Computer Aided Geometric Design 35–36 (2015) 42–55

Contents lists available at ScienceDirect

Computer Aided Geometric Design

www.elsevier.com/locate/cagd

Poisson-driven seamless completion of triangular meshes

Marco Centin [a], Nicola Pezzotti [a,b], Alberto Signoroni [a,*]

[a] Information Engineering Dept. DII, University of Brescia, Italy
[b] Computer Graphics and Visualization group, Intelligent Systems, TU Delft, The Netherlands

ARTICLE INFO

Article history:
Received 24 February 2015
Accepted 11 March 2015
Available online 26 March 2015

Keywords:
Poisson Reconstruction
Restricted Delaunay triangulation
Hole filling
Mesh completion
Geometric modeling
3D scanning

ABSTRACT

The production of high-quality 3D mesh models has seen important technological advancements in recent years and is increasingly becoming a crucial asset for several application domains. However, the intrinsic nature of the problem (acquisition constraints and real objects complexity) makes hole filling and mesh completion still critical tasks for the effectual finalization of the models. In this work, we propose a new solution for filling holes and gaps and for reconstructing missing parts of mesh models that is capable to guarantee the full preservation of the input mesh. On the implicit function obtained from a Poisson Reconstruction computed on a set of oriented points derived from the input mesh, we develop an interpolation method which allows a fast and continuous surface-oracle computation which is exploited to efficiently guide a restricted Delaunay triangulation. This can be used to confine the tessellation inside the holed zones, while protecting the input mesh and the boundary curves so that simple tailoring routines can merge the obtained surface-reconstruction driven patches with the input mesh without stitching artifacts. The proposed technique is versatile and evidences its effectiveness in dealing with multiple complex holes, gaps and surface blending, and it compares favorably with respect to different, and usually more specialized, hole filling and global mesh repair techniques. As a parallel benefit, we also show how the same proposed solution can be exploited as an effective and high-quality meshing or remeshing tool.

1. Introduction

The way we are able to sense, acquire and process 3D shapes has been subject to relevant advancements in recent years. Modern 3D digitization technologies are becoming a primary asset in many demanding application domains (biomedical, cultural heritage, industrial design and manufacture, 3D printing, architecture and construction, entertainment industry, etc.) for their ability of effectively produce high-quality 3D digital models of real-world objects. However, because of the typical lack of complete visibility due to the complex shape of real objects or caused by specific acquisition setup constraints, even the most modern scanning technologies require *mesh completion* steps which take care of filling holes of various topologies (including multiple islands, gaps and cuts of various shapes). A real benefit comes when the mesh portions (patches) created by the completion tools integrate seamlessly with respect to the quality and type of triangulation of the input mesh, while preserving as much as possible the original geometry and connectivity. Hole filling can be thought either as a global operator (e.g. when the model is required to be watertight) or as a localized intervention (e.g. when only specific holes should be closed). For certain critical completion tasks (e.g. large holes or missing parts), there could also be the need to control or

* Corresponding author. Tel.: +39 030 3715 432; fax: +39 030 380 014.
E-mail address: alberto.signoroni@unibs.it (A. Signoroni).

http://dx.doi.org/10.1016/j.cagd.2015.03.006
0167-8396/

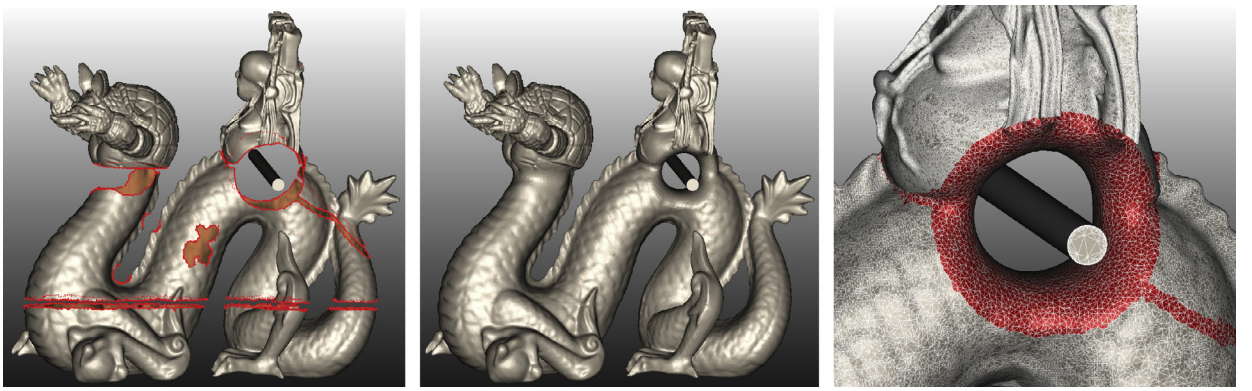


Fig. 1. A challenging completion example. The input (a) has many different configurations of boundaries: from simple holes to gaps, bridges, islands and a large central hole involving parts of different boundary curves whose topology has been suggested by adding a traversing cylinder. Our method is capable, without any additional user input, to create a natural completion (b) with smooth seamless transitions between the input meshes, which are fully preserved (c). The input mesh has 681k vertices, 1356k triangles and the algorithm terminates in 31 s on PC with an Intel® Core i7-4790 processor and 16 GB of RAM.

suggest the topology of the completion according to the specific modeling requirements. Fig. 1a is representative of a variety of different challenging needs. The example also includes, as a particular kind of mesh completion, the merging of different object parts that were separately acquired. In all cases, the finalization of the model requires an automatic reconstruction of the missing parts even if the underlying topology cannot be easily inferred by the mesh boundaries. Despite several mesh completion solutions have been proposed in the geometry processing literature, the achievement of a single tool capable to deal with all the above requirements still constitutes an open field of investigation. In this work, we propose a novel method which addresses all the above issues by introducing and exploiting an effective integration of a robust volumetric reconstruction with a high-quality mesh generation technique. A preview of the obtained results is given in Fig. 1b and 1c. Additionally, and remarkably, the very same proposed technique can be exploited as an effective tool for meshing and remeshing purposes (see Fig. 13 for a preview).

1.1. Related work

The task of filling holes has many overlaps with several chapters of 3D geometry processing, namely *Surface Reconstruction* (Berger et al., 2014), *Surface Meshing* and *Mesh Repairing* (Attene et al., 2013), depending on how the word "hole" is interpreted: a part of the input to be reconstructed, a new surface part to be meshed or some artificially produced gaps. We will mainly orient our interest to the case where holes are considered as parts of the model to be reconstructed, and we refer the reader to Attene et al. (2013) and Ju (2009) for a thorough review of mesh repair techniques. Hole-filling methods can be roughly classified into two categories: *volumetric* and *surface-based*. Methods are said to be *volumetric* (e.g. Davis et al., 2002; Bischoff et al., 2005; Podolak and Rusinkiewicz, 2005; Wu et al., 2008) whenever they rely on some inferred volumetric structure (a regular 3D grid, an octree, a tetrahedral subdivision or a more sparse volume sampling) which provides inside/outside information of the object. On the other hand, methods which primarily work with properties of the surface (such as boundary topology or curvature) are said to be *surface-based* (e.g. Liepa, 2003; Cohen et al., 2004; Varnuška et al., 2005; Zhao et al., 2007; Attene, 2010; Wang et al., 2012; Xie and Zou, 2013; Harary et al., 2014). Since our algorithm employs a Poisson reconstruction step, it can be considered volumetric. A method having some similar cues with respect to what we present here was developed by Podolak and Rusinkiewicz (2005), where the object volume is determined by an octree subdivision whose cells can be labeled as inside or outside (*atomic volumes*). Our technique can be considered an extension of this principle, where the labeling heuristics has been replaced with a surface reconstruction method used to drive the meshing of the hole patches.

Application requirements, especially in the context of demanding digitization pipelines, which justify the need of developing a new versatile method for the completion of triangular mesh models, and which constitute the highlights of the proposed solution, are listed and discussed below:

Input preservation A number of *surface reconstruction* techniques (e.g. Kazhdan and Hoppe, 2013; Süßmuth et al., 2010; Alliez et al., 2014; Kazhdan, 2005) naturally produce watertight reconstructions, and can therefore be thought as hole filling methods. However, the main weakness of using such approaches for hole filling is that a complete re-meshing of the input must be done as well, which alters the input and, due to the typical smoothing effects of these methods, may result in loosing sharp features and fine details. Contrarily, for acquired geometries, especially those produced by high-end 3D scanners, a fundamental requirement consists in the full preservation of the parts of the input mesh which are not involved in the completion or in other specific editing phases. Furthermore, since the completion is performed at a global scale, usually it is not possible for users to guide the methods to only work on some specified mesh portions.

Topological complexity handling Many hole filling methods are only suitable for filling *simple holes* (i.e. disk-shaped holes, enclosed by a single boundary loop). Most of the surface-based methods (Liepa, 2003; Cohen et al., 2004; Varnuška et al., 2005; Zhao et al., 2007; Wang et al., 2012; Xie and Zou, 2013) belong to this category. An exception is the MeshFix solution proposed by Attene (2010), which has proven to be a successful lightweight technique suited for small simple holes, but also capable to deal with some more complex apertures; its main limitation is that it assumes the target object as a single connected component and, even when this assumption is satisfied, its heuristics can fail in presence of large gaps, which can cause, as a side effect, the deletion of some parts from the original mesh. In addition, since the technique operates globally, it cannot be used selectively on specified input portions. In general, the limited capability to only handle simple holes severely restricts the possible applications to simple repairing routines, and this further motivated us in choosing a volumetric strategy. The problem of handling complex topological changes on triangular meshes can be found in some mesh editing (Pavić et al., 2010) and deformation (Wojtan et al., 2009; Müller, 2009) approaches. These works have some similar cues with what we obtain here. However, they often handle topological changes as critical configurations, where different mesh parts intersect, thus requiring tailored repair procedures.

Seamless completion Another important aspect is related to the capability of the filling technique of guaranteeing a seamless patching, entailing a pleasing transition between the original mesh and the generated patches, without noticeable stitchings on the boundaries. By using a volume-protection mechanism we insert and protect boundary vertices in the tessellation, so that the transition between the input and the patches is seamless. From another point of view, in order to obtain a visually-pleasing completion, contextual information could be taken into account. If the application domain is restricted to filling simple holes in CAD meshes, more specialized methods such as Wang et al. (2012) are able to also recover sharp features and corners across the hole patches (however this approach cannot be easily extended to holes with complex topology). Other techniques consider contextual information (automatically obtained, Sharf et al., 2004; Harary et al., 2014 or user-specified, Kraevoy and Sheffer, 2005) for reproducing features of the object. Despite in the present work we do not dig into these aspects, our method is also suitable to be used in combination with context-based approaches, which can be introduced either at a pre-processing level (feature insertions) or as post-processing optimization steps (similarly to Harary et al., 2014).

1.2. Overview of the method and contributions

Given an input mesh M and a set of boundaries of M, our method comprises three steps:

- *Reconstruction oracle*: a set of oriented points is sampled from M and used to compute a Poisson surface reconstruction (Kazhdan and Hoppe, 2013) from which an *implicit function*, represented by an octree discretization of the space and a set of radial basis functions centered in the octree nodes, is derived. We implemented a novel efficient interpolation technique on unconstrained octrees for generating a continuous *surface oracle* that can be exploited for fast querying the object volume within a restricted Delaunay mesh generation approach (Edelsbrunner and Shah, 1994; Amenta et al., 2000; Boissonnat and Oudot, 2006).
- *Confined tessellation of the hole zones*: we introduce a modified Delaunay refinement process that employs a generic protection volume that is used in order to set up a confinement of the tessellation in the holed zones. This way we are able to efficiently generate hole patches with seamless transitions and free of self-intersections.
- *Tailoring*: the hole patches are then merged into the input mesh by robustly matching the input and the completion boundary loops, which are stitched together in order to produce the final output.

The proposed method is capable to efficiently produce a (complete or user-guided) high-quality patching of the input mesh, by handling *complex topological configurations*, as the one depicted in Fig. 1a, and providing *input preservation* and satisfactory results (see Fig. 1b) while guaranteeing *seamless transitions* between the preserved original mesh and the generated completion surfaces, as depicted in Fig. 1c.

The rest of the paper is organized as follows: Section 2 first describes the tools constituting the core of our seamless completion method consisting in the synergic integration of Poisson reconstruction with a restricted Delaunay triangulations through a fast interpolated oracle and the use of protection volumes. Section 3 presents the whole pipeline and other implementation details about how to exploit the above technique in order to solve challenging mesh completion tasks. Section 4 shows and discuss the different possible applications and implications of the proposed technique, also providing comparisons with respect to other approaches to hole filling and mesh repair.

2. Delaunay meshing of Poisson reconstructions

Before describing the complete processing method (see Section 4), we here introduce the core aspects of the proposed integration of a Poisson reconstruction method into a Delaunay meshing procedure. At first, we revise some background about restricted Delaunay triangulations and introduce some notation (Section 2.1) allowing us to present our modified refinement strategy (Section 2.2). Then, after recalling the core ideas of Poisson Reconstruction (Section 2.3), we explain how we implemented a surface oracle which queries the solution octree using a fast and continuous interpolation (Section 2.4).

2.1. Restricted Delaunay triangulations

Delaunay triangulation (Delaunay, 1934) provides a canonical discretization of the space in simplices (triangles in $\mathbb{R}^2$, tetrahedra in $\mathbb{R}^3$, or n-simplices in $\mathbb{R}^n$). When dealing with discretization of 2D surfaces in $\mathbb{R}^3$, one can basically adhere to two different approaches: either extending planar methods to surfaces (like what has been done by Chew, 1993), or looking at surface meshes as subcomplexes of three-dimensional meshes. The latter approach has proven to be a successful way of dealing with quality surface meshing and is based on the notion of *restricted Delaunay complex*, whose definition and properties have been introduced by Edelsbrunner and Shah (1994). Approximating a surface can then be tackled by building a three-dimensional triangulation (composed of tetrahedra) and restricting the complex to the elements whose Voronoi cell intersects the surface. Under certain sampling conditions, the resulting restricted complex is homeomorphic to the surface, and guaranteed Hausdorff approximation of it has been proved first by Amenta et al. (2000), and then by Boissonnat and Oudot (2006), who also designed an elegant algorithm for quality mesh generation. One of the key concepts behind their method is that it only needs to access the surface through a *surface oracle*:

Definition 1. Let S be a C^2-continuous surface in $\mathbb{R}^3$ without boundaries. For all $\mathbf{p}, \mathbf{q} \in \mathbb{R}^3$, let $l(\mathbf{p}, \mathbf{q})$ denote the line segment identified by $\mathbf{p}$ and $\mathbf{q}$. The *surface oracle* of S is the function $\mathcal{S}: \mathbb{R}^3 \times \mathbb{R}^3 \to \mathscr{P}(\mathbb{R}^3)$ such that, for every $p, q \in \mathbb{R}^3$, $\mathcal{S}(\mathbf{p}, \mathbf{q}) = l(\mathbf{p}, \mathbf{q}) \cap S$, that is, $\mathcal{S}(\mathbf{p}, \mathbf{q})$ is the set of the intersections of the segment $l(\mathbf{p}, \mathbf{q})$ with S.

The Delaunay Iterative Refinement method is a greedy incremental strategy for constructing, given a C^2 surface S without boundaries, a *loose ϵ-sample* (Boissonnat and Oudot, 2006) E of S and a restricted Delaunay tessellation of E, denoted with $\mathrm{Del}|_S(E)$ which is a good approximation of S for sufficiently small ϵ. We now introduce some additional notation to facilitate a high-level description of the original algorithm and of our modified approach, while avoiding implementation-dependent details (Boissonnat and Oudot, 2006). We denote with $\mathcal{T}$ the restricted Delaunay tessellation which is iteratively updated by the algorithm, and with $\mathcal{S}$ the underlying surface oracle. The tessellation is built in two steps that can be summarized as follows:

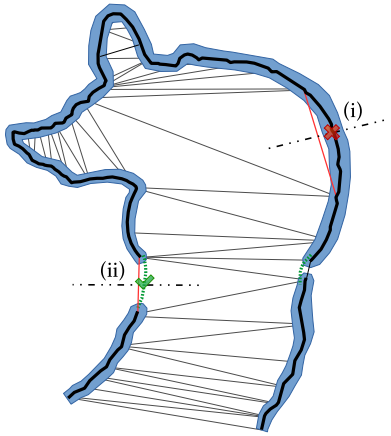
- Initial triangulation: an initial Delaunay tessellation $\mathrm{Del}|_S(E_0)$ is computed, based on some initial points E_0. These can be given as user input or randomly sampled by computing oracle intersections;
- Iterative refinement: elements of the triangulations are tested with respect to certain *quality criteria* $\mathcal{Q}$ about their shape and local approximation error (Jamin et al., 2014), and a list of *bad* elements $\mathcal{B}$ is produced. The triangulation is refined using the Bowyer–Watson algorithm (Watson, 1981; Hornus and Boissonnat, 2008) and the restricted complex $\mathcal{T}$, which contains the faces of the tetrahedra whose dual segment intersects the surface, is iteratively updated. At each step, the "worst" element is processed, and the algorithm terminates when there are no more bad elements (i.e. $\mathcal{B} = \emptyset$).

It follows that the output of the method is determined once that the oracle $\mathcal{S}$, the quality criteria $\mathcal{Q}$ and the initial point set E_0 are fixed, so we can write: $\mathcal{T} = \mathcal{D}(\mathcal{S}, \mathcal{Q}, E_0)$. Being a crucial input of the method and for the reconstruction to be effective, the oracle $\mathcal{S}$ must be generated according to *speed* and *continuity* requirements. Execution speed is important because of the high number of queries required during the refinement process and continuity is even more important (and cannot taken for granted in general) since inconsistent intersections produce surface cracks and affect the convergence. These requirements are easily satisfied for synthetic oracles, but become critical once that they involve complex data structures originating from other techniques. This is indeed the case here, as exposed in Section 2.4 where our effective oracle is presented. Finally, we observe that the method has an inherent global nature, in fact it is not possible to guide the refinement to work only within a specified zone in the space and there is no guarantee that the initial points E_0 are still belonging to the final triangulation $\mathcal{T}$. The latter problem has been addressed, in the context of generation of mesh with sharp features, by the *protecting balls* method (Cheng et al., 2007). In our setting, a different protection will be used in order to confine the tessellation to stay inside the holed zones, while blocking the refinement in proximity of the input mesh.

2.2. Confinement and protection volumes

Let us assume that an initial holed mesh M is given and that we have a surface oracle $\mathcal{S}$ which is a faithful approximation of M in proximity of the existing surface, while providing oracle intersections with an implicit surface in the holed zones. Our aim is to create a *completion mesh* M' by setting up a tessellation of the hole zones while blocking the refinement in proximity of the known mesh parts (in order to avoid their re-triangulation). Furthermore, if we were able to insert and protect the mesh vertices in a neighborhood of each boundary curve, then we could easily match them across M and M' and extract the hole patches to be inserted in M through simple tailoring routines. We can achieve both goals by extending the method through the definition of a *protection volume* $\mathcal{P}$ that blocks all the refinements within its interior. Fig. 2a presents a sketch of the described refinement procedure, while the algorithm in Fig. 2b provides a high-level description of the new method, which now also depends on $\mathcal{P}$, so that we can write $\mathcal{T} = \mathcal{D}(\mathcal{S}, \mathcal{Q}, \mathcal{P}, E_0)$.

A suitable protection volume to be used within the above modified Delaunay Refinement method can be represented by a *cushion* covering M, mathematically defined as the set: $\mathcal{P}_r(M) = \{\ \mathbf{p} \in \mathbb{R}^3 : |d(M, \mathbf{p})| < r\ \}$. $\mathcal{P}_r(M)$ can be practically implemented by considering a point-to-mesh distance and defining a scalar field by computing the distance to the r-cushion



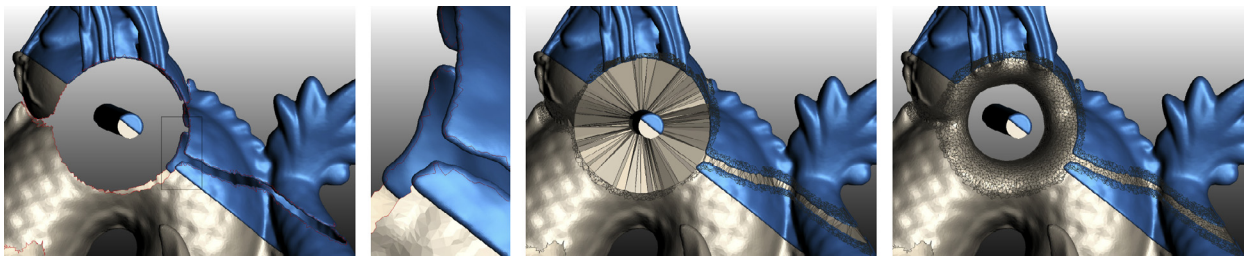
(a) Sketch of the modified refinement

Algorithm 1 Modified Delaunay Refinement

```
function T = D(S, Ω, P, E0)
    T ← initial tessellation of E0
    B ← bad elements of T with respect to Ω
    while B ≠ ∅ do
        t ← pull worst element from B
        s ← dual segment of t
        p ← intersection of s with S
        if p is outside P then
            refine T by adding p
            update B with new bad elements
        end if
    end while
end function
```

(b) Modified refinement procedure

Fig. 2. (a) Outline of the proposed refinement: by considering a thin cushion surrounding the input mesh as a protection volume we are able to limit the refinement of the known parts (i) and triangulate the holed zones (ii). At the same time we allow boundary vertices to be inserted and preserved, so that cutting out the hole patches is simplified. (b) High-level description of the proposed refinement method.



(a) M and a section of $\mathcal{P}$ (b) $\mathcal{P}_r(M)$ (c) Initial tessellation (d) Refinement $\mathcal{D}(\mathcal{S}, \Omega, \mathcal{P}, E_0)$

Fig. 3. Tessellation of complex holes: subfigure (a) shows a portion (in blue) of the protecting volume $\mathcal{P} = \mathcal{P}_r(M)$ and the input mesh M (in gray). Note how the boundaries (red lines) are fully incorporated in $\mathcal{P}_r(M)$ (b). The faces of the tetrahedra of the Delaunay tessellation of the initial points E_0 are shown in (c), while (d) shows the faces of the restricted complex after the modified refinement process. The thin cushion volume $\mathcal{P}_r(M)$ effectively protects the initial points in E_0 which are in proximity of M, confining the tessellation inside the hole zones. The Poisson-driven oracle correctly guides the refinement process to adhere to the topology of the reconstruction (images (c)–(d)). (For interpretation of the references to color in this figure, the reader is referred to the web version of this article.)

of the mesh (which can be obtained in an efficient way using space partitioning data structures, such as kD-trees). Fig. 3 explains this confinement applied to a single complex hole. In Fig. 3a and 3b the protecting volume is shown on top of the input mesh M. For visualization purposes only a portion of $\mathcal{P}_r(M)$ is displayed (in blue) to also reveal the underlying mesh (in gray). An initial tessellation of the input mesh is done by computing a Delaunay triangulation of the vertices extracted from M (Fig. 3c). The bad triangular elements of this tessellation are then iteratively refined using a Poisson-driven oracle. This leads to hole patches that properly infer the correct topology (Fig. 3d).

2.3. Poisson Reconstruction: background

The Poisson Reconstruction technique was introduced by Kazhdan et al. (2006). The method assumes that the object to be reconstructed is a smooth closed surface, that is the boundary of its volume. Sampled oriented points can be used to approximate the surface normals, which can be seen as the gradient of the characteristic function of the object (relaxed to be a smooth function). By the Divergence Theorem, this leads to a formulation of a Poisson problem, aiming at reconstructing a scalar function by knowing its Laplacian. In a first approach (Kazhdan, 2005), the system was discretized on a regular grid and solved in the Fourier domain. The high memory overhead of using a regular grid has then been addressed (Kazhdan et al., 2006) by discretizing the problem on a basis of compactly supported Radial Basis Functions, centered on the nodes of an adaptive octree. Similar approaches related to the creation of level-of-detail in geometric data have been proposed (Duguet et al., 2005). In Kazhdan and Hoppe (2013) a screening correction of the Poisson equation was introduced, which greatly improves the preservation of details in the reconstruction. As observed already back in the original approach, since Poisson Reconstruction provides a natural way of producing watertight models out of normal samples, it could be used for filling gaps and holes. However, its direct exploitation for this goal also produces a completely new mesh (generated with a marching-cubes-like method, Kazhdan et al., 2007) which will cause the loss of many input features, besides producing a

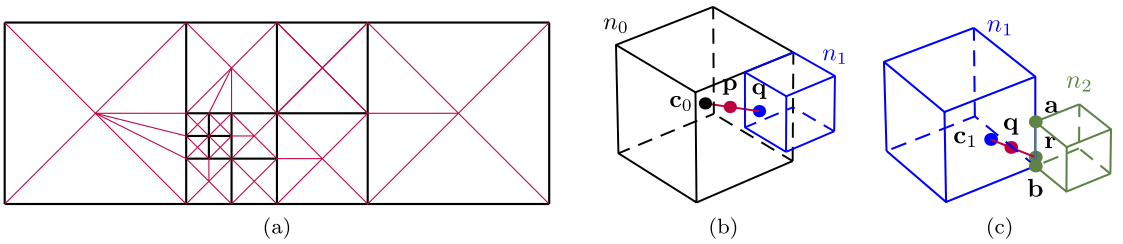


Fig. 4. (a) 2D-equivalent of the interpolation complex. (b) and (c) interpolation strategy: first, $\mathbf{p}$ is projected on a face of its leaf node (b); then an edge-projection seeks for (possibly) deeper neighboring corner values (c).

not well-shaped triangulation. Differently, we exploit the evaluation of a surface oracle which operates directly on the above reconstruction octree.

2.4. Poisson reconstruction surface oracle

As defined in Boissonnat and Oudot (2006) and Jamin et al. (2014), a *surface oracle* is a method for computing intersections between a segment (in space) and an implicit surface. Since the Poisson solution does not provide a true *signed distance field*, the most natural way of computing an intersection is to approximate it by using the *bisection method*. This approach reduces the problem to the one of being able to efficiently querying the solution field in a given space coordinate. A direct way of probing the Poisson field is to traverse the octree associated to the surface reconstruction and compute the contribution of the basis functions whose support contains the query point. Using the notations of Kazhdan et al. (2006), if $\mathcal{O}$ is the set of all octree nodes and F_n denotes the basis function centered in the node $n \in \mathcal{O}$, then $\tilde{\chi}(\mathbf{p})$, the solution field in $\mathbf{p} \in \mathbb{R}^3$ is:

$$\tilde{\chi}(\mathbf{p}) = \sum_{F_n \in \mathcal{F}_{\mathcal{O},F}} c_n F_n(\mathbf{p}) \tag{1}$$

where c_n denote the solution coefficients corresponding to $n \in \mathcal{O}$. Since the functions in $\mathcal{F}_{\mathcal{O},F}$ are compactly supported, the summation can be restricted to the nodes $n \in \mathcal{O}$ such that the support of F_n contains $\mathbf{p}$. This formula is however too computationally expensive to be used in our surface oracle, since it involves a lot of re-evaluations of the basis functions for many nodes (the support of F_n is a ball centered in n with radius 1.5 times the width of n). In order to obtain both accurate and fast interrogations of the solution field, we compute, in a pre-processing step, the values $\{v_n^{(0)}, \ldots, v_n^{(7)}\}$ of the solution field on the corners of n and the center value $v_{\mathbf{c},n}$ (which we approximate as the average of the corners). During this pre-processing, the coordinates and sizes of the octree nodes are mapped from the unit cube into world coordinates (Kazhdan et al., 2006), and the solution field is rescaled linearly so that the reconstruction iso-value is actually 0, while negative values identify the inside of the object and positive the outside part (Kazhdan and Hoppe, 2013). This effectively translates the Poisson field into a scalar field suited to be queried by an oracle that exploits the bisection method.

After this step we are left with the problem of defining a *continuous interpolation* over an octree with arbitrary topology. A similar problem was considered by Duguet et al. (2005), but their interpolation was based on center value only. Other approaches consider to use the generation of the *dual octree* (Lewiner et al., 2010) as a workaround. The main difficulty of this task is due to the fact that face-adjacent nodes can be at different depths in the octree. To address this issue we here introduce a fast continuous interpolation on the tetrahedral complex obtained by connecting the center of each node with the corners and face centers of the face-adjacent nodes at any depth, as depicted in Fig. 4a (where for visual clarity we used an analogous 2D complex on a quad-tree). The scalar field is then linearly interpolated inside each tetrahedra where, instead of explicitly building said complex, we rely on the following two lightweight projections (represented in Fig. 4b and 4c):

- Face projection: given a query point $\mathbf{p}$, we first look for the leaf node n_0 containing $\mathbf{p}$ (let $\mathbf{c}_0$ be its center and $v_{\mathbf{c}_0}$ its value). We compute the projection $\mathbf{q}$ of $\mathbf{p}$ on the n_0 cube from $\mathbf{c}_0$. $\mathbf{q}$ will fall on a face f_0 of n_0 (if the projection falls exactly on an edge or on a corner we conventionally choose one face), allowing us to look for a possibly smaller leaf node n_1 (Fig. 4b) by moving in the direction of the face normal. Consequently, let f_1 be the smallest face containing $\mathbf{q}$ and denote with $\mathbf{c}_1$ its center. In the next step we compute, by interpolation, the field value $v_{\mathbf{q}}$ in $\mathbf{q}$, so that $v_{\mathbf{p}}$ can be computed by linearly interpolating between $v_{\mathbf{c}_0}$ and $v_{\mathbf{q}}$.
- Edge projection: given the face-projection $\mathbf{q}$ and the face f_1, we observe that we can safely approximate the value $v_{\mathbf{c}_1}$ of the field in $\mathbf{c}_1$ as the average of the face corners (we already searched for the smallest leaf along the direction given by the normal to f_1). We then project $\mathbf{q}$ from the face center, obtaining a point $\mathbf{r}$ on some edge e (Fig. 4c). Note that every non-corner point on an edge is shared by maximum 4 cubes and, since we already touched two of them, we can search for the other two by following the direction determined by the projection and excluding the already visited cubes (this can be practically implemented by the opportune placement of two search point for likewise nearest-leaf queries). This will find a (possibly smaller) leaf node n_2 which has an edge containing $\mathbf{r}$. From the knowledge of the node n_2 we can retrieve the two innermost corner points $\mathbf{a}, \mathbf{b}$ along the edge e.

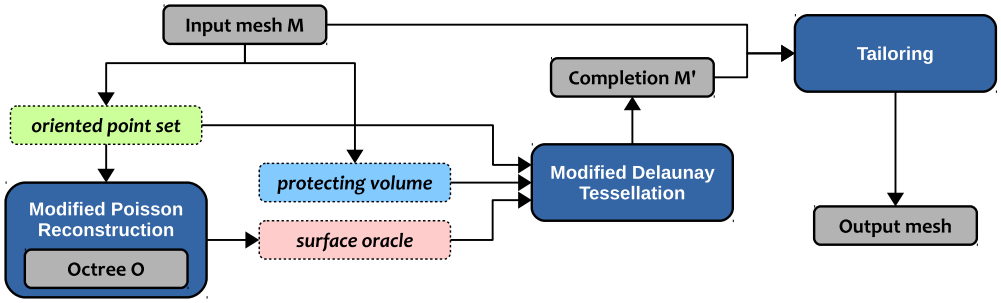


Fig. 5. Block diagram of the proposed method. (For interpretation of the references to color in this figure, the reader is referred to the web version of this article.)

The corners $\mathbf{a}, \mathbf{b}$, together with the face and node center, are the vertices of the interpolation tetrahedron and the value $v_{\mathbf{p}}$ is then computed by linear interpolating the corner values:

$$\begin{cases} v_{\mathbf{p}} = (1-\alpha)\cdot v_{\mathbf{c}_0} + \alpha \cdot v_{\mathbf{q}} \\ v_{\mathbf{q}} = (1-\beta)\cdot v_{\mathbf{c}_1} + \beta \cdot v_{\mathbf{r}} \\ v_{\mathbf{r}} = (1-\gamma)\cdot v_{\mathbf{a}} + \gamma \cdot v_{\mathbf{b}} \end{cases}, \tag{2}$$

where the coefficients α, β and γ are:

$$\begin{cases} \alpha = \|\mathbf{p}-\mathbf{c}_0\|/\|\mathbf{q}-\mathbf{c}_0\| \\ \beta = \|\mathbf{q}-\mathbf{c}_1\|/\|\mathbf{r}-\mathbf{c}_1\| \\ \gamma = \|\mathbf{r}-\mathbf{a}\|/\|\mathbf{b}-\mathbf{a}\| \end{cases}. \tag{3}$$

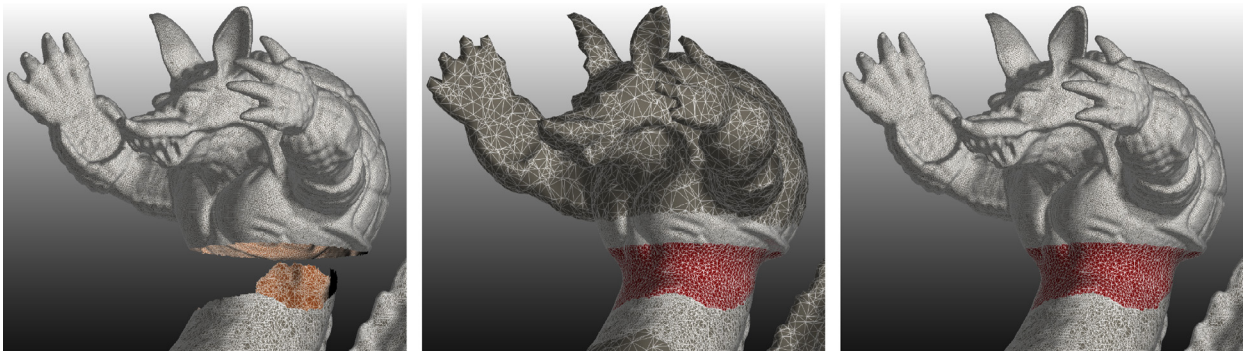
This defines a continuous field since it corresponds to a linear interpolation on a tetrahedral complex. The efficiency of this approach is given by the fact that we rely on two simple projections and a total of 4 nearest-leaf queries (one for the query point $\mathbf{q}$, one for finding the node n_1 and two for finding n_2).

3. Poisson-driven mesh completion pipeline

We are given an input mesh M, which comes from a high-quality object acquisition and surface reconstruction pipeline. M presents holes of various nature, caused by occlusions or missing data and it has been potentially edited by the user by approaching different parts which were acquired/reconstructed separately to compose the final model. Our goal is to finalize the mesh by creating completion patches for the various missing portions while fully preserving the input mesh. The zones to be completed are provided by the user as a selection of boundary curves (holes and gaps which are not selected by the user should not be filled). Fig. 5 provides a description of our mesh completion pipeline, where gray boxes represent the data involved, while blue boxes are the algorithms. The mapping between these elements (technically described in Section 2) is shown with arrows. The dotted-line boxes represent abstractions above the underlying data, which can be thought as "functors" acting inside each algorithm: the input mesh M virtually produces an oriented point set (which is the input of the Poisson reconstruction and acts as a set of initial points for the triangulation) and the protecting volume; similarly, the surface oracle is a functor which operates on the octree obtained in the reconstruction. The Delaunay refinement method is used in order to produce a *completion mesh* M', which is a mesh that provides a good representation of the selected boundaries and a tessellation of the missing parts (while the triangulation on the already known mesh portions remains intentionally coarse thanks to the effect of the protection volume). The "tailoring" step defines a heuristics for matching the selected boundary loops and defining a *cut* which isolates the hole patches. In the following we describe each step in detail while in Fig. 6 the main phases are visually represented.

3.1. Input requirements and preprocessing

Since we rely on being able to efficiently process the mesh data, both in a combinatorial and geometrical sense, the input mesh is assumed to be represented by an efficient data structure (in our implementation we used a half-edge data structure). Furthermore, M has to be equipped with a spatial sorting data structure for their vertices (e.g. a kD-tree or an octree). Finally, we assume the mesh to be oriented and manifold, which are standard requirements in this scenario. A pre-processing step must take care of these aspects if they were not satisfied. However, even manifold oriented meshes can have a large variety of degeneracies (Attene et al., 2013) and some care is required for the completion to succeed. Since the user input for our method is a set of boundary curves, the only additional assumption we make is that those boundaries are free of defects (such as topological noise). If this is not the case one can either apply an initial boundary fixing routine or request user intervention (which can be as simple as deleting an n-ring of the boundary). Degeneracies



(a) Input mesh M (b) Completion mesh M' (c) Final result

Fig. 6. The input mesh M to be completed (a) is used in order to create a *completion mesh* M' (b), obtained with a Delaunay tessellation of the reconstruction oracle $\mathcal{S}$. Observe how in M' only the completion zone is finely triangulated, while other parts remain coarse. An n-ring of the boundary curves (in the picture, $n = 16$) was inserted and protected in the tessellation. Sharing these vertices between M and M' simplifies the selection of the new faces (red color) and their final insertion in M (c). (For interpretation of the references to color in this figure legend, the reader is referred to the web version of this article.)

which are geodesically distant from the considered boundaries will have no effect on the result, thus they will not be considered nor corrected by our algorithm.

3.2. Building the reconstruction oracle

The *reconstruction oracle* $\mathcal{S}$ consists in an octree with corner value data equipped with the constant-time interpolation of the implicit function, as explained in Section 2.4. To compute it, a set of oriented points needs to be sampled from M and used to feed the Poisson multi-grid solver. $\mathcal{S}$ required to be well close to the mesh M in the proximity of the selected boundaries. The tolerance of such approximation is controlled by tuning the radius r of the protection volume $\mathcal{P}_r(M)$ and this requirement can be easily met by sampling enough mesh points in a geodesic neighborhood of the boundaries, which can be in turn approximated by an n-ring if the mesh is uniform enough (otherwise trivial iterative edge-split routines can be used for sampling a denser point set). In order to obtain a globally consistent reconstruction, we uniformly sample mesh points on each connected component of the input. This expedient has multiple beneficial effects:

- The surface oracle takes into account every connected component of the model. This provides a topologically aware global reconstruction (without self intersections);
- The topology of the completion can be guided by the insertion of auxiliary synthetic elements. For example, the dataset of Fig. 1a has 8 connected components comprising an artificially inserted cylinder having the role of suggesting and guiding the nearby "hole preserving" completion (see Fig. 1c);
- The point subsampling helps the zero level of the oracle to stay inside the protection cushion. This allows to take benefit from the above global reconstruction feature avoiding the cost of useless triangulation refinements (and related oracle computations) over already known portion of the data. Along with the already seen lightweight oracle computation, this constitutes another relevant factor for the computational efficiency of our mesh completion solution.

3.3. Generating the completion mesh

Once the reconstruction oracle $\mathcal{S}$ is ready, a completion mesh M' is generated by a restricted Delaunay triangulation. The tessellation $\mathcal{T} = \mathcal{D}(\mathcal{S}, \mathcal{Q}, \mathcal{P}, E_0)$ is determined by the following factors:

- The *surface oracle* $\mathcal{S}$. The main function of $\mathcal{S}$, described in Section 3.2, is to efficiently guide the refinement in the holed zones, while providing the proper topology.
- The *quality criterion* $\mathcal{Q}$. We defined a criterion based on the edge length, which is chosen heuristically in order to match an average edge length across the boundaries. One advantages of the proposed technique is that, if the hole patches need to have different edge properties (e.g. adaptive or very finely generated), it is sufficient to modify this criterion.
- The *initial points* E_0. We select from the input mesh M (Fig. 6a) and insert in the tessellation an n-ring of each boundary curve as well as a set of sparsely sampled mesh points. This allows to produce a good initial tessellation and to pick vertices on connected components which are not directly accessible from the boundary rings (Boissonnat and Oudot, 2006). The protection volume will guarantee the presence in M' of the vertices in E_0 which are inside $\mathcal{P}$, and in particular of the vertices connected to the completion boundaries.
- The *protection volume* $\mathcal{P}$. As explained in Section 2.2, we use protection volumes which consist of thin *mesh cushions* $\mathcal{P}_r(M)$ (see Fig. 3), whose radius is slightly greater than the targeted edge length for the hole patches, so that the boundary vertices are fully contained and can be easily matched in the final tailoring.

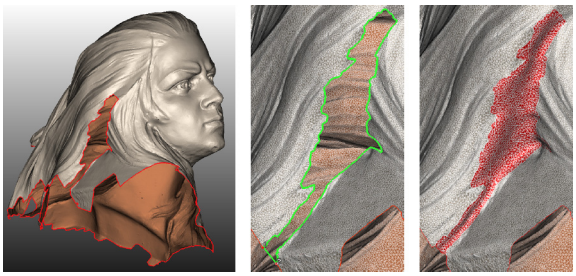


Fig. 7. Tailoring a partial hole. (For interpretation of the references to color in this figure, the reader is referred to the web version of this article.)

The *completion mesh* M' is defined as the result of this tessellation process. In practice, M' is refined only in the holed zones, while the other parts remain coarse (see Fig. 6b).

3.4. Tailoring the hole patches

After the completion mesh M' is generated, the hole patches must be extracted from M' and inserted in M to generate the final output (see Fig. 6c). This procedure is made simple by the fact that the n-ring of each boundary curve was inserted in the tessellation and protected by $\mathcal{P}_r(M)$, so that a tailoring procedure can rely on a matching between the boundary vertices of M and vertices of M'. Note that this matching may not be exact, since spurious vertices of the boundary of M may not be present in M' and there might be small differences in the boundary tessellation. In order to merge M and M' we apply the following steps:

- *Cut* M': the vertices of the interested boundary curves of M and their 1-ring are found and matched in M'. The resulting vertex set (dilated by inserting the neighbor vertices of the matched 1-ring) defines a *cut* of M', hereinafter denoted with C;
- *Select seed vertices of* M': for each interested boundary curve of M, we select a set of vertices V of M' such that every $v \in V$ is inside the interested hole patch. This can be achieved in many different ways, for example by exploring the neighborhood of the boundary curves and picking vertices of M' which are distant from M;
- *Expand to connected-components*: the seed vertices V are expanded to their connected component which takes into account the cut vertices C. We remark that at this point we are assuming that the boundary loops are defining a meaningful boundary for the holed zones (i.e. the mesh must be fixed (Attene et al., 2013) in an n-ring of those boundaries). If this is not the case, then C might not define a *cut* for M' and the result of the current step is the expansion to the entire mesh M'. This degenerate case can be easily identified and the hole filling is aborted. This step can be practically implemented as a simple region growing procedure, where triangles are inserted in a queue based on their adjacency, avoiding triangles that have vertices in the cut C or triangles that lie in the protecting volume (i.e. that intersect M);
- *Insert new faces*: the faces of M' corresponding to the hole patches are selected and inserted in M. Eventually, possible duplicated vertices are joined together and small holes of few triangles (caused by slightly different connections between M and M') are triangulated using Liepa's technique (Liepa, 2003).

There are cases where model parts requiring completion are not initially identifiable by a closed boundary curve. Fig. 7 shows a typical case where a localized filling is needed for a portion of the data defined by an open boundary curve. The model presents a large hole but the user wants to reconstruct only a strip inside the hair region. This situation can be easily handled by our method. In order to determine a valid cut loop on the completion mesh M', the user-defined extremities of an open portion of the boundary curve (in green) are connected by the Dijkstra algorithm on M'. The completion patch is then extracted as explained above, but the bridging portion of the boundary curve is only considered part of the cut (not matched to M).

4. Results and applications

The difficulty of evaluating a mesh completion method resides in the differences among various application requirements. In this section, we discuss the practical usefulness of the proposed method in different cases and provide some comparison with other approaches. The method was implemented in C++ and requires no specialized library except OpenMesh (Botsch et al., 2002) and the code of the Poisson reconstruction method provided by the authors of Kazhdan and Hoppe (2013), which was extended to implement our interpolation technique. In the examples we used a minimum octree depth of 5, maximum depth 8 and a scale factor of 2.2 (Kazhdan and Hoppe, 2013). All tests were executed on a PC with an Intel® Core i7-4790 processor and 16 GB of RAM.

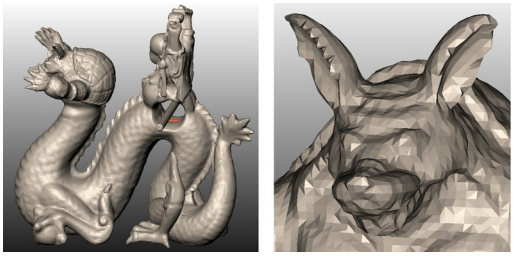


Fig. 8. VolFill (Davis et al., 2002) – requires complete remeshing. (For interpretation of the references to color in this figure, the reader is referred to the web version of this article.)

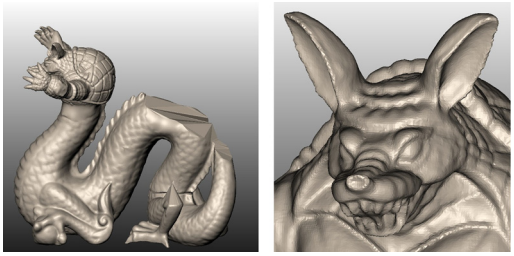


Fig. 9. MeshFix (Attene, 2010) – assumes a single component.

Table 1
Application requirement comparison

Method	Input preservation	Seamless transition	Complex holes	Topology control	Configurable quality	Selective usage
VolFill (Davis et al., 2002)	no	–	yes	yes	no	no
Liepa (Liepa, 2003)	yes	yes	no	no	no	yes
AtomicVol (Podolak and Rusinkiewicz, 2005)	yes	no	yes	yes	no	no
Poisson (Kazhdan et al., 2006)	no	–	yes	yes	no	no
SRPS (Alliez et al., 2014)	no	–	yes	yes	yes	no
MeshFix (Attene, 2010)	limited	yes	limited	no	yes	no
Poisson-Driven	yes	yes	yes	yes	yes	yes

4.1. Comparison with other approaches

The example of Fig. 1 shows a challenging application of the proposed technique to both hole filling and mesh editing (execution time on the given platform 31 s). In particular, different models were assembled by the user by simply approaching the different mesh parts. A mesh completion is then run in order to create a new watertight model. Figs. 8 and 9 are the results obtained using VolFill (Davis et al., 2002) (exec. time $>$ 1 min) and MeshFix (Attene, 2010) (exec. time 47 s). Since the first method is volumetric, it is capable of generating a topologically correct closure. Its major drawback is that it forces a new meshing of the resulting signed distance field with the Marching Cubes method, with the result of ruining important features of the mesh also in zones far from the holes. Furthermore, since the method rely on local morphological operations, large holes remain open (red color in Fig. 8). MeshFix is a lightweight surface-based technique which tends to better preserve the input mesh, but its heuristics can fail if the input comes split into multiple connected components. In this case, the greedy hole closing policy can produce a deletion of parts that become no longer accessible from the biggest component (in Fig. 9 the top part was removed). Table 1 provides a speculative comparison of different approaches from an application perspective by highlighting the salient features considered in this work, i.e. preservation of the input data, quality of the transition between input (if preserved) and completion parts, whether the method handles complex holes (such as islands and gaps), the possibility of introducing user constraints to suggest the topology of the completion, the configurable quality of the resulting hole patches and the possibility of using the technique selectively on portions of the mesh. It emerges that the proposed method presents advantageous solutions and a high flexibility in handling all these application-relevant aspects.

4.2. Model repair and finalization

Fig. 10a shows a high-quality model of a Buddha wood statue where for visibility reasons it was not possible to scan the inner parts of the hands. The hole to be repaired, although topologically simple, is immersed in a context of articulated structures (touching fingers) and has a convoluted edge. The restoration (Figs. 10b and 10c) is fully satisfactory for both its shape and for the absence of self-intersections. Another challenging example is shown in Fig. 10d and 10e, where the Bunny model has been heavily sliced. The proposed method demonstrates the ability to recover the original shape despite the presence of non-trivial gaps and islands. Fig. 11 provides an example of the complete finalization of a high-quality 3D model of a complex artistic object. The input has 4617 holes of various sizes and shapes (Fig. 11a) and the model is finalized in two steps. At first, few partial holes are filled in order to connect a thin missing strip from the basement (Fig. 11b) and occluded parts of the flowers (Fig. 11c). Then the remaining small holes are filled while excluding the biggest boundary loop (Figs. 11e and 11d).

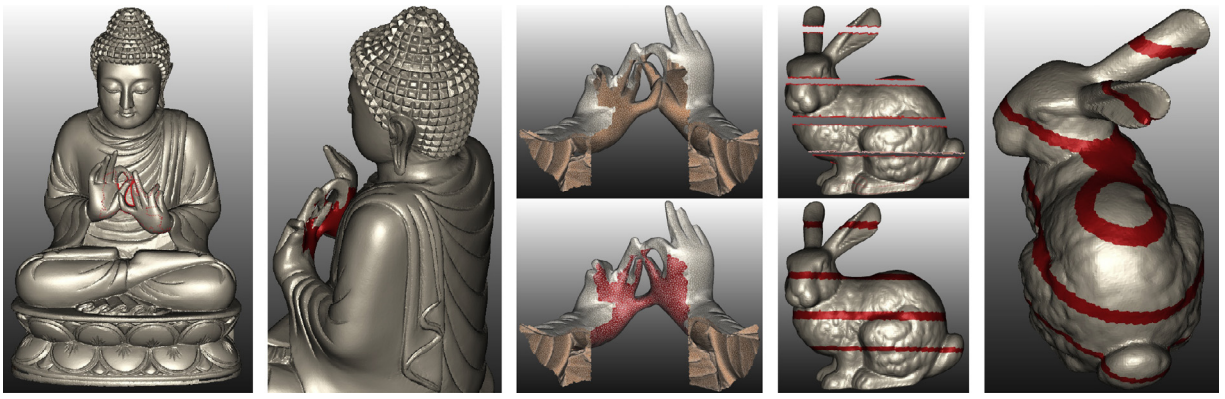


Fig. 10. Examples of filling holes with complex boundary shape (a)–(c) or non-trivial gaps and islands (d)–(e).

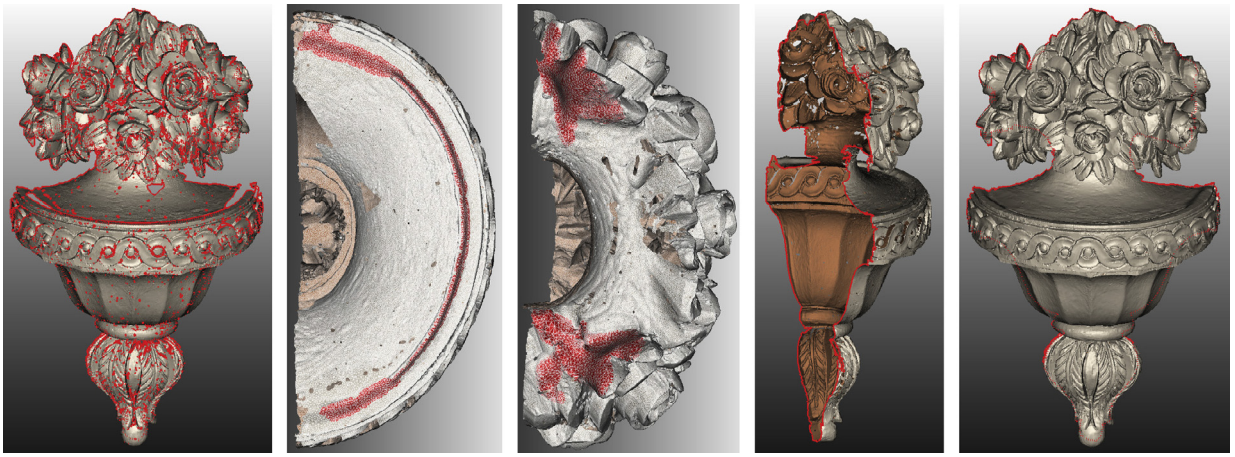


Fig. 11. Example of finalization of a model. The input mesh (a) has 4617 boundary loops. In order to finalize the model, the user has applied two partial fill on the flower basement (b) and in occluded zones (c). After this operation it is possible to exclude a single boundary loop (d) and fill the other holes, leading to a finalized model which preserves every detail of the original mesh.

4.3. Completion-based creative editing

Fig. 12 shows a creative editing application (similar to what happens in Fig. 1) on a real-world model. A reconstruction of a statue is edited by deleting the part of the mesh corresponding to the wings (Fig. 12c). A different model (rose) is then positioned in proximity of the hole (Fig. 12d) and our technique is used for generating a blend surface between the models (notice that the method does not suffer at all from the clear-cut triangle density disparity among the two model parts). It is worth noting that, differently from dedicated solutions addressing the specific task of mesh composition (e.g. Lin et al., 2008), our method automatically gives us the ability to merge parts of the mesh in the very same framework developed for hole filling, guaranteeing at the same time good quality patches seamlessly integrated with respect to the original mesh parts without the need of a full model remeshing.

4.4. Poisson-driven meshing/remeshing

By dropping the requirement of input mesh preservation, the proposed meshing strategy can also be used as a replacement of the meshing algorithm coupled with the Poisson reconstruction method (Kazhdan et al., 2007), since it defines a natural way of generating high-quality feature-preserving adaptive meshes out of a Screened Poisson reconstruction (Kazhdan and Hoppe, 2013). This is consistent to what is underlined in Botsch et al. (2010) about the fact that a Delaunay-triangulation approach can be considered a valuable remeshing strategy. This is preferable with respect to a reconstruction followed by some simplification algorithm since the result is directly based on the underlying solution field. Fig. 13 is a comparison between different meshing strategies. We also consider for comparison the surface reconstruction package of the CGAL library (Alliez et al., 2014) which tackles the same problem by implementing a variant of the Poisson

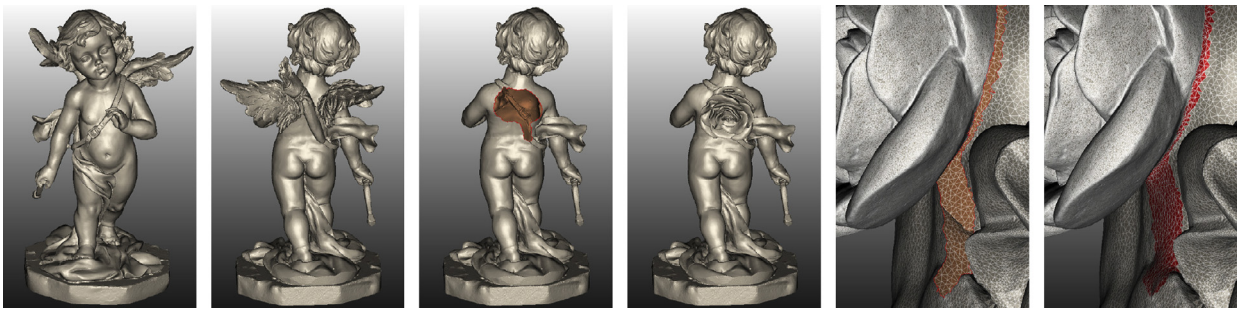


Fig. 12. Example of blending obtained by the proposed method. The input mesh (a)–(b) is edited by the user who selected and eliminated the triangles corresponding to the wings of the angel statue, thus creating a hole (c). A model of a flower was then positioned near the holed zone, and our algorithm was used to create a correct blending (d). Observe that the models have different triangle density (e) and that the hole patch creates a natural blend, while reconstructing the missing parts (f).

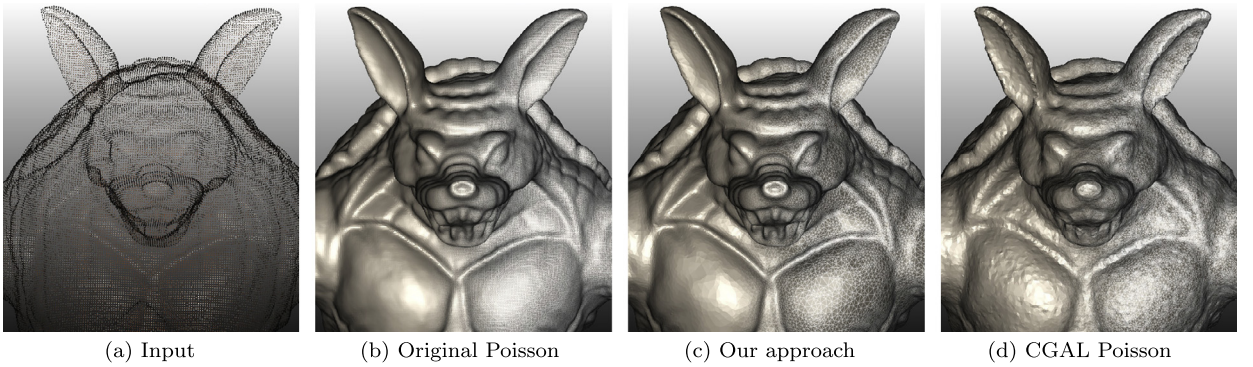


Fig. 13. The proposed meshing strategy can be used in place of the original contouring algorithm (Kazhdan et al., 2007). The reconstruction derived from a point set (a) can be contoured either using the original method (b) or a Delaunay meshing based on our oracle (c). In (d) we provide a direct comparison of the result produced by the surface reconstruction package of the CGAL (Alliez et al., 2014), using the same input and a similar quality criteria.

reconstruction solving a Poisson system on the underlying tetrahedra. However, as evidenced in Fig. 13d, the results produced by a Screened Poisson reconstruction using the same input (Figs. 13b and 13c) are superior in terms of input detail preservation and smoothness of the solution, due to the denser discretization of the domain (given by the solution octree) and the effect of the screening correction. The meshing or – depending on the nature of the input – remeshing solution, naturally made available by our approach, allows to capture all the fine details while using fewer triangles (168k vs 535k of the marching-cube Poisson solution), producing a well-shaped adaptive mesh with denser triangles for higher curvature zones. Execution times were 18.4 s for the original screened Poisson reconstruction, 121.1 s for the CGAL variant (Alliez et al., 2014) and 31.2 s for our method.

4.5. Further remarks

As for computational complexity, the execution time breakdown reported in Table 2 evidences the speed of the method which remains well compatible with the practical needs of the productive contexts for high quality 3D modeling. Despite its proven versatility, robustness and efficiency in dealing with all the even challenging examined cases, the method presents some limitations which are mainly related to how the Poisson reconstruction close the gaps. Namely, if the mesh portions to be joined were too far apart, the Poisson driven closure could not adhere to the desired topology. Another possible issue we already mentioned, is when the Poisson reconstruction is not able to introduce specific geometric features that one may want to synthesize or reproduce from the context knowledge. These two apparently different limitations could actually find a unifying solution within our approach, without requiring any major modifications. In fact, localized and structured point sets can be added to the input of the Poisson reconstruction either as a guiding support to the closure or to introduce contextual knowledge. However, the development and testing of these solutions are out of the scope of this work and are left to future investigations.

5. Conclusion

Motivated by the objective of finding a unified solution to the problems of filling holes and finalizing 3D models, we presented and tested a method that realizes a synergic combination of a reference surface reconstruction technique with

Table 2
Execution times (milliseconds).

Model	Pre-processing	Reconstruction	Meshing	Tailoring	Total
ABDragon – Fig. 1, 1356k faces	548	9845	20014	771	31178
Hurricane – Fig. 7, 427k faces	106	2852	3616	111	6685
Cupido – Fig. 12, 929k faces	462	3614	4725	415	9216
Armadillo – Fig. 13c, 346k faces	0	14314	16874	0	31188
Buddha HD – Fig. 10a, 552k faces	219	3257	2781	213	6470
Sliced Bunny – Fig. 10d, 58k faces	24	3279	3656	70	7029

a high quality mesh generation method. The proposed Poisson-driven approach allows to close complex holes, islands, gaps and missing parts with a seamless integration of the patching triangles along the mesh boundaries, with guaranteed and homogeneous mesh quality. It offers interesting performances for a volumetric method and compares favorably with respect to the state-of-the-art of surface-based completion techniques. The practical usefulness of the method is given by the versatility and effectiveness in handling a large variety of experimental setups and requirements.

Acknowledgements

This work was partially supported by the collaborative research project ABC3D funded by MIUR (Italian Ministry of University and Research) and Regione Lombardia (Italy), PO Competitivitá FESR 2007–2013.

References

Alliez, P., Saboret, L., Guennebaud, G., 2014. Surface reconstruction from point sets. In: CGAL User and Reference Manual, 4.4 edition. CGAL Editorial Board.
Amenta, N., Choi, S., Dey, T.K., Leekha, N., 2000. A simple algorithm for homeomorphic surface reconstruction. In: Proceedings of the Sixteenth Annual Symposium on Computational Geometry. SCG'00, pp. 213–222.
Attene, M., 2010. A lightweight approach to repairing digitized polygon meshes. Vis. Comput. 26 (11), 1393–1406.
Attene, M., Campen, M., Kobbelt, L., 2013. Polygon mesh repairing: an application perspective. ACM Comput. Surv. 45 (2), 15:1–15:33.
Berger, M., Tagliasacchi, A., Seversky, L.M., Alliez, P., Levine, J.A., Sharf, A., Silva, C.T., 2014. State of the art in surface reconstruction from point clouds. In: Eurographics 2014-State of the Art Reports, The Eurographics Association, pp. 161–185.
Bischoff, S., Pavic, D., Kobbelt, L., 2005. Automatic restoration of polygon models. ACM Trans. Graph. 24 (4), 1332–1352.
Boissonnat, J.-D., Oudot, S., 2006. Provably good sampling and meshing of Lipschitz surfaces. In: Proceedings of the Twenty-Second Annual Symposium on Computational Geometry. SCG'06, pp. 337–346.
Botsch, M., Kobbelt, L., Pauly, M., Alliez, P., Lévy, B., 2010. Polygon Mesh Processing. CRC press.
Botsch, M., Steinberg, S., Bischoff, S., Kobbelt, L., 2002. Openmesh-a generic and efficient polygon mesh data structure. In: OpenSG Symp.
Cheng, S.-W., Dey, T.K., Ramos, E.A., 2007. Delaunay refinement for piecewise smooth complexes. In: Proceedings of the Eighteenth Annual ACM–SIAM Symposium on Discrete Algorithms. SODA'07, pp. 1096–1105.
Chew, L.P., 1993. Guaranteed-quality mesh generation for curved surfaces. In: Proceedings of the Ninth Annual Symposium on Computational Geometry. SCG'93, pp. 274–280.
Cohen, E., Tekumalla, L.S., Cohen, E., 2004. A hole-filling algorithm for triangular meshes. Tech. rep. School of Computing, University of Utah.
Davis, J., Marschner, S., Garr, M., Levoy, M., 2002. Filling holes in complex surfaces using volumetric diffusion. In: 3D Data Processing Visualization and Transmission. Proceedings. First International Symposium on 2002, pp. 428–441.
Delaunay, B., 1934. Sur la sphere vide. Izv. Akad. Nauk SSSR, Otdelenie Matematicheskii i Estestvennyka Nauk 7 (793–800), 1–2.
Duguet, F., Hernandez, C., Drettakis, G., Schmitt, F., 2005. Level of detail continuum for huge geometric data. In: ACM SIGGRAPH 2005 Posters. SIGGRAPH'05.
Edelsbrunner, H., Shah, N.R., 1994. Triangulating topological spaces. Tech. rep., Champaign, IL, USA.
Harary, G., Tal, A., Grinspun, E., 2014. Context-based coherent surface completion. ACM Trans. Graph. 33 (1), 5:1–5:12.
Hornus, S., Boissonnat, J.-D., 2008. An efficient implementation of Delaunay triangulations in medium dimensions. Research report RR-6743. INRIA.
Jamin, C., Alliez, P., Yvinec, M., Boissonnat, J.-D., 2014. CGALmesh: a generic framework for Delaunay mesh generation. Research report RR-8256. INRIA.
Ju, T., 2009. Fixing geometric errors on polygonal models: a survey. J. Comput. Sci. Technol. 24 (1), 19–29.
Kazhdan, M., 2005. Reconstruction of solid models from oriented point sets. In: Proceedings of the Third Eurographics Symposium on Geometry Processing. SGP'05.
Kazhdan, M., Bolitho, M., Hoppe, H., 2006. Poisson surface reconstruction. In: Proceedings of the fourth Eurographics symposium on Geometry processing. SGP'06, pp. 61–70.
Kazhdan, M., Hoppe, H., 2013. Screened Poisson surface reconstruction. ACM Trans. Graph. 32 (3), 29:1–29:13.
Kazhdan, M., Klein, A., Dalal, K., Hoppe, H., 2007. Unconstrained isosurface extraction on arbitrary octrees. In: Proceedings of the Fifth Eurographics Symposium on Geometry Processing. SGP'07, pp. 125–133.
Kraevoy, V., Sheffer, A., 2005. Template-based mesh completion. In: Proceedings of the Third Eurographics Symposium on Geometry Processing. SGP'05.
Lewiner, T., Mello, V., Peixoto, A., Pesco, S., Lopes, H., 2010. Fast generation of pointerless octree duals. In: Symposium on Geometry Processing 2010. In: Comput. Graph. Forum, vol. 29, pp. 1661–1669.
Liepa, P., 2003. Filling holes in meshes. In: Proceedings of the 2003 Eurographics/ACM SIGGRAPH Symposium on Geometry Processing. SGP'03, pp. 200–205.
Lin, J., Jin, X., Wang, C.C., Hui, K.-C., 2008. Mesh composition on models with arbitrary boundary topology. IEEE Trans. Vis. Comput. Graph. 14 (3), 653–665.
Müller, M., 2009. Fast and robust tracking of fluid surfaces. In: Proceedings of the 2009 ACM SIGGRAPH/Eurographics Symposium on Computer Animation. SCA'09, pp. 237–245.
Pavić, D., Campen, M., Kobbelt, L., 2010. Hybrid booleans. Comput. Graph. Forum 29 (1), 75–87.
Podolak, J., Rusinkiewicz, S., 2005. Atomic volumes for mesh completion. In: Proceedings of the Third Eurographics Symposium on Geometry Processing. SGP'05.
Sharf, A., Alexa, M., Cohen-Or, D., 2004. Context-based surface completion. ACM Trans. Graph. 23 (3), 878–887.
Süßmuth, J., Meyer, Q., Greiner, G., 2010. Surface reconstruction based on hierarchical floating radial basis functions. Comput. Graph. Forum 29 (6), 1854–1864.
Varnuška, M., Parus, J., Kolingerová, I., 2005. Simple holes triangulation in surface reconstruction. In: Proceedings of Algoritmy, pp. 280–289.

Wang, X., Liu, X., Lu, L., Li, B., Cao, J., Yin, B., Shi, X., 2012. Automatic hole-filling of cad models with feature-preserving. Comput. Graph. 36 (2), 101–110.
Watson, D.F., 1981. Computing the n-dimensional Delaunay tessellation with application to Voronoi polytopes. Comput. J. 24 (2), 167–172.
Wojtan, C., Thürey, N., Gross, M., Turk, G., 2009. Deforming meshes that split and merge. ACM Trans. Graph. 28 (3), 76:1–76:10.
Wu, X.J., Wang, M.Y., Han, B., 2008. An automatic hole-filling algorithm for polygon meshes. Comput-Aided Des. Appl. 5 (6), 889–899.
Xie, W.-C., Zou, X.-F., 2013. A triangulation-based hole patching method using differential evolution. Comput. Aided Des. 45 (12), 1651–1664.
Zhao, W., Gao, S., Lin, H., 2007. A robust hole-filling algorithm for triangular mesh. Vis. Comput. 23 (12), 987–997.