point resides) is larger than a certain threshold angle, we consider $\boldsymbol{p}$ as belonging to the boundary of the hole $\partial\Upsilon$. Figure 4 shows an example of the automatic hole detection for the incomplete dragon model.

5 Surface patch selection

After the holes of the original point surface have been identified, they should be filled in a way that is minimally distinguishable from their surrounding regions while maintaining certain degrees of continuity with boundaries. Because there is no information inside the hole region, it would be better to use the context information obtained from other regions that do not contain holes. If the original model has enough context information, holes can be filled automatically (both shape and appearance) by conforming with the natural properties of the model. This can be achieved by automatically translating, rotating, and possibly warping copies of points from another region that is most similar to the hole region.

Without global parameterization, surfaces lack a natural intrinsic spatial structure, which can facilitate searching and selecting similar surface regions for the holes. To tackle this underlying difficulty, we have to resort to a volumetric embedding structure (in this paper we utilize the hierarchical octree structure) to define where and how to search for and select adequate patches.

In this paper, we propose to perform local parameterization, to compare both the shape and appearance of the local surface patches discretized by the octree structure. In Sect. 5.1, we will address the local parameterization techniques. The patch similarity is based on computing curvature-driven digital signatures, which will be addressed in Sect. 5.2. In addition, there is no guarantee that this automatic approach will always give the best solution from a human perspective. Thus, it can be complemented by user intervention. We will address manual selection issues in Sect. 5.3. The surface patch selection is performed separately for geometry and texture.

5.1 Local parameterization

In order to compare both the shape and appearance of the local surface patches $\Gamma_i \subset \Gamma$, we perform local parameterization for the surface patches in each octree cell up to a certain layer λ. λ is determined from the highest layer of the existing holes, and it should also satisfy the condition that each octree cell beneath the layer contains at most one surface patch that is homeomorphic to a topological disc; otherwise we do not parameterize the "high-genus" surface patch ($genus \geq 1$) and leave it alone. If the surface patch does not contains a hole, several existing local parameterization methods can be utilized to parameterize each local patch. In this paper, we utilize the minimum distortion parameterization method in [34].

Let $X : [0, 1] \times [0, 1] \rightarrow \Gamma_i$ be the mapping from a parameter domain D_i to a surface patch Γ_i. Then the parameterization becomes the minimization problem with the following cost function:

$$C(X) = \sum_{j \in M} \{X(\boldsymbol{s}_j) - \boldsymbol{p}_j\}^2 + \epsilon \int_{D_i} \gamma(\boldsymbol{s}) d\boldsymbol{s}, \tag{1}$$

where $\gamma(\boldsymbol{s}) = \int_\theta (\frac{\partial^2}{\partial r^2} X_{\boldsymbol{s}}(\theta, r))^2 d\theta$, and $\boldsymbol{s} = (u, v) \in D_i$, $\boldsymbol{p} \in \Gamma_i$. $X_{\boldsymbol{s}}(\theta, r)$ denotes local polar reparameterization defined as $X_{\boldsymbol{s}}(\theta, r) = X(\boldsymbol{s} + r[\begin{smallmatrix}\cos(\theta)\\ \sin(\theta)\end{smallmatrix}])$. The first term in Eq. 1 represents squared error of the parameterization given by constraints and the second term estimates the distortion by integrating the squared curvature $\gamma(\boldsymbol{x})$ in the parameter domain D. Let the inverse mapping of X be $U : \Gamma_i \rightarrow [0, 1] \times [0, 1]$. To minimize Eq. 1 and to obtain U, Zwicker et al. [34] proposed a discretization method under the point cloud setting.

Since the octree cells are aligned with the world coordinate system (instead of the object-space), the surface patches inside the octree cells can be of arbitrary shape. For example, in Fig. 5 (a), the local patch inside the octree cell is "three-sided". This would make our local surface parameterization undesirable if it maps an "n-sided" surface patch onto the $[0, 1] \times [0, 1]$ parameter domain. This problem can be remedied by selecting local "four-sided" patches in an object-space fashion. We use PCA to obtain the principal directions $\boldsymbol{u}$, $\boldsymbol{v}$, $\boldsymbol{w}$ of the local patch, where the origin $\boldsymbol{o}$ of these local axes resides at its center of mass. Given a user-specified patch length L, we can check if a point $\boldsymbol{p}$ belongs to the "four-sided" local patch by testing if $\overrightarrow{\boldsymbol{op}} \cdot \boldsymbol{u} \leq L$ and $\overrightarrow{\boldsymbol{op}} \cdot \boldsymbol{v} \leq L$, where $\overrightarrow{\boldsymbol{op}} = \boldsymbol{p} -$

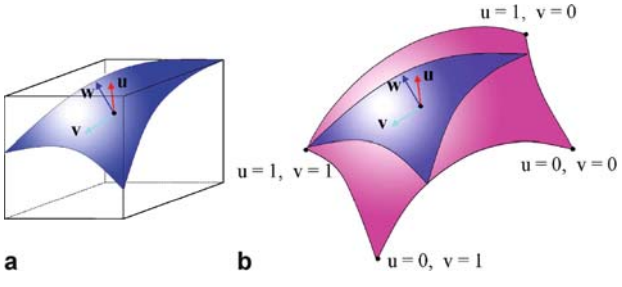

Fig. 5. a A local surface patch inside the octree cell; **b** automatically selecting local "four-sided" surface patches to facilitate local parameterization

$\boldsymbol{o}$. The checking process can start from $\boldsymbol{o}$. If it belongs to the "four-sided" local patch, we can continue to check its neighbors in a recursive way. The four farthest corner points can be set as positional constraints in the first term of Eq. 1 (see Fig. 5 (b)).

If the surface patch contains holes, the distortion part in (Eq. 1) is hard to estimate around the hole boundary $\partial\Upsilon$. We parameterize Γ_i by fixing parameter values of hole boundary using the reference plane fitted by the points in Γ_i. Let n_h be the number of holes in Γ and the k-th hole Υ_i^k be contained in the surface patch Γ_i. We use PCA to obtain the principal directions $\boldsymbol{u}$, $\boldsymbol{v}$, $\boldsymbol{w}$ of Γ_i, where the local reference plane is spanned in the $\boldsymbol{u}$ and $\boldsymbol{v}$ directions. Having the reference plane as a parameter domain D_i, we can calculate parameter values of points $\boldsymbol{p} \in \partial\Upsilon_i^k$ by direct projection (see Fig. 6). After obtaining the parameter values for the boundary points in Υ_i^k, we can set them as additional constraints (first term) in Eq. 1, and obtain the parameter values for the other points in Γ_i.

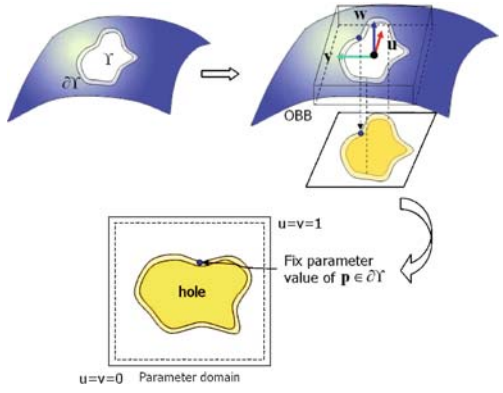

Fig. 6. Local parameterization of patches with a hole

5.2 Automatic selection

After local parameterization, we can compare the surface patches based on their curvature (Sect. 5.2.1) and color information by computing their "digital signatures" (Sect. 5.2.2). The selected best-matching patch can be warped to the hole region by solving Poisson equations (Sect. 6) directly over the surface patches.

5.2.1 Curvature estimation

We can analyze both geometry and texture similarities based on the curvature and color information of the patches discretized by the octree structure. Color information at each point for comparing appearance is typically given by the scanned data. Curvature values for comparing geometry need to be estimated at each point. For triangular meshes, there are several existing methods [22, 26] to estimate surface curvatures that are quite stable with noisy data using wide neighbor area and weights. In this paper, we take the similar *normal voting* approach of [22] to approximate the surface curvature at each point in the point cloud.

Let $\kappa_{\boldsymbol{p}}(\boldsymbol{T}_\theta)$ be the surface normal curvature at point $\boldsymbol{p}$ in the unit length tangent direction $\boldsymbol{T}_\theta$. The symmetric matrix:

$$\boldsymbol{M}_{\boldsymbol{p}} = \frac{1}{2\pi} \int_{-\pi}^{\pi} \kappa_{\boldsymbol{p}}(\boldsymbol{T}_\theta) \boldsymbol{T}_\theta \boldsymbol{T}_\theta^t d\theta \tag{2}$$

has eigenvectors that are equivalent to the principal directions $\{\boldsymbol{T}_1, \boldsymbol{T}_2\}$, and its eigenvalues $\{m_{\boldsymbol{p}}^1, m_{\boldsymbol{p}}^2\}$ are related to the principal curvatures $\{\kappa_{\boldsymbol{p}}^1, \kappa_{\boldsymbol{p}}^2\}$ as:

$$\kappa_{\boldsymbol{p}}^1 = 3m_{\boldsymbol{p}}^1 - m_{\boldsymbol{p}}^2, \quad \kappa_{\boldsymbol{p}}^2 = 3m_{\boldsymbol{p}}^2 - m_{\boldsymbol{p}}^1. \tag{3}$$

Note that the normal vector $\boldsymbol{N}_{\boldsymbol{p}}$ is another eigenvector of $\boldsymbol{M}_{\boldsymbol{p}}$ associated with the eigenvalue 0.

In our discretized point cloud setting, we consider the local neighborhood $\{\boldsymbol{q}_1, ..., \boldsymbol{q}_m\}$ around $\boldsymbol{p}$ to estimate the principal curvature of the surface at point $\boldsymbol{p}$. For each neighboring point $\boldsymbol{q}_i$, we estimate the normal curvature at point $\boldsymbol{p}$ as:

$$\kappa_i = \frac{\Delta\vartheta_i}{\Delta s}, \tag{4}$$

where ϑ_i is the turning angle, and s is the arc length (see Fig. 7). We project the normal of $\boldsymbol{q}_i$ (denoted as $\boldsymbol{N}_{\boldsymbol{q}_i}$) onto the plane Π_i that contains $\boldsymbol{N}_{\boldsymbol{p}}$, $\boldsymbol{p}$, and $\boldsymbol{q}_i$, to obtain the projected vector $\tilde{\boldsymbol{N}}_{\boldsymbol{q}_i}$. $\Delta\vartheta_i$ is the change in turning angle, and it can be computed as:

$$\cos(\Delta\vartheta_i) = \frac{\boldsymbol{N}_{\boldsymbol{p}} \cdot \tilde{\boldsymbol{N}}_{\boldsymbol{q}_i}}{||\tilde{\boldsymbol{N}}_{\boldsymbol{q}_i}||}. \tag{5}$$

The change in arc length Δs can be estimated as the geodesic distance from $\boldsymbol{q}_i$ to $\boldsymbol{p}$.

After computing the directional normal curvature from each neighboring point $\boldsymbol{q}_i$, we can approximate Eq. 2 as:

$$\tilde{\boldsymbol{M}}_{\boldsymbol{p}} = \frac{1}{2\pi} \sum \omega_i \kappa_i \boldsymbol{T}_i \boldsymbol{T}_i^t, \tag{6}$$

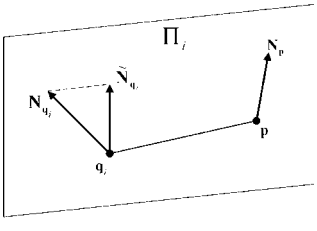

Fig. 7. Normal voting for curvature estimation

where the weight ω_i must satisfy the constraint $\sum \omega_i = 2\pi$. By analyzing the eigenvalues of $\tilde{\boldsymbol{M}}_{\boldsymbol{p}}$, we can estimate the principal curvatures from (Eq. 3).

5.2.2 Finding similar patches

After parameterization, the local patches cannot be directly compared with the hole patch, since the hole region contains no geometric/textural information. The comparison has to be performed outside the hole region based on its parametric correspondence (see Fig. 8). However, there is still an extra degree of freedom to rotate the local patches on their parametric plane before comparison. To tackle this problem, we quantize the local parametric orientation to 24 discrete angles, i.e., $i \times 15°$, $i = 0, 1, \ldots, 23$. We rotate the local patches on their parametric plane, and pre-compute 24 different rotated parametric copies of the same patch. Due to the symmetry about the uv axis, we can save memory space by storing only six rotated copies at each octree cell. Similar strategies of preprocessing textures are used in [28, 29].

To find similar patches, we use curvature information for geometry comparison and (R, G, B) color values for appearance. We select several signatures, which represent both the geometry and color information of a given patch. For geometry, we select six signatures $(f_1, \ldots, f_6)$ related to the curvatures, considering the fact that the curvatures are the intrinsic geometric properties of a surface patch. They are defined as:

$$\begin{aligned} f_1 &= \frac{\sum_j \kappa_{1j}}{n_i}, \quad f_2 = \max_j\{\kappa_{1j}\}, \quad f_3 = \min_j\{\kappa_{1j}\} \\ f_4 &= \frac{\sum_j \kappa_{2j}}{n_i}, \quad f_5 = \max_j\{\kappa_{2j}\}, \quad f_6 = \min_j\{\kappa_{2j}\}\ , \end{aligned} \tag{7}$$

where κ_{1j} is the maximum curvature of $\boldsymbol{p}_j \in \Gamma_i$, κ_{2j} is the minimum curvature of $\boldsymbol{p}_j$, and n_i is the number of points in Γ_i. For texture comparison, we use nine signature values, $g_1, \ldots, g_9$, which represent the average, the maximum, and the minimum color value of R, G, B, respectively.

Given a hole patch Υ_k, $(k \in \{0, \ldots, n_h\})$, we need to compare it with other surface patches Γ_i, $(i = 0, \ldots, n_\lambda)$ using signatures. The similarity function $\mathrm{S}(f_j^{\Upsilon_k}, f_j^{\Gamma_i})$ (S_j for short) can be defined as:

$$\mathrm{S}_j = \left(1 - \frac{d_j}{d_{\max,j}}\right)^r, \tag{8}$$

where $f_j^{\Upsilon_k}$ is the j-th signature value of the patch Υ_k, $d_j = |f_j^{\Upsilon_k} - f_j^{\Gamma_i}|$, $d_{\max,j} = \max_j\{d_j\}$, $j = 0, \ldots, n_i$. r represents the sensitivity of S along d_j, and we simply set $r = 2$.

Then, the similarity between two surface patches is defined as a normalized value of the weighted sum, which is obtained by comparing each signature:

$$\text{similarity}(\Upsilon_k, \Gamma_i) = \frac{\sum_j w_j \mathrm{S}_j}{\sum_j w_j}, \tag{9}$$

After calculating the similarity value of each surface patch, we can find the most similar patch Γ_{i*}, which has the maximum similarity value as a candidate to fill Υ. However, the automatic approach may not always guarantee the best results from a human perspective. So, our algorithm can provide several candidates that have the maximum similarity values, and the users can select one of them as Γ_{i*}. Also, the most similar patches are not necessarily the same for geometry and texture filling, so we can find different Γ_{i*} for geometry and texture separately.

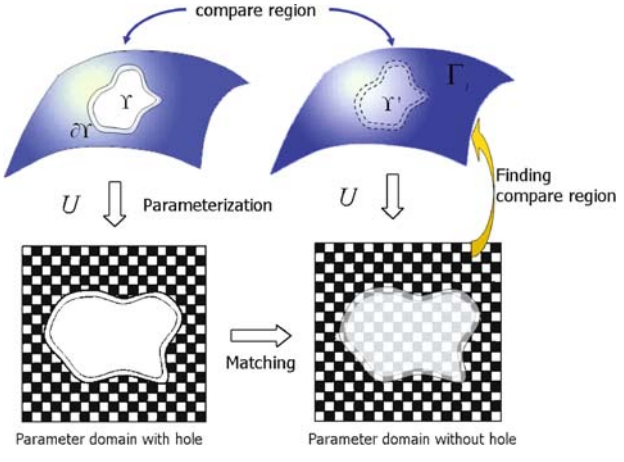

Fig. 8. Surface patch comparison outside the hole region

5.3 Manual selection

The automatic selection approach based on local parameterization works very well for local surface patches that are relatively flat and simple. For complicated local geometry such as the claw of the dragon model in Fig. 9, however, local parameterization would face severe stretch and distortion, which make it inappropriate for subsequent processing. In some surface completion tasks, the scanned model doesn't have enough context information, or the automatic selection in Sect. 5.2 may not give a global solution because of the octree structure. For example, in Fig. 1, the octree discretization cannot give any plausible solution for the missing hand of the Santa Claus model without user intervention. In our system, we provide efficient and accurate user interaction techniques to handle incomplete point sets.

Let us assume that a 3D closed curve $\mathcal{C}$ located on the surface patch is given by a user, and this curve separates the input point set into two parts. We want to find these two parts divided by $\mathcal{C}$. Because there is no connectivity

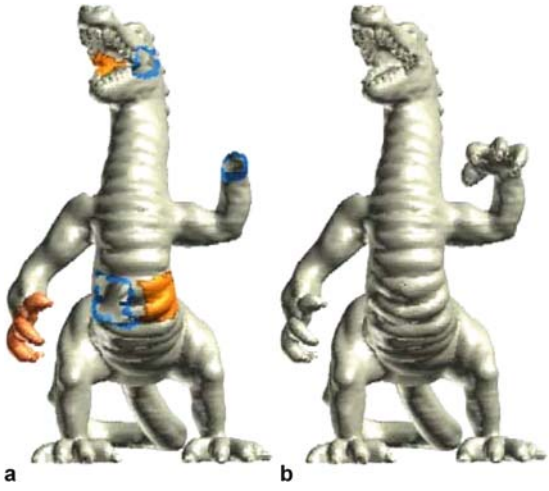

Fig. 9a,b. Repairing the dragon model by user manipulation; **a** the user selects the most similar regions (orange region) from the model's context; **b** the final repaired model

between pure points, we embed Γ and $\mathcal{C}$ into the volumetric grids used in Sect. 4 for efficiency. We visit the grid points in two opposite directions starting from the cells intersecting $\mathcal{C}$. The grid points that we visit are only limited to those that sandwich the point surface. The front propagation on the narrow band grids is similar to the tangential flow approach, and [17] also suggest a similar technique in 2D. Based on the assumption that the input point set is obtained from a two-manifold surface model, this front propagation works very well in all of our experiments. Figure 9 shows an example of surface completion based on user manipulation.

5.4 Boundary detection

The hole filling process introduced in Sect. 6 is basically achieved by translating, rotating, and possibly warping copies of the points from the selected region to the target hole region based on their boundary information. At this stage, the boundary of the selected patch $\partial\Gamma_{i^*}$ should be detected. The same technique used in Sect. 4 can be applied to decide whether a point in Γ_{i^*} belongs to $\partial\Gamma_{i^*}$ or not. Because there may be several closed boundary curves in one patch, we need to check and separate them. Let us assume that if $\boldsymbol{p_1}$ and $\boldsymbol{p_2}$ belong to different boundary curves then $\text{dist}(\boldsymbol{p_1}, \boldsymbol{p_2}) > r$, where $\text{dist}(\boldsymbol{p_1}, \boldsymbol{p_2})$ is the Euclidean distance between $\boldsymbol{p_1}$ and $\boldsymbol{p_2}$, and r is the tolerance value. We construct a Euclidean minimum spanning tree (EMST) of boundary points. Then we examine each edge of the EMST, and cut those edges longer than r (see Fig. 10). The remaining connected points compose the closed boundary curves. Actually, the points in the remaining EMST are not connected to form closed curves. However, we can simply collect all the separated boundary points, since their connectivity information is not needed for the patch alignment process later (Sect. 6.1).

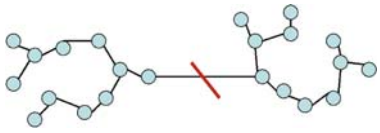

Fig. 10. The Euclidean minimum spanning tree (EMST) and boundary curve separation

6 Filling holes

Before filling a hole with the selected copy region, the initial alignment of these two regions should be performed with respect to a rigid body transformation (Sect. 6.1). After initial alignment between the selected context patch and the original hole-region, there may be still some gaps that cannot be further reduced by rigid body transformation. At this moment, the Poisson equation is used to solve for the remaining displacement field (Sects. 6.2 and 6.3). The displacement field is generally smooth based on the Poisson equation. However, their geometric and textural detail/sharp features are maintained as much as possible, since they are input as the guidance field in the Poisson equation, and can be preserved in a least squared sense. A hole filling step is also performed twice for geometry and texture, respectively.

6.1 Initial alignment

Because we have no point in the hole, the transformation from $\partial\Gamma_{i^*}$ to $\partial\Upsilon$ should be performed to align the copy region with the hole. Here, $\partial\Upsilon$ is the banded point surface region around the hole, and $\partial\Gamma_{i^*}$ is the set of points on Γ_{i^*} corresponding to $\partial\Upsilon$. In our current work, we use the ICP method, which gives a quite reasonable result for the registration between two banded regions. Since ICP does not work when the two point sets are not close, we align the centroid of two point sets and coordinates of two OBBs first. Also, normal information is used to check whether the copy patch is turned over. Let $\boldsymbol{T}$ be the (4×4) matrix that represents rigid body transformation. Let $\boldsymbol{q}_j \in \partial\Gamma_{i^*}$ be the nearest point of $\boldsymbol{p}_j \in \partial\Upsilon$, and the distance between two point sets be defined as:

$$d(\partial\Upsilon, \partial\Gamma_{i^*}) = \sum_j ||\boldsymbol{p}_j - \boldsymbol{q}_j||^2. \tag{10}$$

The traditional ICP method finds the matrix $\boldsymbol{T}$ that satisfies:

$$\min_{\boldsymbol{T}} d(\partial\Upsilon, \boldsymbol{T}\partial\Gamma_{i^*}). \tag{11}$$

A similar procedure can be applied to the texture alignment by modifying the distance between two points $\boldsymbol{p}_1 = (x_1, y_1, z_1, r_1, g_1, b_1)$ and $\boldsymbol{p}_2 = (x_2, y_2, z_2, r_2, g_2, b_2)$ to include additional color information. We use the following distance measure, similar to the *colored ICP* method [12]:

$$d_6(\boldsymbol{p}_1, \boldsymbol{p}_2) = \left[\begin{array}{c} (x_1 - x_2)^2 + w_1(r_1 - r_2)^2 + \\ (y_1 - y_2)^2 + w_2(g_1 - g_2)^2 + \\ (z_1 - z_2)^2 + w_3(b_1 - b_2)^2 \end{array} \right]^{\frac{1}{2}}. \quad (12)$$

In our experiments we choose $w_1 = w_2 = w_3 = 1/3$.

Because this initial alignment uses only boundary regions, it may not give the most appropriate solution conforming with the model's context. (E.g., in Fig. 1, the copied hand of the Santa Claus model would need a mirror reflection before the final warping.) If the alignment is not appropriate according to the overall context, users can rotate, translate, or symmetrically reflect the patch after the automatic alignment process.

6.2 Poisson model

Partial differential equation (PDE) techniques have been widely used in many visual computing applications. The PDE methods model graphical objects as solutions of certain elliptic PDEs with boundary constraints. The PDE model uses only boundary conditions to recover all of the hole interior information and offers high-order continuity, as well as energy minimization properties. Both the geometry and color information of the surface patch that has highest similarity can be blended with the hole patch by solving PDEs.

In our implementation, to merge two point sets smoothly with boundary conditions, we choose the Poisson equation with Dirichlet boundary conditions [23, 32], which is a second-order PDE and can be solved more efficiently:

$$\nabla^2 f = \operatorname{div} \boldsymbol{h} \quad \text{over } \Omega, \quad \text{with } f|_{\partial\Omega} = f^*|_{\partial\Omega}, \quad (13)$$

where $\boldsymbol{h}$ is the guidance vector field, $\nabla^2 = (\frac{\partial^2}{\partial x^2} + \frac{\partial^2}{\partial y^2} + \frac{\partial^2}{\partial z^2})$, and div $\boldsymbol{h}$ is the divergence of $\boldsymbol{h}$. It can be verified that the Poisson equation (Eq. 13) is the solution of the minimization problem:

$$\min_f \int_{\Omega} |\nabla f - \boldsymbol{h}|^2, \quad \text{with } f|_{\partial\Omega} = f^*|_{\partial\Omega}, \quad (14)$$

If the guidance field $\boldsymbol{h}$ is the gradient of some guidance function g (i.e., $\boldsymbol{h} = \nabla g$), we can define a correction function $\tilde{f} = f - g$. In this way, the Poisson equation becomes the Laplacian equation with boundary conditions:

$$\nabla^2 \tilde{f} = 0 \text{ over } \Omega, \quad \text{with } \tilde{f}|_{\partial\Omega} = (f^* - g)|_{\partial\Omega}, \quad (15)$$

In our hole filling problem, the guidance function g can be simply the geometry and color information of the selected surface patch (see Fig. 11).

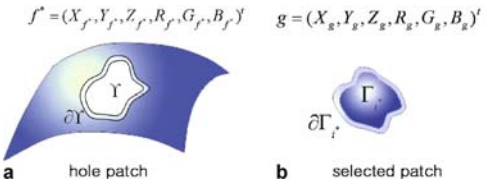

Fig. 11. The selected "most" similar patch is warped to the hole patch by solving a Poisson equation, where the guidance function g is simply the geometry and texture information of the selected patch

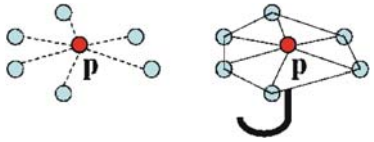

Fig. 12. Local umbrella operator for computing the Laplacian at each point

6.3 Poisson solver

The Laplacian equation (Eq. 15) can be solved directly on the point sampled surfaces. It can be discretized over both the selected surface patch and the hole patch on each point sample. To specify the boundary condition, each point $\boldsymbol{p} \in \partial\Gamma_{i^*}$ is mapped to the nearest point in $\partial\Upsilon$ before evaluation.

The discretization of Eq. 15 simply means the discretization of the Laplacian operator $\nabla^2 \bar{f}$, which can be approximated using the *scale-dependent umbrella operator* [8]:

$$\mathrm{U}(\boldsymbol{p}_i) = \sum_{j=0}^{n} \varepsilon_{\boldsymbol{p}_i, \boldsymbol{q}_j} (\bar{f}(\boldsymbol{q}_j) - \bar{f}(\boldsymbol{p}_i)), \quad (16)$$

where n is the number of neighboring points of $\boldsymbol{p}_i$, $\boldsymbol{q}_j \in N(\boldsymbol{p}_i)$, and the weights $\varepsilon_{\boldsymbol{p}_i, \boldsymbol{q}_j} = \frac{1}{\|\boldsymbol{q}_j - \boldsymbol{p}_i\|} / \sum_{k=0}^{n} \frac{1}{\|\boldsymbol{q}_k - \boldsymbol{p}_i\|}$ are in inverse proportion to their distances. Figure 12 describes the meaning of this operator.

Now, by applying Eq. 16 with position and color values to all the points $\boldsymbol{p} \in \Gamma_{i^*}$, we can formulate a sparse linear system $\boldsymbol{A}\bar{\boldsymbol{f}} = \boldsymbol{b}$, which can be solved numerically using a conjugate gradient method.

7 Experimental results

Our system is implemented on a Microsoft Windows XP PC with Xeon 2.80 GHz CPU, 2.00 GB of RAM. We tested our system on several incomplete point surfaces, and recorded the statistics of our system's performance in Table 1. Figure 13 shows an example of a scanned Buddha model with one large hole, which is detected and filled

Table 1. The time performance of our shape and appearance completion algorithm (in seconds)

Model	Points	Holes	Param.	Finding	Filling
Buddha	13,942	1	No	131 s	16 s
Chameleon	99,835	1	Yes	193 s	108 s
Dragon	52,982	3	No	53 s	32 s
Female	123,369	3	No	62 s	89 s
Iphigenie	144,622	3	Yes	104 s	361 s
Male	145,177	6	Yes	153 s	653 s
Rocker arm	39,501	1	Yes	103 s	87 s
Santa	71,438	3	No	67 s	78 s

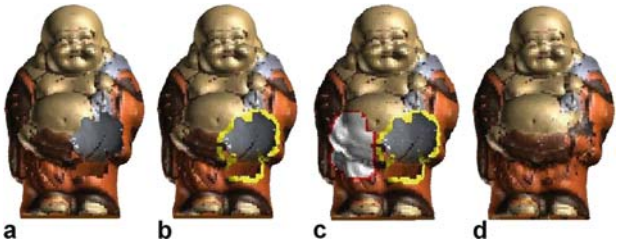

Fig. 13a–d. Repairing the Buddha model under user manipulation; **a** the original point set with one large hole; **b** automatic hole detection (with hole boundary in yellow); **c** the selected surface patch (only for shape) is rendered in white and the corresponding boundary in red; **d** final result

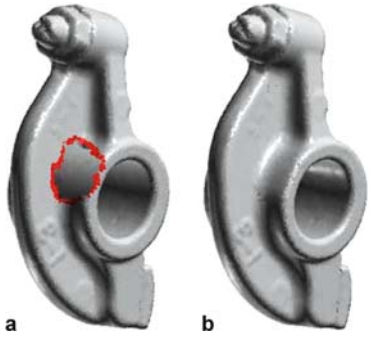

Fig. 14a,b. Repairing the sharp feature of the rocker arm model automatically: **a** the original point set with one hole; **b** the sharp feature is repaired from the model's context

Fig. 15a–c. Automatically repairing the Iphigenie model based on local parameterization: **a** the original point surface with several missing parts; **b** the holes are detected automatically (with hole boundary points drawn in red); **c** the model is repaired conforming to the context information based on local parameterization

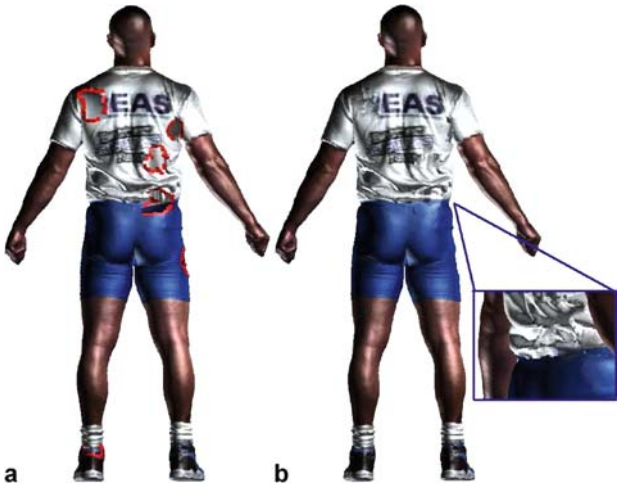

Fig. 16a,b. Automatic shape and appearance repair based on local parameterization: **a** the holes in the original point set can be detected automatically; the points near the hole boundaries are rendered in red. **b** both the shape and appearance of the hole regions can be repaired based on local parameterization

automatically. The shape and appearance of the missing part is completed with details taken from the existing parts of the body. The sharp feature of rocker arm model in Fig. 14 can be also repaired gracefully and automatically based on its context information. The Iphigenie model (Fig. 15) and the male model (Fig. 16) are both non-trivial examples with several complicated missing parts, which are all detected and repaired automatically conforming to their context information based on the local parameterization. Also, very complicated texture of the chameleon model (Fig. 17) is repaired from model's context using same method. The local parameterization approach can only handle relatively simple holes and is not suitable for repairing the hand of the Santa Claus model (Fig. 1), the claw of the dragon model (Fig. 9), or the foot of the female model (Fig. 18). The missing parts in all these models are very complicated, making them inappropriate for local parameterization. The holes of these models are all detected automatically using our active contour approach. The hand of Santa Claus, the claw of the dragon, and the foot of the female model are repaired interactively by user manipu-

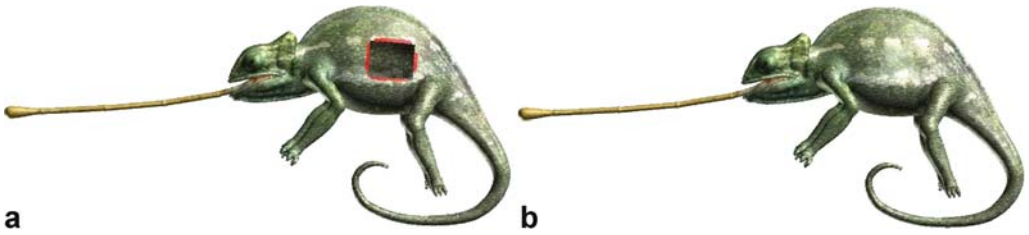

Fig. 17a,b. Automatically repairing shape and appearance of the chameleon model; **a** the red points are near the boundary of the hole; **b** complicated texture information can be repaired from the model's context based on local parameterization

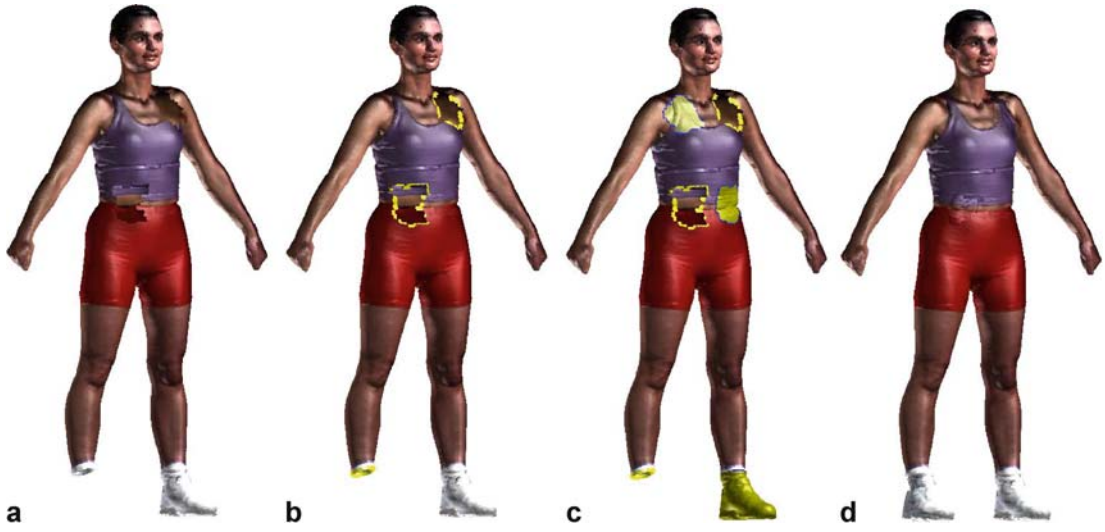

Fig. 18a–d. Repairing the female model under user manipulation: **a** the original point set with three complicated holes; **b** automatic hole detection (with hole boundary in yellow); **c** the selected surface patch (only for shape) is rendered in yellow and the corresponding boundary in blue; **d** final result

lation, while other relatively "flat" holes are repaired automatically conforming to their context information based on local parameterization.

The number of surface patches used in Sect. 5.2.2 is dependent on the level of the octree. In the case of the Iphigenie model, an average of 282 patches are used for comparison. These results may be affected by some decision parameters that are introduced at several stages. In the hole detection stage, user-defined threshold is needed to detect boundaries. When estimating curvature, the neighborhood size is an important parameter, which we set as nine for our experiments. Also, the weights to calculate similarity in (Eq. 9) and color ICP in (Eq. 12) are simply set as equal average values.

8 Conclusion

In this paper we have developed a novel surface content completion system that can repair both shape and appearance of incomplete point set inputs. The entire model repair pipeline consists of hole detection, automatic/interactive patch selection, and hole filling via the cut-and-paste operation. We utilize the active contour method to facilitate robust hole detection from the noisy and defective data sets. Our surface content completion enables automatic context-based geometry and texture filling simultaneously. We use boundaries to align the local patches and to solve Poisson equations for warping the patch to cover the hole region and to achieve a smooth transition across the hole boundary. Local parameterization is utilized to facilitate automatic patch selection, alignment, and the warping process. Compared with other existing work on surface completion, our method can automatically repair color information in addition to just geometry, and can achieve better efficiency, since all these operations are performed on a local 2D parameterization domain rather than on the volume through the volumetric-embedding mechanism. Our surface completion framework and its constituents are of particular value to computer graphics applications such as model reconstruction and shape modeling.

Acknowledgement This research was conducted at Stony Brook University when Ms. Seyoun Park was an exchange Ph.D candidate in the Computer Science Department and Center for Visual Computing. This work was partially supported by KRF grant M07-2003-000-20301-0 to S. Park, and by NSF grant ACI-0328930, ITR grant IIS-0326388, and the Alfred P. Sloan Fellowship to H. Qin. The Santa, rocker arm, male, and female models are courtesy of Cyberware, Inc.

References

1. Amenta, N., Bern, M., Kamvysselis, M.: A new Voronoi-based surface reconstruction algorithm. Proc. SIGGRAPH, pp. 415–421 (1998)
2. Bajaj, C.L., Bernardini, F., Xu, G.: Automatic reconstruction of surfaces and scalar fields from 3D scans. Proc. SIGGRAPH, pp. 109–118 (1995)
3. Bernardini, F., Mittleman, J., Rushmeier, H., Silva, C., Taubin, G.: The ball-pivoting algorithm for surface reconstruction. IEEE Trans. Vis. Comput. Graph. **4**, 349–359 (1999)
4. Biermann, H., martin, I., Bernardini, F., Zorin, D.: Cut-and-paste editing of multiresolution surfaces. ACM Trans. Graph. **21**(3), 312–321 (2002)
5. Carr, J.C., Beatson, R.K., Cherrie, J.B., Mitchell, T.J., Fright, W.R., McCallum, B.C., Evans, T.R.: Reconstruction and representation of 3D objects with radial basis functions. Proc. SIGGRAPH, pp. 67–76 (2001)
6. Clarenz, U., Diewald, U., Dziuk, G., Rumpf, M., Rusu, R.: A finite element method for surface restoration with smooth boundary conditions. Comput. Aided Geom. Des. **5**, 427–445 (2004)
7. Davis, J., Marschner, S.R., Garr, M., Levoy, M.: Filling holes in complex surfaces using volumetric diffusion. Proceedings International Symposium on 3D Data Processing, Visualization, and Transmission, Padova, Italy, June 19–21, 2002, pp. 428–438. IEEE Computer Society (2002)
8. Desbrun, M., Meyer, M., Schröder, P., Barr, A.H.: Implicit fairing of irregular meshes using diffusion and curvature flow. Proc. SIGGRAPH, pp. 317–324 (1999)
9. Edelsbrunner, H., Mucke, E.P.: Three-dimensional alpha shapes. ACM Trans. Graph. **13**(1), 43–72 (1994)
10. Fu, H., Tai, C., Zhang, H.: Topology-free cut-and-paste editing over meshes. Proceedings: Geometric Modeling and Processing, Beijing, China, April 13–15, 2004, pp. 173–182. IEEE Computer Society (2004)
11. Hoppe, H., DeRose, T., Duchamp, T., McDonald, J., Stuetzle, W.: Surface reconstruction from unorganized points. Proc. SIGGRAPH, pp. 71–78 (1992)
12. Johnson, A.E., Kang, S.B.: Registration and integration of textured 3D data. Proceedings International Conference on Recent Advances in 3D Digital Imaging and Modeling, Ottawa, Canada, May 12–15, 1997, pp. 234–241. IEEE Computer Society (1997)
13. Ju, T.: Robust repair of polygonal models. ACM Trans. Graph. **23**(3), 888–895 (2004)
14. Kass, M.A.W., Terzopoulos, D.: Snakes: active contour models. Int. J. Comput. Vis. **1**(4), 321–331 (1987)
15. Kraevoy, V., Sheffer, A.: Template-based mesh completion. Proceedings of Eurographics Symposium on Geometry Processing, pp. 13–22 (2005)
16. Masuda, T.: Filling the signed distance field by fitting local quadrics. Proceedings International Symposium on 3D Data Processing, Visualization, and Transmission, pp. 1003–1010 (2004)
17. Mortensen, E.N., Barrett, W.A.: Interactive segmentation with intelligent scissors. Graph. Models **60**(5), 349–384 (1998)
18. Levin, D.: The Approximation power of moving least-squares. Mathematics of Computation, **67**(224), 1517–1531 (1998)
19. Pauly, M., Mitra, N.J., Giesen, J., Gross, M., Guibas, L.J.: Example-based 3D scan completion. Proceedings Eurographics Symposium on Geometry Processing, pp. 23–32 (2005)
20. Podolak, J., Rusinkiewicz, S.: Atomic volumes for mesh completion. Proceedings Eurographics Symposium on Geometry Processing, pp. 33–41 (2005)
21. Ohtake, Y., Belyaev, A., Alexa, M., Turk, G., Seidel, H.P.: Multi-level partition of unity implicits. ACM Trans. Graph. **22**(3), 463–470 (2003)
22. Page, D.L., Koschan, A., Sun, Y., Pail, J., Abidi, M.A.: Robust crease detection and curvature estimation of piecewise smooth surfaces from triangle mesh approximations using normal voting. Proceedings International Conference on Computer Vision and Pattern Recognition, Vol. 1, pp. 162–167 (2001)
23. Perez, P., Gangnet, M., Blake, A.: Poisson image editing. ACM Trans. Graph. **22**(3), 313–318 (2003)
24. Sharf, A., Alexa, M., Cohen-Or, D.: Context-based surface completion. ACM Trans. Graph. **23**, 878–887 (2004)
25. Savchenko, V., Kojekine, N.: An approach to blend surfaces. Proc. Computer Graphics International, Bradford, UK, July 1–5 (2002)
26. Taubin, G.: Estimating the tensor of curvature of a surface from a polyhedral approximation. Proceedings International Conference on Computer Vision, pp. 902–907 (1995)
27. Verdera, J., Caselles, V., Bertalmío, M., Sapiro, G.: Inpainting surface holes. Proceedings International Conference on Image Processing, Barcelona, Spain, Sept. 14–17, 2003. IEEE Computer Society (2003)
28. Wang, B., Wang, W., Yang, H., Sun, J.: Efficient example-based painting and synthesis of 2d directional texture. IEEE Trans. Vis. Comput. Graph. **10**(3), 266–277 (2004)
29. Soler, C., Cani, M.P., Angelidis, A.: Hierarchical pattern mapping. ACM Trans. Graph. **21**(3), 673–680 (2002)
30. Whitaker, R.T.: A level-set approach to 3d reconstruction from range data. Int. J. Comput. Vis. **3**, 203–231 (1998)
31. Xie, H., McDonnell, K., Qin, H.: Surface reconstruction of noisy and defective data sets. Proc. Visualization, Austin, TX, Oct. 10–15, 2004, pp. 259–266. IEEE Computer Society (2004)
32. Yu, Y., Zhou, K., Xu, D., Shi, X., Bao, H., Guo, B., Shum, H.Y.: Mesh editing with poisson-based gradient field manipulation. ACM Trans. Graph. **23**, 644–651 (2004)
33. Zhao, H.K., Osher, S., Fedkiw, R.: Fast surface reconstruction using the level set method. Proceedings IEEE Workshop on Variational and Level Set Methods, p. 194 (2001)
34. Zwicker, M., Pauly, M., Knoll, O., Gross, M.: Pointshop3d: an interactive system for point-based surface editing. Proceedings SIGGRAPH, pp. 322–329 (2002)




SEYOUN PARK is a Ph.D. Candidate in the Department of Industrial Engineering at KAIST(Korea Advanced Institute of Science and Technology). She received a B.S. degree (2002) and M.S. degree(2004) all in Industrial Engineering from KAIST. Her main research interests are in general geometric modeling, point-based graphics, computational geometry, and Bio-CAD. For more information, please send a mail to parksy@vmslab.kaist.ac.kr.

XIAOHU GUO is a Ph.D. candidate in the Department of Computer Science at State University of New York at Stony Brook. He has a B.S. degree (2001) in Computer Science from the University of Science and Technology of China. He received his M.S. degree (2004) in Computer Science from the State University of New York at Stony Brook. His research interests include computer graphics, geometric and physics-based modeling, computer animation and simulation, scientific visualization, human-computer interaction, virtual reality, and computer vision. For more information, please visit http://www.cs.sunysb.edu/∼xguo.

HAYONG SHIN is an associate professor in the Department of Industrial Engineering at KAIST(Korea Advanced Institute of Science and Technology). Before joining KAIST, He worked for DaimlerChrysler Corp., CubicTek Co. and LG Electronics, developing commercial and in-house CAD/CAM software. He received a BS from Seoul National University in 1985, an MS and a PhD from KAIST in 1987 and 1991, all in industrial engineering. His main research interests are in the area of geometric modeling, tool path generation, process planning, and computational geometry. He can be reached at hyshin@kaist.ac.kr.

HONG QIN is Associate Professor of Computer Science at State University of New York at Stony Brook. In 1997, Professor Qin was awarded NSF CAREER Award from the National Science Foundation (NSF). In December, 2000, Professor Qin received Honda Initiation Grant Award. In April, 2001, Professor Qin was selected as an Alfred P. Sloan Research Fellow by the Sloan Foundation. His areas of expertise include geometric modeling, graphics, physics-based simulation, computer aided geometric design, and human-computer interaction. At present, he is an associate editor of IEEE Transactions on Visualization and Computer Graphics (TVCG) and he is also on the editorial board of The Visual Computer (International Journal of Computer Graphics). In 2005, he co-chaired the 23rd Computer Graphics International Conference (CGI 2005). For further information, please visit http://www.cs.sunysb.edu/∼qin.