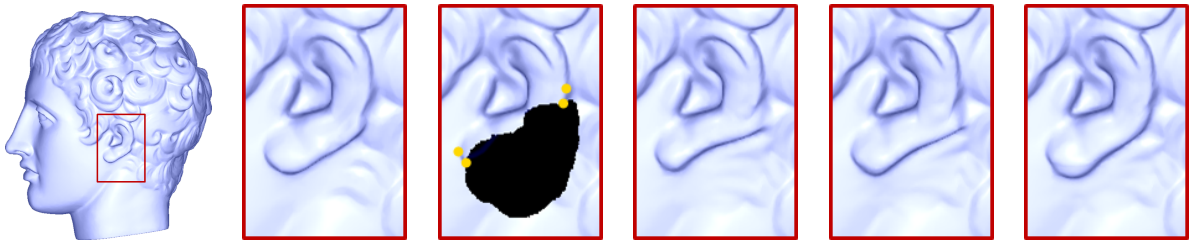
Eurographics Symposium on Geometry Processing 2014
Thomas Funkhouser and Shi-Min Hu
(Guest Editors)

Volume 33 (2014), Number 5

Feature-Preserving Surface Completion Using Four Points

G. Harary[1], A. Tal[1], and E. Grinspun[2]

[1]Technion - Israel Institute of Technology
[2]Columbia University



(a) original object (b) zoom-in of original (c) hole & four points (d) context-based without assistance (e) context-based with assistance (f) our completion

Figure 1: Completing a broken artefact using four points of user input. The hole spans part of the ear and parts of the curls. As the curls differ from each other, the hole cannot be completed by copying another part of the head. Our completion (f) uses the user's four points (c), to complete the original object (a,b). For problems such as this, involving large holes whose reconstruction requires consideration of semantics and context, our user-guided process compares favorably to (d) recent, automated context-based approaches [HTG14], (e) even if they are extended in a straightforward manner to respect user constraints.

Abstract

We present a user-guided, semi-automatic approach to completing large holes in a mesh. The reconstruction of the missing features in such holes is usually ambiguous. Thus, unsupervised methods may produce unsatisfactory results. To overcome this problem, we let the user indicate constraints by providing merely four points per important feature curve on the mesh. Our algorithm regards this input as an indication of an important broken feature curve. Our completion is formulated as a global energy minimization problem, with user-defined spatial-coherence constraints, allows for completion that adheres to the existing features. We demonstrate the method on example problems that are not handled satisfactorily by fully automatic methods.

Categories and Subject Descriptors (according to ACM CCS): I.3.5 [Computer Graphics]: Computational Geometry and Object Modeling—Curve, surface, solid, and object representations

1. Introduction

Archaeological artefact illustrators use pen-and-ink as well as 2d illustration software, to record the appearance of found objects. Tina Ross, a contemporary illustrator, writes

> Drawing of archaeological material... helps to bring to life aspects of the objects now lost through time and destruction. Decoration, shape, manufacture details, and reconstruction possibilities can be highlighted and allow further study by specialists. [Ros]

An important task of archaeological illustrators is *interpretation and reconstruction illustration* [Bar77], a process that seeks to depict archaeological finds in a manner that establishes context and completes a broken visual aesthetic (see Fig. 2); this helps specialists to focus on cultural understanding, and helps the broader public to decypher and appreciate ancient artefacts.

While present day illustrators have been mainly trained in 2d drawing, what if we could provide

effective tools to aid such illustrators in reconstructing the 3d geometry of a broken artefact? Given geometry with large, missing pieces, we set out to develop a tool that aids artists in developing a meaningful completion of the shape.

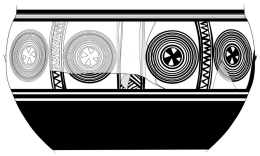


Figure 2: ***Broken vessel, interpreted, reconstructed.*** *Illustrator: Tina Ross.*

While our specific problem is less studied, the broader question of hole filling has been more comprehensively explored by the geometry processing community. When 3d models are acquired by scanning tangible objects, the resulting geometry often contains many small holes, due to occlusion, noise, and other factors. Early work filled these holes with a *smooth* completion [DMGL02, Lie03]. Later approaches employed *context* to reconstruct missing fine details and texture, by reusing pieces from other parts of the same [SACO04, HTG14] or similar objects [KS05, PMG*05].

Many of these pioneering approaches have the benefit of being autonomous. In the case of archeological reconstruction, we take the view that expert user guidance is beneficial. Archaeological illustrators imbue their works with context, history, and semantics extending beyond the reach of current computational approaches. In Fig. 1, our process completes the ear and its surroundings in a natural manner that is challenging to achieve with fully automated methods; e.g., context based completion attaches the ear to the hair (d-e).

Contributions: Our approach seeks to minimize, streamline, and make intuitive the interaction and input requested of the user. To achieve this goal we propose the following:

Simple, efficient user input: Our algorithm follows the user's guidelines and generates aesthetic meshes. We present a high level modeling tool that lets the user influence the outcome by simply indicating the prominent missing fine feature curves. To streamline user input, we ask the user to provide only four points per feature curve that crosses the hole, as seen in Fig. 1(c). These points are then used to construct an Euler spiral, which was shown to suit human perception of missing curves [GXH01, HT12].

Context-based completion respecting user constraints: We introduce a completion algorithm that respects the existing feature curves. The completion is formulated as a global energy minimization problem that takes into account the user's constraints (§3). This is done by building triangle strips (a sub-mesh) along each spiral, which avoid twists and connect smoothly to the hole's surroundings (§4). Each such sub-mesh divides the hole into smaller sub-holes. These are completed smoothly and then the entire completed region (i.e., both the triangle strips and the completed sub-holes) is modified so as to minimize the energy, while taking special care of the spirals' surroundings (§5).

We demonstrate our process on several challenging cases, evaluating the approach both qualitatively and quantitatively.

2. Related work

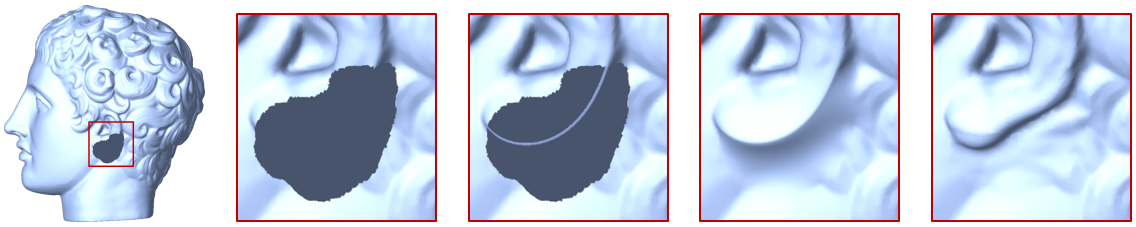
Smooth surface completion: The earlier work on surface completion may be classified as volumetric methods, which convert the mesh into a signed distance function over a grid [DMGL02, VCBS03, Ju04, BPK05, GLWZ06], or methods that operate directly on the given mesh [Lie03, CDD*04, PMV06, ZGL07]. Next, post-processing can be used to recover sharp features [CC08]. For a comprehensive survey, see [Ju09]. These methods typically suffice for small holes, which do not contain fine geometric features.

Context-based surface completion: The context-based approach manages to restore the fine geometric features to the extent that the exemplars serve as priors. These priors are either patches from the input mesh [SACO04, BSK05, HTG14], patches from other meshes [GSH*07], or an entire region from other meshes [KS05, PMG*05]. These methods are intended for objects with sufficient exemplars as priors and for not-too-large holes.

User-guided surface completion: When the above requirements do not hold, user input can be beneficial. Kraevoy and Sheffer [KS05] let the user mark several points on the model and on a template, in order to assist the alignment. Takayama et al. [TSS*11] present GeoBrush, which allows the user to copy-and-paste 3D surface geometry. Both methods rely on the presence of another region on the template that fits the missing region, an assumption we do not make. Bendels et al. [BGK06] let the user model the missing geometry on top of an initial smooth completion. This region is then refined by copying patches from other parts of the object. Like other modeling methods, it might require expertise and user effort. Wang et al. [WLL*12] introduce a method for CAD models, which first completes feature curves that cross the hole and then completes smoothly the remaining holes.

Our method is inspired by that of Sun et al. [SYJS05] for *image* completion. There, the user marks the important missing structure information by sketching lines through the hole. Then, the image along the curves is reconstructed, followed by the completion of the remaining unknown regions.

We also wish the user to indicate the meaningful feature lines through the hole. However, since our curves are three dimensional, the user cannot draw them in practice. Therefore, we let the user mark only the end-points of the curves and our algorithm constructs suitable curves, before completing the hole.



(a) input (b) feature curve & surroundings comp. (c) smooth comp. (d) final result

Figure 3: ***Algorithm outline.*** *Given an object with a large hole (a), the user marks four points on a missing ridge and the curve and its surroundings are constructed (b). Then, the remaining holes are smoothly completed (c). This smooth completion is modified to get the final result (d).*

3. Algorithm outline

Our goal is to complete a surface while being respectful of the features marked by the user. Algorithm 1 outlines our feature-aware completion method (Fig. 3). Given four points marked by the user, which lie on a broken feature curve (a ridge or a valley) that crosses the hole, the algorithm first completes the curve that fits these points and then constructs two triangle strips along it (Fig. 3(b)) (§4). As a result, the hole is split into several smaller holes. Next, these holes are smoothly completed (Fig. 3(c)). This can be done using a variety of methods for smooth completions [Lie03, CDD*04, PMV06]. We use the advancing-front method of Zhao et al. [ZGL07]. Then, each patch on the smooth completion or on the triangle strips, termed *target patch*, is replaced by a more suitable source patch, by minimizing the global error (1) in an iterative manner (Fig. 3(d)) (§5). The iterative refinement is accelerated by a coarse-to-fine approach, where larger neighborhoods are first used to reduce the low frequency error, and smaller neighborhoods then reduce the high frequency error.

Algorithm 1 Our user-guided surface completion algorithm

1: For each set of the user's four points:
   Construct the feature curve and the triangle strips along it (§4)
2: Smoothly complete the rest of the hole
3: For different object resolutions (from coarse to fine):
4: Iteratively, for each target patch
   Replace it by a source patch, minimizing the global error (1) (§5)
5: Project the target of the fine object to the coarse object

We formulate our problem as a global energy minimization problem, as follows. Let S denote the *source*, the input surface, and T denote the *target*, the unknown surface that completes the hole. Let T_i and S_j be patches in T and S, respectively. We can distinguish between patches that lie along the feature curves (T_F) and other patches on T. It is vital that along the curve the completion would use spatially coherent source patches, which will maintain the continuity of the curve. Therefore, we define our minimization problem using two distinct dissimilarity metrics between patches, one for T_F ($\mathcal{D}_2$) and the other for the rest of the target ($\mathcal{D}_1$):

$$\varepsilon(T,S) = \frac{1}{N_T - N_{T_F}} \sum_{T_i \in T \setminus T_F} \min_{S_j \in S} \mathcal{D}_1(T_i,S_j) + \frac{1}{N_{T_F}} \sum_{T_i \in T_F} \min_{S_j \in S} \mathcal{D}_2(T_i,S_j), \quad (1)$$

where N_T is the number of patches in T and N_{T_F} is the number of patches in T_F. We elaborate on the definitions of $\mathcal{D}_1$ and $\mathcal{D}_2$ in §5.3.

4. Completion of the feature curves and their surroundings

For each feature curve, the user defines its endpoints on the boundary of the hole. Our algorithm first produces a suitable curve that connects these end-points (§4.1). Next, the curve's surroundings are constructed along it in a way that preserves smoothness at the hole's boundary and avoids twists along the curve (§4.2).

4.1. Curve construction

Given boundary conditions, the most common way to construct a curve is to use splines [Far93]. Splines are fast and easy to compute, but they are not always the curves preferred by the human visual system [HT12]. We seek a curve that can be specified as easily, and yet is more appealing.

Ullman [Ull76] and Knuth [Knu79] showed that there are several properties that are important for the appeal of a curve. These include invariance to similarity transformation, symmetry, extensibility, smoothness, and roundness. We use the *3D Euler Spiral* [GXH01, HT12], which satisfies these properties. This curve is defined as a curve for which the curvature κ and the torsion τ change linearly along the curve:

$$\kappa(s) = \kappa_0 + \Delta\kappa \cdot s, \quad \tau(s) = \tau_0 + \Delta\tau \cdot s,$$

where s is the arc-length parametrization and $\kappa_0, \Delta\kappa, \tau_0$ and $\Delta\tau$ are constants.

Fig. 4 illustrates our interface: the user marks only four points on a broken feature line. Two of the points marked by the user are the end points, $\mathbf{x}_0$ and $\mathbf{x}_f$. The other two points help the system to determine the tangents, $\mathbf{t}_0$ and $\mathbf{t}_f$, at the end points. This interface is very intuitive and easy to use. In practice, we use Euler spirals to complete ridges and valleys [OBS04] that cross the hole.

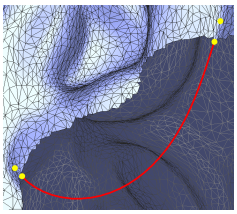


Figure 4: ***Curve completion.*** *Given the user's end points (yellow), our algorithm computes a 3D Euler Spiral.*

Given the boundary conditions, the Euler curve $\gamma(s)$ that starts at $\mathbf{x}_0$ with tangent $\mathbf{t}_0$ and minimizes both the difference between the curve's position at $s = L$ (L being the length of the curve) and $\mathbf{x}_f$, and the difference between the curve's tangent at $s = L$ and $\mathbf{t}_f$, is sought:

$$e = \|\gamma(L) - \mathbf{x}_f\|^2 + \|\mathbf{t}(L) - \mathbf{t}_f\|^2.$$

We follow the gradient-descent method of [HT12], which minimizes e, by finding the parameters, $\kappa_0, \Delta\kappa, \tau_0, \Delta\tau$ and L, of the desired curve.

4.2. From a feature curve to triangle strips along it

Given a 3D Euler Spiral, our goal is to construct a sub-mesh, which consists of two triangle strips, along it (one on each side of the curve). The question is how to define the locations of the additional vertices. We define a frame, a coordinate system, that traverses along the spiral, in which the newly-introduced vertices reside. On this frame, we enforce two constraints. First, the strips should connect smoothly to the hole's boundaries. Second, the strips should have minimal twist. Once the frame is set, we define the positions of the new vertices at this frame in the proximity of the spiral and build strips between consecutive vertices.

The standard approach to define a set of frames along a given curve is to use the Frenet frame [dC76]. The problem here is that twists might result, which could cause self intersections. Another option is to use the Bishop frame [Bis75, WJZL08]. This frame avoids twists, but the strips might not connect smoothly to both sides of the hole. We propose a variation on the Bishop frame, which avoids this pitfall, as explained below.

Bishop frame-based triangle strips: Our frame's axes should coincide both with the spiral's tangents and with the normals to the surface at the two end-points of the curve, $\mathbf{v}_0$ and $\mathbf{v}_L$. This is done by defining two Bishop-frames and interpolating between them, as follows. Let us denote a Bishop framed curve as $\{\gamma; \mathbf{t}, \mathbf{m}_1, \mathbf{m}_2\}$, where γ indicates the curve along which the axes "move", $\mathbf{t}(s)$ is the tangent of $\gamma(s)$, and $\mathbf{m}_1(s)$ and $\mathbf{m}_2(s)$ are perpendicular to $\mathbf{t}(s)$ and to each other. In our solution, the first frame, $\Gamma_{B1} = \{\gamma; \mathbf{t}, \mathbf{m}_1, \mathbf{m}_2\}$, coincides with the curve's tangent and normal to the surface at $\mathbf{v}_0$. The second frame, $\Gamma_{B2} = \{\gamma; \mathbf{t}, \mathbf{n}_1, \mathbf{n}_2\}$, coincides with the curve's tangent and normal at $\mathbf{v}_L$. Note that the two frames share one axis—the $\mathbf{t}$ axis.

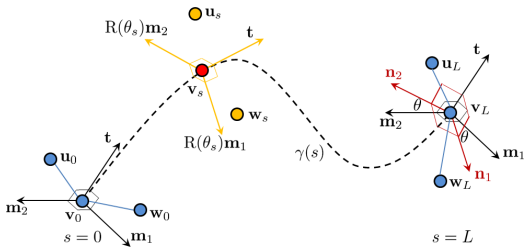


Figure 5: ***Bishop frames along the curve.*** *A Bishop frame that is assigned at $s = 0$ (black) might be twisted relatively to a Bishop frame that is assigned at $s = L$ (red). The existing vertices on the hole's boundary are marked in blue. The newly defined vertices $\mathbf{u}_s$ and $\mathbf{w}_s$ (orange) are defined using our interpolated frame (orange) at $\mathbf{v}_s$ (red).*

To interpolate between the two frames, Γ_{B1} and Γ_{B2}, we compute the angle θ about $\mathbf{t}$ between the two frames at s=L (see Fig. 5). We then define the interpolated frame as

$$\Gamma_{Bs} = \mathrm{R}(\theta_s)\Gamma_{B1},$$

where R is a rotation matrix about $\mathbf{t}\&\theta_s = \alpha_s\theta$, $0 \le \alpha_s \le 1$.

This frame twists around $\gamma(s)$ only as needed to bridge the gap (θ) between the frames Γ_{B1} and Γ_{B2} at s=L. Moreover, the definition of θ_s assures that if the arc-length s is sampled uniformly, then θ_s changes uniformly; thus, the twist is spread uniformly along the curve. It should be noted that this solution is equivalent to the discrete parallel transport presented by Bergou et al. [BWR*08], which minimizes the elasticity along a rod.

Let $\mathbf{u}_0$ and $\mathbf{w}_0$ be $\mathbf{v}_0$'s two adjacent vertices on the boundary of the hole (Fig. 5) and $\mathbf{u}_L$ and $\mathbf{w}_L$ be $\mathbf{v}_L$'s adjacent vertices. Having defined a frame with a minimal twist along $\gamma(s)$, we should build a strip with a smooth connection to the hole's boundary. To do so, using a bilinear interpolation, we set the positions of the newly introduced vertices $\mathbf{u}_s$ ($\mathbf{w}_s$) next to each vertex $\mathbf{v}_s$ on $\gamma(s)$ as:

$$\begin{aligned}
\mathbf{u}_s \cdot \mathbf{t}(s) &= (1-\alpha_s)(\mathbf{u}_0 \cdot \mathbf{t}(0)) + \alpha_s(\mathbf{u}_L \cdot \mathbf{t}(L)) \\
\mathbf{u}_s \cdot \mathrm{R}(\theta_s)\mathbf{m}_1(s) &= (1-\alpha_s)(\mathbf{u}_0 \cdot \mathbf{m}_1(0)) + \alpha_s(\mathbf{u}_L \cdot \mathbf{n}_1(L)) \\
\mathbf{u}_s \cdot \mathrm{R}(\theta_s)\mathbf{m}_2(s) &= (1-\alpha_s)(\mathbf{u}_0 \cdot \mathbf{m}_2(0)) + \alpha_s(\mathbf{u}_L \cdot \mathbf{n}_2(L)).
\end{aligned}$$

Finally, we build two triangle strips between consecutive vertices, one on each side of the curve.

5. From two triangle strips to hole filling

Given a curve and two triangle strips along it, our goal is to complete the remaining holes so as to minimizes (1). An

initial smooth completion is constructed for each hole using the advancing-front method of Zhao et al. [ZGL07]. Then, for each patch on this initial completion T_i, a more suitable patch from the remaining of the object is found and replaces that patch. We found that simply selecting the "best" patch might lead to erroneous completion. Instead, we first find several candidate patches and then select the best one (§5.1). Patches in the vicinity of the completed curve (T_F) get special care (§5.2). Finally, the positions of the vertices on the completion are modified, so as to decrease (1) (§5.3).

5.1. Candidate selection of similar patches

Our goal is to replace each initial target patch with a better one. Since the initial completion is smooth and inaccurate, local information cannot be trusted. Even more global descriptors, such as the Global Point Signature (GPS) [Rus07] or the Heat Kernel Signature (HKS) [SOG09], do not suffice. This is so since, as will be discussed in §5.2, patches that lie on the user's curve should maintain spatial coherence, and this information is not necessarily captured by the descriptor. Therefore, rather than selecting a patch whose descriptor is the most similar, we view the good matches as "hints" regarding the locations where the best patch may be found. In what follows, we briefly describe the patch descriptor and the dissimilarity measure we use.

Patch descriptor: The HKS descriptor [SOG09] depicts the diffusion of heat across the surface over time. The heat values over different times provide an effective multi-scale signature for shape matching. We utilize the HKS-based patch descriptor of [HTG14], described hereafter. We define a patch around vertex $\mathbf{v}$ to be the connected neighborhood that falls within a ball of radius R centered at $\mathbf{v}$. Its descriptor consists of two parts. The first is the average of the HKS values of the patch's vertices. Averaging is performed for each time t, resulting in a vector of averages (HKS_μ). The second part is the variance of these values, resulting in a vector of variances (HKS_{σ^2}). Given a patch P_i, its descriptor is hence

$$\mathrm{HKS}(P_i) = \{\mathrm{HKS}_\mu(P_i)_{[0,1]}, \mathrm{HKS}_{\sigma^2}(P_i)_{[0,1]}\},$$

where the notation $[0,1]$ indicates that each part is normalized separately to $[0,1]$, by setting the L_∞-norm to 1.

The question is which time domain to use, as different time domains reflect on the range in which the two patches match. We use four time domains, which were found empirically to be beneficial for normalized objects: the entire time domain and its three quartiles. Thus, for each patch we have four descriptors, each is compared only to descriptors that have the same time domain.

Dissimilarity measure: We define the dissimilarity measure between T_i and S_j as

$$||\mathrm{HKS}_\mu(T_i) - \mathrm{HKS}_\mu(S_j)||^2 + ||\mathrm{HKS}_{\sigma^2}(T_i) - \mathrm{HKS}_{\sigma^2}(S_j)||^2.$$

Candidates selection: Given a target patch $T_i \in T$, for each of its four descriptors, we search for the k most similar source patches $S_j \in S$, $j = 1,...,k$ (we use $k = 0.1\%$ of total source vertices). This set of candidate patches is denoted as $\mathcal{S}(T_i)$. We discuss the selection of the best patch among these candidates in §5.3.

5.2. Matching feature curves

Our aim is to reconstruct a broken feature curve and its surroundings, by using a similar curve from the source. For that, we first find for each feature curve and its neighboring patches a matching curve with its neighboring patches from the source as we elaborate below. Next, using the matched patches, the feature curve is modified together with the entire completed region (§5.3).

One way to find a matching curve from the source would be to apply some curve detection algorithm on surfaces and use the most similar curve. This approach was not found to be beneficial, as these algorithms usually produce short jagged curves. Moreover, this solution ignores the shape of the surroundings of the curves. Instead, we consider the patches in the vicinity of the spiral.

We are given the 3D Euler Spiral γ_T and suppose that we are also given a candidate matching source curve γ_S on the object (we will discuss later how to construct γ_S). We measure the dissimilarity between the two curves, γ_T and γ_S, by considering the alignment error not only between them, but also between patches in their vicinity.

Specifically, let $T_i \in T_F$ be a target patch whose center lies on the given curve γ_T and $S_j \in \mathcal{S}(T_i)$ be its matching source patch. In addition, let $\mathfrak{D}(\gamma_i, \gamma_j)$ be a scalar-valued *dissimilarity metric* between two curves and $\mathbb{D}(T_i, S_j)$ be a scalar-valued *dissimilarity metric* between two patches. We define our similarity error to be

$$\mathcal{E}(\gamma_T, \gamma_S) = \mathfrak{D}(\gamma_T, \gamma_S) + \frac{1}{N_{T_F}} \sum_{T_i \in T_F} \min_{S_j \in S_F} \mathbb{D}(T_i, S_j). \quad (2)$$

The dissimilarity between two curves is defined as follows. Let $(\mathrm{R}, \mathbf{t})$ be the rigid transformation that aligns the two curves, $\mathbf{v}_i \in \gamma_T$ be the sample points on γ_T, l_T be the number of sample points on γ_T, and $\mathbf{v}_j \in \gamma_S$ be the sample points on γ_S. $\mathfrak{D}(\gamma_T, \gamma_S)$ is defined as

$$\mathfrak{D}(\gamma_T, \gamma_S) = \frac{1}{l_T} \sum_{\mathbf{v}_i \in \gamma_T} \min_{\mathbf{v}_j \in \gamma_S} ||(\mathrm{R}\,\mathbf{v}_j + \mathbf{t}) - \mathbf{v}_i||^2.$$

The dissimilarity between the curves' surroundings is defined using the patches whose centers lie on the curves. Let $\mathbf{v}_k^{T_i}$ be a vertex of T_i, $\mathbf{v}_k^{S_j}$ be its corresponding point on S_j under $(\mathrm{R}, \mathbf{t})$ that aligns γ_S with γ_T, and N_i be the number of vertices in T_i. The dissimilarity between the patches, $\mathbb{D}(T_i, S_j)$, is defined as:

$$\mathbb{D}(T_i, S_j) = \min_{\mathrm{R}^*} \frac{1}{N_i} \sum_{k=1}^{N_i} ||(\mathrm{R}^*\,\mathbf{v}_k^{S_j} + \mathbf{t}^*) - \mathbf{v}_k^{T_i}||^2,$$

where $\mathbf{t}^*$ is the translation vector from the center of S_j to the center of T_i. Here, the closest source patch S_j is the one whose center has the smallest L_2-norm to the transformed center of T_i under $(\mathrm{R},\mathbf{t})$.

What is this γ_S? Recall that in §5.1, we already found for each target patch $T_i \in T$ several candidate source patches $\mathcal{S}(T_i)$. Algorithm 2 uses this set to find a matching source curve that minimizes (2). This procedure iterates over four steps, decreasing (2) at each iteration: First, we find for each T_i the set of transformations that aligns its center to the centers of its matching $\mathcal{S}(T_i)$. Next, using these transformations, we define the candidate matching curves. Then, among these, we select the curve that minimizes (2). Finally, we update the set of candidate patches. We elaborate below.

Algorithm 2 Our feature curve matching algorithm

1: While $\mathcal{E}(\gamma_T,\gamma_S)$ decreases
2: Find several rigid xforms using $(T_i,\mathcal{S}(T_i))$ matches
3: Define candidate curves, γ_S's, using the above xforms
4: Select the γ_S, among the candidates, which min. $\mathcal{E}$
5: Update $\mathcal{S}(T_i)$

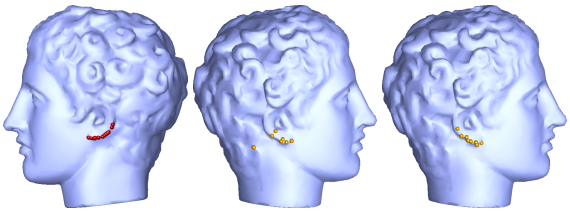
Transformation selection: We use the centers of $\mathcal{S}(T_i)$, $\mathbf{v}_c^{\mathcal{S}(T_i)}$ as "guesses" for the positions of points on possible source curves. These are reasonable guesses, since the patches in $\mathcal{S}(T_i)$ were found to match portions of the query curve and its surroundings. Yet, these guesses might not necessarily be spatially coherent, thus might not form a curve.

To find a set of spatially coherent vertices, we search for a transformation that suits several "guesses." For each "guess," we randomly select three neighboring guesses (in a ball with radius $5|e_{avg}|$, where $|e_{avg}|$ is the averaged edge length), along with their matching target vertices (the center of each T_i). Therefore, we have four source vertices (centers of patches in $\mathcal{S}(T_i)$'s) and four matching target vertices. We find the rigid transformation that aligns these four pairs of vertices, by minimizing the MSE between the positions of the transformed target vertices and the positions of their matching source vertices. We regard this transformation as the one that transforms γ_T to the curve γ_S that will be described next.

Candidate selection of source features: Given the curve γ_T and the found transformation, the matching source curve γ_S is defined to be the set of ridge (valley) vertices that are closest to the vertices of the transformed γ_T.

Best curve selection & update to the set of candidates: Among the candidate curves, we choose the one that minimizes (2). Once this curve is selected, the set of candidates $\mathcal{S}(T_i)$ is updated. This is done by finding, for each transformed target patch, its k nearest source patches (we use $k = 0.1\%$ of total source vertices). These are the ones whose centers are ridges (valleys) and have the smallest L_2-norm to the transformed center of T_i under the transformation that aligns γ_T with the selected γ_S.

Evaluation: Fig. 6 compares our resulting matching curve to the best matching patches found by our descriptor (§5.1). As can be seen, the matches are highly improved and indeed are spatially coherent.



(a) $\mathbf{v}_c^{T_i}$ (b) naïve $\mathbf{v}_c^{\mathcal{S}(T_i)}$ (c) our $\mathbf{v}_c^{\mathcal{S}(T_i)}$

Figure 6: ***Our curve (& surroundings) matching algorithm produces spatially-coherent matches.*** *(a) Input. (b) Using our patch descriptor, some of the matches are wrong. (c) Spatial coherence improves the results considerably.*

5.3. Error minimization

Given the set of candidate similar patches for each $T_i \in T$, we should select among them the best one. We then use it to decrease (1), by moving the vertices to new positions that better fit the selected matched patches.

After applying §5.1-5.2, (1) becomes

$$\varepsilon(T,S) = \frac{1}{N_T - N_{T_F}} \sum_{T_i \in T \setminus T_F} \min_{S_j \in \mathcal{S}(T_i)} \mathcal{D}_1(T_i,S_j) + \frac{1}{N_{T_F}} \sum_{T_i \in T_F} \min_{S_j \in \mathcal{S}(T_i)} \mathcal{D}_2(T_i,S_j). \quad (3)$$

We next define the dissimilarity metrics $\mathcal{D}_1$ and $\mathcal{D}_2$, which are used to find the rigid transformation that best aligns the patches. $\mathcal{D}_2$ is used for patches whose centers lie on the curve, thus it should comply with some global rigid transformation, whereas $\mathcal{D}_1$ is used for all the other patches.

To define $\mathcal{D}_1$, we use the HKS descriptor, which was already calculated in §5.1, to find for each vertex in T_i ($\mathbf{v}_k^{T_i}$) its corresponding vertex in $S_j \in \mathcal{S}(T_i)$ ($\mathbf{v}_k^{S_j}$). Using these pairs, we calculate the rigid transformation between T_i and S_j by solving the following least-squares minimization problem:

$$\mathcal{D}_1(T_i,S_j) = \min_{\mathrm{R}} \sum_{k=1}^{N_i} ||(\mathrm{R}\,\mathbf{v}_k^{S_j} + \mathbf{t}) - \mathbf{v}_k^{T_i}||^2, \quad (4)$$

where $\mathbf{t}$ is the translation vector from the center of S_j to the center of T_i.

For $\mathcal{D}_2$ we use the transformation of the curve found in §5.2, which returns for each vertex of T_i ($\mathbf{v}_k^{T_i}$) its corresponding vertex in S_j ($\mathbf{v}_k^{S_j}$) (N_i is the number of vertices in T_i):

$$\mathcal{D}_2(T_i,S_j) = N_i \mathbb{D}(T_i,S_j), \quad (5)$$

Having defined $\mathcal{D}_1$ and $\mathcal{D}_2$, Algorithm 3 is utilized to minimize (3). The algorithm iterates on two steps, while decreasing (3): First, for each target patch T_i, its most similar source patch $S_j \in \mathcal{S}(T_i)$, the one that minimizes either $\mathcal{D}_1$ or $\mathcal{D}_2$, is found. Second, the positions of the vertices are refined to reduce (3), as explained below.

Algorithm 3 Our error minimization algorithm

1: While $\varepsilon(T,S)$ decreases
2: For each target patch T_i:
3: Find the most similar patch in $\mathcal{S}(T_i)$
4: Update the positions of the vertices in the filled hole

Surface update: Given a corresponding source patch for each target patch, our goal is to move each of the target vertices to a new position, which decreases (3). Let $\mathbf{v} \in T_i$ be a vertex in the filled hole, $T^{\mathbf{v}}$ be the set of patches that contain $\mathbf{v}$, and $S^{\mathbf{v}}$ be the set of their most similar source patches. For each patch $S_k \in S^{\mathbf{v}}$, we identify an *anchor point* $\mathbf{u}_k$ that best fits $\mathbf{v}$ when S_k is aligned to $T_k \in T^{\mathbf{v}}$. Note that $\mathbf{u}_k$ is a point lying on the surface, but not necessarily a vertex. Let $\tilde{\mathbf{u}}_k$ be the anchor point after applying the rigid transformation of (4) or (5) (depending whether the patch is on the curve or not) that aligns S_k to T_k.

The contribution of $\mathbf{v}$ to (3) is derived from its contribution to (4) or (5) for all $T_k \in T^{\mathbf{v}}$:

$$\varepsilon_{\mathbf{v}} = \frac{1}{N_T} \sum_{k=1}^{K} (\mathbf{v} - \tilde{\mathbf{u}}_k)^2.$$

Minimizing this equation yields the locally optimal position

$$\mathbf{v}_d = \frac{1}{K} \sum_{k=1}^{K} \tilde{\mathbf{u}}_k. \tag{6}$$

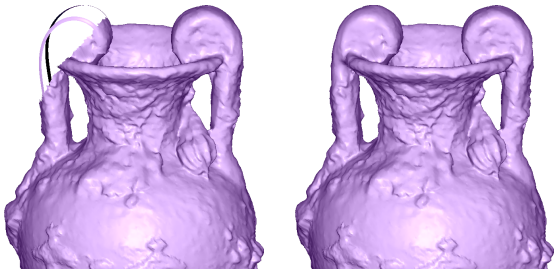
This solution, however, is overly smooth. To solve this pitfall, we replace the average in (6) with a weighted average:

$$\mathbf{v}_d = \frac{1}{\sum_{k=1}^{K} \omega_k} \sum_{k=1}^{K} \omega_k \tilde{\mathbf{u}}_k + \delta\mathbf{v}. \tag{7}$$

Here, the more similar S_k and T_k are, the larger ω_k is. $\delta\mathbf{v}$ enforces smoothness on the boundary of the hole.

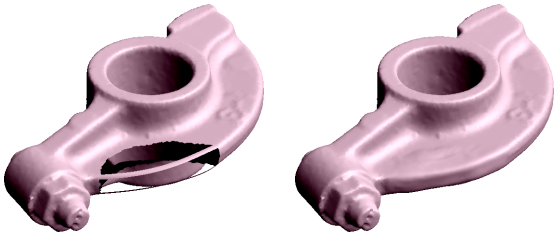
6. Results and implementation

Qualitative evaluation: Figs. 1, 7-13 present our results on several challenging cases, which have irregular 3D textures. In Fig. 1(f) we complete a broken ear of a head model. This case is difficult, as the ear should not be connected to the near-by hair. Moreover, as the curls differ, the neighborhood of the broken ear should not be copied from the neighborhood of the other ear. We compare our result to that of [HTG14] and to our modification to [HTG14], where a user's curve is used to get a better initialization. Our result is evidently better than these two.



(a) hole & strips (b) our completion

Figure 7: ***Completing an asymmetric amphora***



(a) hole & strips (b) our completion

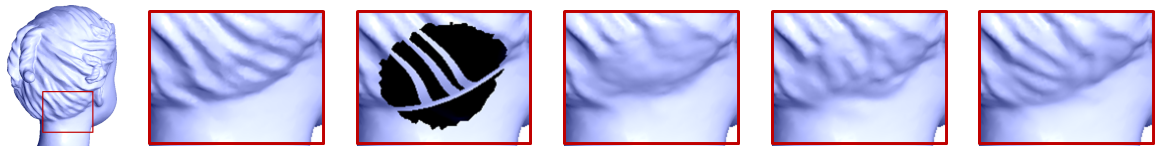
Figure 8: ***Completing a hole on a CAD model***

Fig. 7 displays our result on an asymmetric, very noisy, amphora. Despite the asymmetry, our completion manages to nicely complete the broken handle. Fig. 8 shows our completion on a CAD model, where the curved hole was nicely maintained thanks to the curves defined by the user. Fig. 9 illustrates a case of an irregular pattern, where there is no region on the model to copy from. The user's input guides the reconstruction of the hairdo and its salient features. Fig. 10 shows another example of an asymmetric model with irregular patterns, for which our algorithm manages to complete the hole in an appealing manner.

Figs. 9-10 also provide comparisons of our results to those of [HTG14] and our modification to it (a better initialization that considers the input curves). Our completions outperform the others, as they look more similar to the original object.

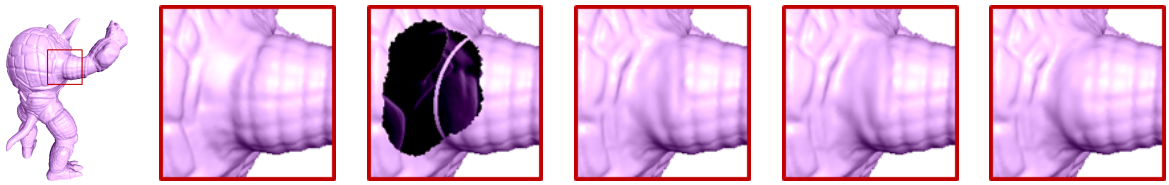
Figs. 11-12 present our completions of real scanned archaeological artefacts. These models are challenging not only because they are asymmetric (they are hand-made), and thus the valleys differ from each other and the circles are not exactly circular, but also because they are noisy. Nevertheless, our algorithm produces compelling completions, which nicely capture the shape of the missing region.

Quantitative evaluation: When coming to evaluate a completion quantitatively, one should recall that there is no "correct" completion, only one that looks appealing (as the information is indeed missing). How can one provide evaluation to "visually good"? One option is to simply compare the



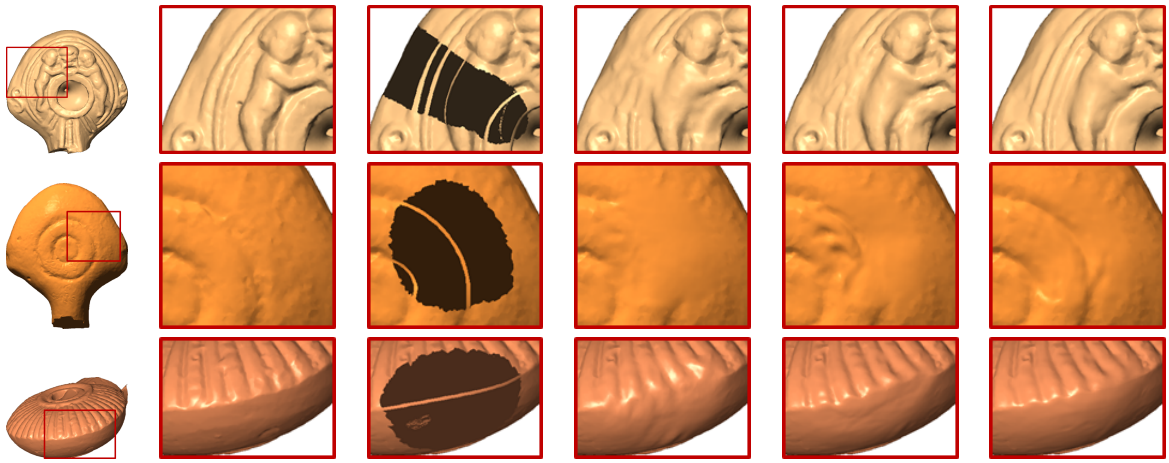
(a) original object | (b) zoom-in of original | (c) hole & strips | (d) context-based without assistance | (e) context-based with assistance | (f) our completion

Figure 9: ***Completing an irregular 3D pattern.*** *Our algorithm completes the hairdo while maintaining the feature lines (f). It is also compared to the completions of [HTG14] (d-e).*



(a) original object | (b) zoom-in of original | (c) hole & strips | (d) context-based without assistance | (e) context-based with assistance | (f) our completion

Figure 10: ***Completing a hole on the armadillo model.*** *Our completion of the missing pattern does not smooth the features.*



(a) original object | (b) zoom-in of original | (c) hole & strips | (d) context-based without assistance | (e) context-based with assistance | (f) our completion

Figure 11: ***Completing manually-broken Hellenistic oil lamps.*** *Our algorithm captures the structure of the missing regions.*

results to the original object (e.g., using the Hausdorff distance). However, such comparisons provide results that are counter-intuitive, i.e., smooth completions get better numbers than those that maintain the feature lines.

Instead, we suggest to evaluate the "characteristics" of the patterns, i.e., determine whether the completed pattern is acceptable in terms of the patterns that can be found elsewhere on the object. To do so, we calculate the mean distances between the maximum curvature directions of the two closest points on the original (unbroken) object and its completion.

Table 1 compares our results to those of Harary et al. [HTG14], with and without using the user's input. Our results are consistently better, except for Fig. 11 (bottom). However, visual inspection indicates that the completions are competitive in this case.

Implementation details: Our error minimization algorithm is accelerated by a coarse-to-fine approach, where larger neighborhoods are first used to reduce the low frequency error, and smaller neighborhoods then reduce the high frequency error. Our implementation employs two scales. This is done by simplifying the object to $\frac{1}{50}$ of its size [GH97] and completing the hole, as explained in §5. Then, the result for the coarse object is mapped to the fine object and is regarded as the initialization. The algorithm of §5 is then reapplied.

An important consideration is the size of the patches. Large patches are essential for capturing the low frequen-

Figure	context-based w.o.	context-based w.	Our result
1	0.21473	0.21484	**0.21442**
7	0.60755	0.02955	**0.02839**
8	0.04491	0.05003	**0.04297**
9	0.00978	0.0097	**0.00954**
10	0.00777	0.00789	**0.00773**
11 (top)	0.34401	0.34204	**0.33913**
11 (middle)	0.35598	0.35676	**0.35597**
11 (bottom)	**0.36197**	0.36304	0.36337

Table 1: ***Our completion is compared to others in terms of matching the "characteristics" of the patterns.*** *This table compares the mean distances between the maximum curvature directions for the objects shown in this paper. The blue cell indicates the best result.*

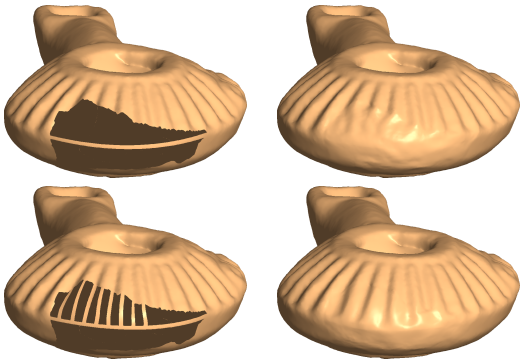
cies of the object, whereas small patches are important for capturing the high frequencies. Moreover, since all the computations in §5 depend on the size of the patch, larger patches result in a slower computation. For the coarse object, we use patches whose size is 1% of the object's surface area. For the fine object, the radius of a patch is set to $R = \left(\frac{1}{|e_{avg}|\cdot\mu} + 1\right) \cdot |e_{avg}|$, where μ is the average curvature of the patterns.

Running times: The algorithm was implemented in C/C++ and ran on a 2.67Ghz Intel i7-processor laptop with 4Gb of memory. The interaction time takes a few seconds, involving only the selection of 4 points per curve. The running time of the fine feature completion (§4) takes 1-2 seconds. The whole completion algorithm takes several minutes. For instance, on the head from Fig. 1, which consists of 106500 vertices, the algorithm runs for 9 minutes.

Limitations: Our method relies on the fact that the user can extract proper end conditions for the 3D Euler spiral. In addition, for very large holes, several feature curves may be necessary, which involves more interaction and thus might take longer. This is illustrated in Fig. 12 (top), which can be compared also to Fig. 11 (bottom), for which one curve suffices, as the hole is smaller. Moreover, there are cases where the 3D Euler spiral does not suit the model in hand (Fig. 13 (top)). In such cases, we let the user utilize a scaled Hermite spline (Fig. 13 (bottom)) instead. Finally, similarly to most earlier algorithms, ours does not guarantee the avoidance of self-intersection. However, in practice, we did not encounter intersections in the cases that we checked.

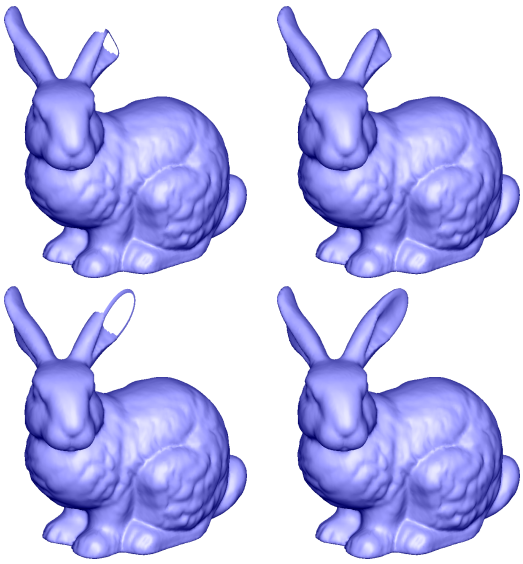
7. Conclusion

This paper presents a novel algorithm for the completion of large holes, which requires a little assistance from the user. Given four points for each curve the user specifies, our algorithm computes a curve that fits them—a 3D Euler spiral. This curve is then augmented by two triangle strips, which not only connect smoothly to the hole boundaries, but also do not twist. These strips are considered constraints to the completion algorithm that minimizes our global error. This minimization is performed by iteratively replacing each



(a) hole & strips (b) our completion

Figure 12: ***Limitation (1).*** *When the hole is very large, several curves should be specified.*



(a) hole & strips (b) our completion

Figure 13: ***Limitation (2).*** *The missing region might not be described well by a 3D Euler Spiral (top). In this case, we let the user specify a scaled Hermite spline instead (bottom).*

patch in an initial smooth completion, by a more suitable patch. An algorithm for choosing the "right" patch, which considers both similarity and spatial coherence, is described.

The quality of our completion is demonstrated via challenging cases, where the input models are asymmetric and contain irregular 3D patterns. These examples include both general objects and archaeological objects. The latter are especially interesting not only because of their importance, but also because archaeological objects are usually highly eroded, noisy, and contain large holes. We also provide quantitative comparison of our results to those of state-of-the-art algorithms. Both our qualitative evaluation and our quantitative one demonstrate that our algorithm manages to com-

plete holes that are very large, which previous algorithms could not handle in a satisfactory manner.

Acknowledgements This research was supported in part by the BSF 2012376, the ISF 1420/12, the Japan Technion Society, the NSF (Grants IIS-1319483, CMMI-1331499, IIS-1217904, IIS-1117257, CMMI-1129917, IIS-0916129), Intel, the Walt Disney Company, Autodesk, Side Effects, and NVIDIA. The models are courtesy of the Stanford 3D Scanning Repository (Bunny, Armadillo), the MIT CSAIL database (Head), and the AIM@SHAPE Shape Repository (Amphora, Bust, CAD model). We also thank Dr. A. Gilboa and the Zinman Institute of Archaeology at the University of Haifa for providing the archaeological models.

References

[Bar77] BARKER P.: *Techniques of Archaeological Excavation*. Batsford, 1977. 1

[BGK06] BENDELS G. H., GUTHE M., KLEIN R.: Free-form modelling for surface inpainting. In *Afrigraph* (2006), pp. 49–58. 2

[Bis75] BISHOP R. L.: There is more than one way to frame a curve. *The American Mathematical Monthly 82*, 3 (1975), 246–251. 4

[BPK05] BISCHOFF S., PAVIC D., KOBBELT L.: Automatic restoration of polygon models. *ACM Transactions on Graphics 24*, 4 (2005), 1332–1352. 2

[BSK05] BENDELS G., SCHNABEL R., KLEIN R.: Detail-preserving surface inpainting. In *VAST* (2005), pp. 41–48. 2

[BWR*08] BERGOU M., WARDETZKY M., ROBINSON S., AUDOLY B., GRINSPUN E.: Discrete elastic rods. *ACM Transactions on Graphics 27*, 3 (2008), 63: 1–12. 4

[CC08] CHEN C., CHENG K.: A sharpness-dependent filter for recovering sharp features in repaired 3D mesh models. *IEEE Transactions on Visualization and Computer Graphics 14*, 1 (2008), 200–212. 2

[CDD*04] CLARENZ U., DIEWALD U., DZIUK G., RUMPF M., RUSU R.: A finite element method for surface restoration with smooth boundary conditions. *Computer Aided Geometric Design 21*, 5 (2004), 427–445. 2, 3

[dC76] DO CARMO M.: *Differential geometry of curves and surfaces*. Prentice Hall, 1976. 4

[DMGL02] DAVIS J., MARSCHNER S., GARR M., LEVOY M.: Filling holes in complex surfaces using volumetric diffusion. In *3DPVT* (2002), pp. 428–441. 2

[Far93] FARIN G.: *Curves and surfaces for computer aided geometric design*. Springer-Verlag Berlin, 1993. 3

[GH97] GARLAND M., HECKBERT P.: Surface simplification using quadric error metrics. In *SIGGRAPH* (1997), pp. 209–216. 8

[GLWZ06] GUO T., LI J., WENG J., ZHUANG Y.: Filling holes in complex surfaces using oriented voxel diffusion. In *ICMLC* (2006), pp. 4370–4375. 2

[GSH*07] GAL R., SHAMIR A., HASSNER T., PAULY M., COHEN-OR D.: Surface reconstruction using local shape priors. In *SGP* (2007), pp. 253–262. 2

[GXH01] GUIQING L., XIANMIN L., HUA L.: 3D discrete clothoid splines. In *International Conference on Computer Graphics* (2001), pp. 321–324. 2, 3

[HT12] HARARY G., TAL A.: 3D Euler spirals for 3D curve completion. *Computational Geometry: Theory and Applications 45*, 3 (2012), 115–126. 2, 3, 4

[HTG14] HARARY G., TAL A., GRINSPUN E.: Context-based coherent surface completion. *ACM Transactions on Graphics 33*, 1 (2014), 5: 1–12. 1, 2, 5, 7, 8

[Ju04] JU T.: Robust repair of polygonal models. *ACM Transactions on Graphics 23*, 3 (2004), 888–895. 2

[Ju09] JU T.: Fixing geometric errors on polygonal models: a survey. *J. of Computer Science and Technology 24*, 1 (2009), 19–29. 2

[Knu79] KNUTH D.: Mathematical typography. *Bulletin AMS 1*, 2 (1979), 337–372. 3

[KS05] KRAEVOY V., SHEFFER A.: Template-based mesh completion. In *SGP* (2005), pp. 13: 1–10. 2

[Lie03] LIEPA P.: Filling holes in meshes. In *SGP* (2003), pp. 200–205. 2, 3

[OBS04] OHTAKE Y., BELYAEV A., SEIDEL H.: Ridge-valley lines on meshes via implicit surface fitting. *ACM Transactions on Graphics 23*, 3 (2004), 609–612. 4

[PMG*05] PAULY M., MITRA N., GIESEN J., GROSS M., GUIBAS L.: Example-based 3D scan completion. In *SGP* (2005), pp. 23: 1–10. 2

[PMV06] PERNOT J., MORARU G., VÉRON P.: Filling holes in meshes using a mechanical model to simulate the curvature variation minimization. *Computers & Graphics 30*, 6 (2006), 892–902. 2, 3

[Ros] ROSS T.:. http://www.tinaross.ca. 1

[Rus07] RUSTAMOV R. M.: Laplace-beltrami eigenfunctions for deformation invariant shape representation. In *SGP* (2007), pp. 225–233. 5

[SACO04] SHARF A., ALEXA M., COHEN-OR D.: Context-based surface completion. *ACM Transactions on Graphics 23*, 3 (2004), 878–887. 2

[SOG09] SUN J., OVSJANIKOV M., GUIBAS L.: A concise and provably informative multi-scale signature based on heat diffusion. *Computer Graphics Forum 28*, 5 (2009), 1383–1392. 5

[SYJS05] SUN J., YUAN L., JIA J., SHUM H.-Y.: Image completion with structure propagation. *ACM Transactions on Graphics 24*, 3 (2005), 861–868. 2

[TSS*11] TAKAYAMA K., SCHMIDT R., SINGH K., IGARASHI T., BOUBEKEUR T., SORKINE O.: Geobrush: Interactive mesh geometry cloning. In *Computer Graphics Forum* (2011), vol. 30(2), pp. 613–622. 2

[Ull76] ULLMAN S.: Filling-in the gaps: The shape of subjective contours and a model for their generation. *Biological Cybernetics 25*, 1 (1976), 1–6. 3

[VCBS03] VERDERA J., CASELLES V., BERTALMIO M., SAPIRO G.: Inpainting surface holes. In *ICIP* (2003), vol. 2, pp. 903–906. 2

[WJZL08] WANG W., JÜTTLER B., ZHENG D., LIU Y.: Computation of rotation minimizing frames. *ACM Transactions on Graphics 27*, 1 (2008), 2: 1–18. 4

[WLL*12] WANG X., LIU X., LU L., LI B., CAO J., YIN B., SHI X.: Automatic hole-filling of CAD models with feature-preserving. *Computers & Graphics 36*, 2 (2012), 101–110. 2

[ZGL07] ZHAO W., GAO S., LIN H.: A robust hole-filling algorithm for triangular mesh. *The Visual Computer 23*, 12 (2007), 987–997. 2, 3, 5