Polygon Mesh Repairing: An Application Perspective

MARCO ATTENE
IMATI-GE Consiglio Nazionale delle Ricerche
and
MARCEL CAMPEN and LEIF KOBBELT
RWTH Aachen University

Nowadays, digital 3D models are in widespread and ubiquitous use, and each specific application dealing with 3D geometry has its own quality requirements that restrict the class of acceptable and supported models. This article analyzes typical defects that make a 3D model unsuitable for key application contexts, and surveys existing algorithms that process, repair, and improve its structure, geometry, and topology to make it appropriate to case-by-case requirements.

The analysis is focused on polygon meshes, which constitute by far the most common 3D object representation. In particular, this article provides a structured overview of mesh repairing techniques from the point of view of the application context. Different types of mesh defects are classified according to the upstream application that produced the mesh, whereas mesh quality requirements are grouped by representative sets of downstream applications where the mesh is to be used. The numerous mesh repair methods that have been proposed during the last two decades are analyzed and classified in terms of their capabilities, properties, and guarantees. Based on these classifications, guidelines can be derived to support the identification of repairing algorithms best-suited to bridge the compatibility gap between the quality provided by the upstream process and the quality required by the downstream applications in a given geometry processing scenario.

Categories and Subject Descriptors: I.3.5 [**Computer Graphics**]: Computational Geometry and Object Modeling

General Terms: Boundary representation, Geometric algorithms, languages, and systems

Additional Key Words and Phrases: Fixing, Topology, Geometry

1. INTRODUCTION

Digital 3D models are key components in a vast number of industrial and scientific sectors, such as product design and manufacturing, gaming and simulation, cultural heritage and archaeology, medicine, bioinformatics and pharmaceutical sciences. In most cases, visualization is just one of the many steps constituting the lifecycle of a digital 3D model. 3D geometry, indeed, often needs to be analyzed and processed through advanced algorithms that typically have strict requirements on the quality and integrity of their input. In practice, these requirements are often not met by models originating from various sources. Thus, adapting imperfect 3D models to such requirements is a task of high importance.

Although many representations have been proposed for 3D models, polygon and triangle meshes are a de facto standard in most domains. Besides being extremely flexible and expressive, polygon meshes have been directly supported by accelerated graphics hardware for several years, and this contributed to their diffusion and establishment.

The vast majority of today's 3D mesh models originate from one of two common data sources: from *digitization* of real-world objects or phenomena, or from *tessellation* of virtual, synthetic data typically produced in a computer. Examples of the first case are surface scanning [Bernardini and Rushmeier 2002], shape from shading [Zhang et al. 1999], 3D photography [Curless and Seitz 2000; Seitz et al. 2006], medical imagery [Zhang et al. 2002], whereas synthetic 3D data often originates from implicit mathematical formulations [Turk and O'Brien 2002], CAD systems [Farouki 1999], or sketch-based modelling [Igarashi et al. 1999]. These two sources of 3D data correspond to different modeling pipelines; both of them lead to surface meshes with particular characteristics which are not always suitable for all possible downstream applications. Thus, in these cases some form of *mesh repairing* is required to adapt the raw models to the requirements of the applications at hand.

Digitized models Although 3D digitization tools are becoming more and more flexible, each specific downstream application has its own requirements that restrict the class of supported 3D models. In industrial design, for example, several processes assume that the mesh does not contain degenerate, or nearly degenerate, elements. In computer graphics, numerous shape analysis tools expect the input mesh to enclose a well-defined solid. Such tools typically fail if the mesh has holes, or produce unpredictable results if the input has self-intersections. In most cases, digitizing a real-world 3D object amounts to capturing several views of the object (i.e. the range images) which are eventually aligned and merged into a single model [Bernardini and Rushmeier 2002]. While this is sufficient for mere visualization purposes, at this stage polygon meshes may contain degenerate elements, overlapping or self-intersecting parts, surface holes, and a number of other *flaws* that make them unsuitable for a wide spectrum of applications.

Synthetic models The currently established pipeline for computer-aided product modeling involves several steps. Most often, a creative designer draws a sketch on paper, a CAD professional translates the sketch to a patch-based boundary representation (B-rep) in a CAD system and, finally, tessellation algorithms generate triangle meshes for downstream applications such as structural simulation or rapid prototype printing, to name a few. Normally, downstream applications assume that their input meshes are closed and consistent manifolds. Thus, the tessellation step should take this requirement into account and produce proper meshes. Unfortunately this is not the case in many tools. Tessellation algorithms typically create a separate mesh per patch and, though each such mesh might respect all the requirements, neighboring patches are often not continuously and consistently connected and, in some cases, they overlap or intersect. This leads to several artifacts which typically require a manual and tedious post-processing to fix the model. This motivates the significant effort that has been spent so far on developing algorithms that are able to (semi-)automatically repair mesh models.

1.1 Overview and Motivation

The objective of this report is to provide a comprehensive reference to mesh repair techniques from a practical application perspective. Thus, we first categorize upstream applications based on the typical characteristics/defects of the meshes they produce. Then we provide a classification of downstream applications based on the requirements they typically impose on their input meshes. Finally we categorize possible defects and existing repair algorithms along with their own specific requirements on the input and the characteristics of the output they produce. By looking at the combinatorics of upstream application (data source), repair method, and downstream application based on these criteria, the reader can choose which repair methods are well suited for the data-link between an upstream-downstream pair (as they are established in various geometry processing pipelines).

Repairing algorithms are categorized based on several characteristics including the approach employed (e.g. local corrections or global remeshing), the quality requirements on their input, and the guarantees of their output. Furthermore, we distinguish between algorithms that fix local connectivity flaws, global topology issues, geometric errors, or a combination of the above. With the exception of algorithms that fix geometric errors, for which a methodological perspective is given by Ju [2009], treatment of the other defect categories is mostly unsurveyed in current literature. Thus, this report collects algorithms that treat either combinatorial, topological, or geometrical issues and puts them under the common category of *mesh repairing techniques* while considering practical problems such as the definition of repairing *workflows* that vertically integrate multiple algorithms for the needs of a specific application. Furthermore, the problem of mesh repairing is described from an **application perspective**, and hence considers the *context* where repairing is necessary by focusing on the aforementioned data-link problem between upstream and downstream applications.

2. DEFINITIONS

Since the facets of polygon meshes can typically be triangulated, in the remainder we focus on meshes with triangular facets. Undesirable characteristics of a triangle mesh can be roughly classified as *topological* or *geometrical* defects. For this reason, in order to better describe the problem, we aim to maintain a clear separation between topology and geometry. We adopt the notation of Attene and Falcidieno [2006] and denote a triangle mesh as a pair $M = (P, \Sigma)$, where P is a set of N vertex positions $\mathbf{p}_i = (x_i, y_i, z_i) \in \mathbb{R}^3$ with $1 \leq i \leq N$, and Σ is an abstract simplicial complex over these vertices which contains all the topological information.

We say that M is *combinatorially* manifold iff Σ is a combinatorial manifold [De Floriani et al. 2003]. In its turn, Σ is a combinatorial manifold iff all its vertices are manifold, and a vertex of Σ is manifold if its neighborhood is homeomorphic to a disk in the topology of Σ.

The *geometric realization* of a simplex σ is the convex hull of the point positions of its vertices and the union of all the geometric realizations of the simplices of M is the geometric realization of M, denoted by $|M|$. $|M|$ is a set of points in $\mathbb{R}^3$ for which a Euclidean topology exists, and we say that M is *geometrically* manifold iff the neighborhood of each point in $|M|$ is homeomorphic to a disk in this topology. Note that a triangle mesh may be manifold in the combinatorial sense and not in the Euclidean one, e.g. when the mesh self-intersects. Also, a geometrically manifold mesh may be not combinatorially manifold, e.g. when there is a topologically singular edge with three incident faces, but two of them coincide geometrically. A triangle mesh is orientable iff all its triangles can be oriented consistently.

Orientable, manifold meshes enclose well-defined solids by separating the embedding space into interior and exterior volumes. In this sense they represent "real" objects.

2.1 Problem Statement and Repairing Guidelines

In this report we loosely define a mesh repairing algorithm to be a process that takes as input a surface mesh M and produces as output a modified version M' of the input where some specific defects or flaws (as categorized and described in Section 3) are removed or alleviated. As already mentioned, the type of algorithm(s) suitable and the type of defect(s) that need to be fixed depend both on the upstream and on downstream applications in a given scenario. Furthermore, it is important to notice that some repairing algorithms have their own requirements on the input. For example, most hole filling algorithms assume that the boundary of each hole is a connected 1-manifold. Also, while fixing specific defects, some repairing algorithms may newly introduce other flaws. Thus, in order to select a suitable algorithm (or a combination of algorithms) for a specific case, it is useful to investigate the context as follows:

(1) What is the upstream application ?
$\rightarrow$ Determines characteristics and defects of M

(2) What is the downstream application?
$\rightarrow$ Determines requirements on M'

(3) Based on this information:
$\rightarrow$ Is it necessary to repair M?

(4) If repairing is necessary:
$\rightarrow$ Is there an algorithm that does it directly?

(5) If direct repair is not possible:
$\rightarrow$ Can several algorithms be used in sequence?

(6) If not:
$\rightarrow$ There is a gap in the state of the art.

As a support for step 1, Section 4.1 provides hints and tables to identify the potential issues related to specific upstream applications. The same kind of support is given for downstream applications (step 2) in Section 4.2. The combined information reported in the tables gives an idea of the kind of repairing that is possibly necessary (step 3) for a specific case. Then, in Section 5 existing algorithms are surveyed and, for each of them, input requirements, defects treated, and output characteristics are outlined. Finally, summary tables report this information to support the analysis of steps 4 and 5.

In Section 5 and in the summary tables, further attributes of the repairing algorithms are discussed (e.g. guaranteed success vs. best effort, automatic vs. interactive, newly introduced flaws, ...), since these are important for the decision process. For instance, if there is the need to have a guaranteed success, one may opt for advanced algorithms that provide such guarantees, whereas if speed is more important than strict quality or success guarantees, one may want to choose simpler heuristics-based algorithms.

In Section 6 we discuss the gaps in the available range of repairing methods that could be identified and show up possible avenues for future work that could provide further valuable contributions in the field.

3. DEFECTS AND FLAWS

Most graphic formats widely used to share 3D models (OFF, VRML, PLY, ...) encode surface meshes through *indexed face sets*; specifically, these file formats contain a first block specifying the positions of the vertices, and a second block where each polygon is represented through a sequence of indices of vertices in the first block. Clearly, files of this type are not guaranteed to represent a valid simplicial complex, as they may easily encode non-manifold and/or non-orientable sets of polygons, isolated elements, and a number of other flaws that are often the source of problems in several application contexts. In the following sections we provide a categorization of the main issues that may need particular treatment. Specifically, we distinguish among issues about local connectivity (Section 3.1), global topology (Section 3.2) and geometry (Section 3.3). Figure 1 illustrates these various flaws and defects.

3.1 Local Connectivity

The first family of problems that most often needs to be treated regards the mesh connectivity. Namely, it might happen that the set of (triangulated) polygons encoded in a file does not constitute a combinatorially manifold simplicial complex. There are various cases of such issues which we list in the following sections. Note that we consider this family of problems to be established by pure mesh connectivity, i.e. it includes issues of the abstract simplicial complex only. Holes and gaps in the mesh do as well manifest themselves in the connectivity (e.g. edges with only one incident face), but almost ever also in the geometric realization. Since the repair of such issues heavily relies on the examination of the available input geometry (and most often involves conceiving new geometry), we consider them geometric issues, which are handled in Section 3.3.

3.1.1 *Isolated Vertices.* A vertex which is not a face of any other simplex of the mesh is said to be isolated. Isolated vertices can often simply be ignored but, if necessary, their removal is very simple and many existing tools provide manual or automatic procedures to do that.

3.1.2 *Dangling Edges.* The specifications of some file formats allow the explicit encoding of mesh edges in addition to vertices and facets. Examples of such formats are OFF and PLY. This feature is most often unused but, when it is, it might encode non-regular surfaces with so-called *dangling*, or *naked* edges; in other words, the file format might represent edges with no incident triangles. A common strategy is to simply ignore or remove these edges but they might also be exploited by some approaches as a useful source of information about the intended underlying geometry to be repaired. Due to their trivial treatment, isolated vertices and dangling edges are not considered any further in the remainder of this report.

3.1.3 *Singular Edges.* When more than two polygons share a common edge, then such an edge is said to be *singular*, *complex*, or *non-manifold*. This situation is

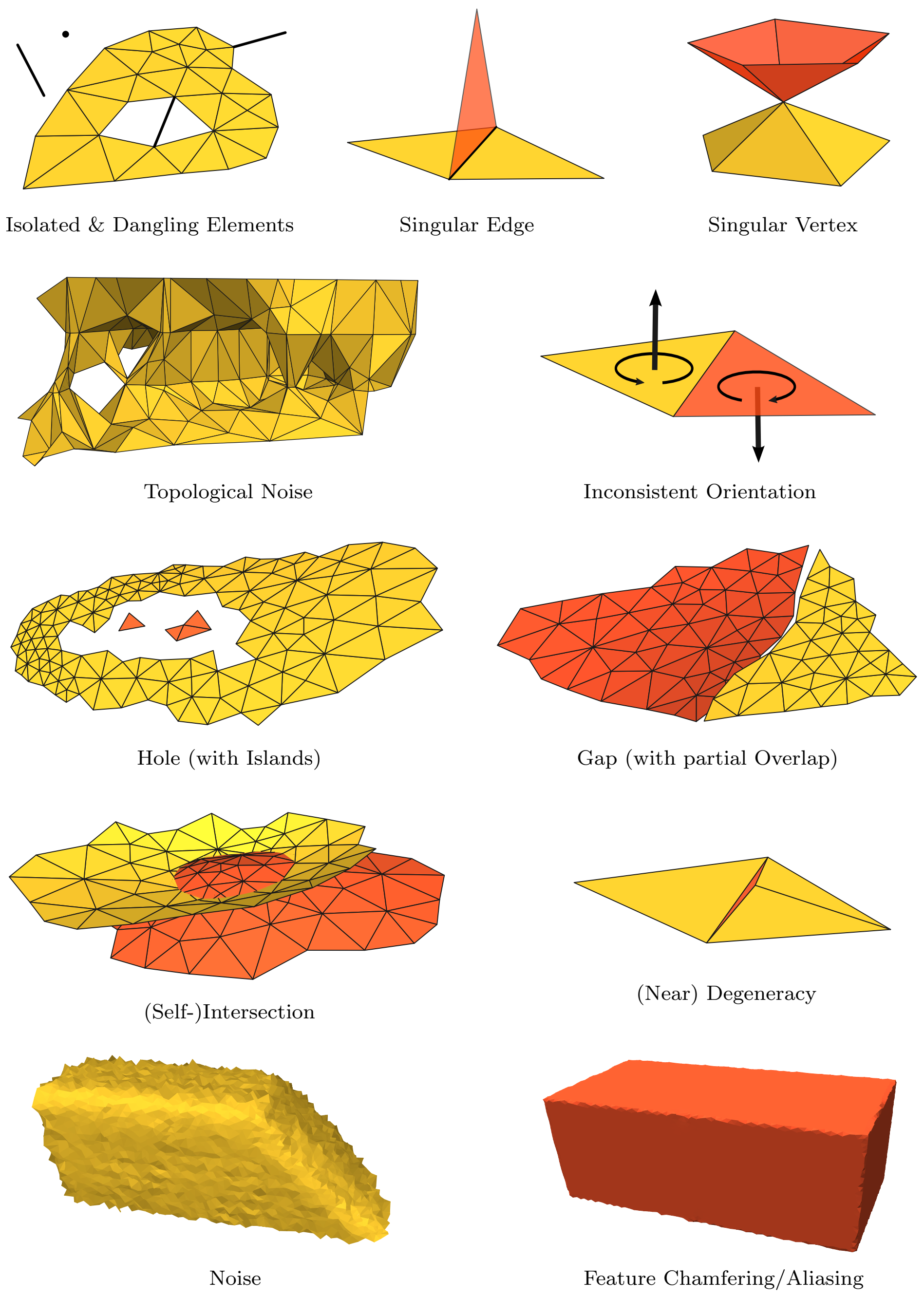

Fig. 1. Illustration of the various types of flaws and defects that can occur in polygon meshes.

not unusual, especially when the mesh comes from tessellation of a multi-patch CAD model. Clearly, even a mesh having just one singular edge is not a combinatorial manifold; indeed, it is easy to prove that the endpoints of a singular edge cannot be manifold vertices. Since the condition of combinatorial manifoldness is required in several application contexts, there is a strong call for solutions that convert such a mesh to a combinatorial manifold. Although this problem is not as easy to solve as the case of unreferenced vertices or edges – due to the inherent ambiguity – some solutions have been proposed in the literature and they are discussed in Section 5.1.1.

3.1.4 *Singular Vertices.* When a vertex is not manifold in the topology of the abstract simplicial complex, it is called a combinatorially *singular* vertex. The detection of such vertices is slightly more complicated than the detection of singular edges. Indeed, while for edges it is sufficient to count the number of incident triangles, for vertices it is necessary to count the number of connected components in the neighborhood. By definition, a mesh with singular vertices cannot be a combinatorial manifold. Solutions to this kind of problems are typically based on the duplication of the singular vertex followed by a re-assignment of each component of the neighborhood to one of the copies (cf. Section 5.1.1).

3.2 Global Topology

Flaws in the global topology of the object at hand are another source of potential problems when a mesh is processed. Herein, with *global topology* we denote the overall topological characteristics of the surface, thus including the number of connected components, the genus, the number of cavities and the orientability.

3.2.1 *Topological Noise.* The term *topological noise* was introduced by Guskov and Wood [2001] to denote a common problem which arises when reconstructing a surface starting from point clouds or when extracting isosurfaces from 3D images. Often, in these processes tiny handles or tunnels, which were not present in the original object, are introduced in the constructed digital model due to aliasing effects or noise in the discrete underlying data. Hence, while a given real-world object has a given topological genus, its digital counterpart may have a different genus (usually much higher). For example, the famous Stanford Buddha statue (`http://graphics.stanford.edu/data/3Dscanrep/`) actually has exactly six topological handles, while the digital model produced by the team at Stanford happens to have 104 such handles. The difference between the two has been conceptualized with the term *topological noise.* Clearly, this topological noise complicates subsequent operations such as remeshing, parameterization, or smoothing. Section 5.1.9 outlines some solutions which have been proposed to eliminate this problem.

3.2.2 *Orientation.* As previously mentioned, polygons in an indexed face set are represented through sequences of vertex indices. Typically, such indices are not randomly listed, but their order indicates the orientation of the polygon. In most rendering systems, the orientation is associated to the visibility of a face. In other words, a polygon is visible only when its bounding vertices are ordered counterclockwise (or clockwise) with respect to the observer's point of view. Some formats, such as VRML, allow the user to specify whether visibility must be granted by a clockwise or counterclockwise ordering, but some others do not, and simply assume a given ordering (most often counterclockwise). To grant full visibility in all rendering systems, and to provide a coherent normal orientation for correct surface analysis, it is often necessary to provide a unique consistent orientation to all the polygons of a mesh. This is typically achieved by selecting a *seed* face and by propagating the orientation to neighboring faces. Nevertheless, some configurations are intrinsically not orientable, which means that to have a consistent orientation the system must necessarily cut the surface [Attene and Falcidieno 2006].

3.3 Geometric Issues

This family of problems regards not only the abstract simplicial complex, but also its geometric realization. In other words, the issues discussed in this section are characterized by the position of the vertices. In contrast to the combinatorial defects discussed so far, this family of problems is further complicated by the need to treat continuous coordinates and intersections that (for efficiency reasons) are required to be expressed and processed with finite precision arithmetics, which typically leads to robustness issues.

3.3.1 *Surface Holes and Gaps.* When digitizing a real-world object through standard laser range scanners, it is usual to encouter occluded parts which cannot be captured because the laser beam is shadowed by other parts of the object. Also, when designing a surface through standard CAD systems, the various tessellated patches are typically slightly displaced in a way that – though the intention of the designer was to construct a continuous surface – adjacent patches are separated by undesired gaps. In other words, there are different scenarios where some of the data cannot be produced directly, and thus must be compensated for in a second step. Such steps are known as *gap closing* and *hole filling.* Typically, *gaps* and *holes* are defined differently and are treated with different strategies. Most often (although done less strictly in early literature) a gap is defined as the empty region *between* two triangulated surface patches that should be continuously connected but are not due to the gap. In contrast, a hole is an undesirably missing piece of surface *within* a triangulated patch. A main difference between the two cases is given by the connectivity of their boundaries: the boundary of a gap, indeed, is typically made of two (or more) disconnected chains of edges; in contrast, the boundary of a hole normally consists of one or more closed edge loops. Opposed to gaps, which often are quite narrow, holes might represent larger areas of missing data, such that their repair poses the further challenge of conceiving a plausible geometry to fill the holes.

Since they are extremely important repairing tasks, gap and hole filling have been studied deeply, and numerous algorithms exist, ranging from simple boundary matching and hole triangulation up to more elaborate techniques which provide some desirable properties, e.g. continuity of the normal field (cf. Sections 5.1.2 and 5.1.3). In their broadest sense, the terms *gap* and *hole* may also denote more specific problems arising in CAD applications such as *cracks* and the so-called *T-junctions.* Cracks are small, elongated portions of missing surface [Nielson et al. 1999] which typically originate from a change of resolution in the tessellation algorithm across adjacent patches. Most cracks are just surface holes and, as such, they can often be fixed by simple triangulation – possibly requiring further considerations due to the near-degenerate triangles that are often generated in this process. Some others, however, are more complex gaps bounded by several curves that make their connection ambiguous, and thus require specific matching procedures that can even require human interaction [Barequet et al. 1998; Attene and Falcidieno 2006]. T-junctions are particular cracks with null area: they are combinatorial holes or gaps where some parts of the boundary coincide geometrically but are not combinatorially consistent. These configurations are often treated by inserting new vertices and edges in order to make the boundaries compatible, but other solutions exist, including the triangulation with zero-area triangles that are eventually fixed in a further step (cf. Section 5.1.5).

When holes are due to occluded parts that could not be scanned, they might be more complex in that they contain so-called *islands*, which are small disconnected pieces of surface that the scanner could capture within the occluded region. The boundary of holes with islands is not connected, which complicates their repair and disqualifies many of the proposed hole filling methods. Hole filling methods that investigate the input globally and consider the mutual relation of multiple boundaries to handle such complex cases most often imply a complete conversion and remeshing of the input (cf. Section 5.2).

Since, due to the involved ambiguities, the task of repairing meshes with holes and gaps is inherently ill-posed, the user might need the freedom of interactively selecting boundaries of holes to be filled or specific pairs of patches to be merged (cf. Section 6.2). When holes are particularly large, it might be useful to reproduce some known pattern from within the patch or from some shape repository to obtain more plausible results than by smooth patches, that are often used for smaller holes. Approaches of this kind are known as *(example-based) mesh completion* (cf. Section 5.1.4).

3.3.2 *Degenerate Elements.* Degenerate triangles are triangles with zero area. Clearly, these elements are the source of several problems for numerous applications, since many useful entities cannot be computed on such triangles (normal vectors, circumscribing circles, barycentric coordinates, ...). Even though many software packages include pre-processing functions to adapt the input mesh to their needs, these functions are often not able to deal with degenerate triangles. Several applications which are strongly based on the compuation of the aforementioned entities (e.g. finite element analysis or Delaunay refinement [Shewchuk 2002]) fail when the mesh contains degenerate triangles or become unstable when it contains nearly degenerate triangles. Some approaches have been introduced to fix this kind of problem. They are discussed in Section 5.1.5.

Apart from these degeneracies and near-degeneracies, the general quality of triangles is an important characteristic for some applications that have pronounced requirements on the triangle aspect ratios, the interior angles, or the uniformity and density of the vertex distribution over the surface. Converting a generic mesh into one meeting such sorts of *continuous quality criteria* is the scope of *surface remeshing*. Though in some cases this process of mesh quality enhancement may be seen as a particular case of repairing, a comprehensive treatment of remeshing, mesh refinement, and mesh simplification methods would lead us too far from the scope of this report. Therefore, we point the interested reader to specific surveys in these fields [Alliez et al. 2008], [Luebke 2001].

3.3.3 *Self-Intersections.* In several application contexts the input mesh is assumed to represent the boundary of some solid volume, and thus it is required not to have self-intersections. While it is relatively easy to check a mesh for self-intersections, resolving them is a challenging problem due to the inherent ambiguities. Self-intersecting meshes are typically generated by tessellation of multi-patch CAD models, by deformation of mesh models, by composing models out of multiple parts without care, or when merging patches reconstructed from partial scans of a 3D object. Due to the ambiguities, there is no common strategy to tackle this problem, but some approaches, which are often tailored to specific scenarios, exist (cf. Section 5.1.6).

3.3.4 *Sharp Feature Chamfering.* Most acquisition techniques, as well as several remeshing and contouring algorithms, restrict each sample or vertex to lie on a specific line or curve whose position is completely defined by a pre-established pattern. In most cases, such a pattern cannot be adjusted to coincide with sharp edges and corners of the model and, consequently, almost none of the samples lie on such sharp features. This leads to aliasing artifacts, i.e. the sharp edges and corners of the original shape are removed by the sampling process and replaced by irregularly triangulated chamfers, which often result in a poor-quality visualization and high L^∞ distortion. Some methods exist that use the available information to deduce the original geometry of the chamfered features and eventually reconstruct them (cf. Section 5.1.7). Having such well-defined sharp features has clear advantages for both visualization and reverse engineering.

3.3.5 *Data Noise.* Every digitization tool has a finite precision. Thus, the acquired raw data of the sampled model contains additive noise from various sources. A main challenge is to remove the noise while preserving the main morphology of the underlying sampled surface, with particular care to high-frequency details like corners, edges, or other sharp features. Noise reduction can be applied either before or after generating the mesh from the raw data. The advantage of denoising a mesh rather than a point cloud is given by the fact that the connectivity information implicitly defines the surface topology and provides fast access to neighboring samples. This has been exploited to devise well-known noise-reduction algorithms for meshes such as Laplacian smoothing and bilateral denoising (cf. Section 5.1.8).

4. APPLICATION-ORIENTED MESH REPAIRING

This section provides a reference to be used as a support when choosing mesh repairing algorithms to link upstream to downstream applications in a given scenario.

4.1 Upstream Applications

In the context of repairing, upstream applications (or *sources*) can be characterized based both on the *nature* of the data modeled (i.e. (physical) real-world data vs. (virtual) concepts) and on the *approach* employed to convert such data into polygon meshes. Both, nature and conversion approach, can be the origin of defects in a mesh. In essence, to identify all the potential defects of a mesh based

Table I. Typical (X) and sporadic (x) defects that might be intrinsic in the concept or data used to create the model.

Nature	noise	holes	gaps	intersections	degeneracies	singularities	topological noise	chamfered features
Digitized (physical)	X	X					X	X
Designed (virtual)			X	X	x	X		

Table II. Typical (X) and sporadic (x) defects originating from specific mesh construction approaches.

Approach	noise	holes	gaps	intersections	degeneracies	singularities	topological noise	chamfered features
Tessellation			X	X	x			
Depth image fusion				X	x	x		
Raster data contouring						X		
Implicit function contouring					X		x	X
Reconstruction from points		x	x				x	x
Height field triangulation								
Solid model boundary extraction						X		

on the upstream application that produced it, it is often sufficient to identify the nature as well as the approach employed. In the remainder we determine the specific properties of the data nature and common mesh construction approaches. The results are summarized in Tables I and II.

Nature A three-dimensional model can originate either from a virtual design process or from digitization of real-world data. If a model is *designed*, the basic concept is typically an abstraction, and downstream applications may face problems such as non-manifoldness, gaps, and intersections. These defects are either caused by inaccuracies in modeling or produced by description processes that are often based on surface representations although solids are meant to be created. Opposed to that, if the model is *digitized*, problems are mostly in the measured data and may include noise, holes, chamfered features, and topological noise due to limitations of the measurement process employed for digitization. These problems are intrinsic in either the abstraction used or the data acquired (cf. Table I).

Approach Then, such abstraction/data is converted into a polygon mesh (if not originally designed in polygonal form), and the conversion itself can be the source of further flaws that depend on the specific approach used. Note that, e.g. for the case of digitized data, we consider topological noise and chamfered features to be already present "in the data" (that in most scenarios lacks precise feature and topology information) since the approach applied to construct a polygon mesh from such digitized data is not responsible for the data-inherent ambiguities that then often cause undesired results during meshing. With that in mind, for the case of *tessellation* of, e.g., a CAD model, gaps and intersections might arise due to the necessarily occurring deviation of each triangulated patch from the original curved surface. Depending on the quality of the tessellation algorithm also (near-)degenerate polygons might be created. In some cases the mesh-based *fusion of range images* is rooted on local operations that might produce meshes with self-intersections. Degeneracies and singularities might also be introduced due to round-off errors and insufficient special case handling. This happens, for example, when the fusion is performed by the *Polygon Editing Tool (PET)* software accompanying the widely diffused Minolta V910 laser scanner. The *contouring of raster data* might produce meshes with singular edges if no disambiguation strategies are used. The *contouring of implicit functions* usually chamfers sharp features if no special measures for adaptation are applied – topological noise can also arise when a fixed sampling pattern is used for contouring, but specialized surface-based methods rather follow the iso-contours to better capture the topology. For the surface *reconstruction from point clouds* various methods are available that do not introduce any defects or flaws (apart from those already inherent in the point cloud), but there are others that might leave holes or gaps and produce meshes with aliasing or topological artifacts even if the point cloud meets certain sampling criteria that would allow for a correct reconstruction. The *triangulation of height fields* is a rather simple process and respective methods usually do not introduce new defects. The *boundary extraction from solid models*, e.g. tetrahedralized fields or point sets, can yield non-manifold meshes with singular edges and vertices in cases of tangential self-contact of the represented object. Table II summarizes these observations.

4.2 Downstream Applications

Giving a precise categorization of downstream applications based on their input requirements would be rather complicated and bulky. Furthermore, such a classification would be inevitably incomplete as particular cases and new applications arise too frequently. For these reasons, herewith we provide an overview of the prototypical requirements of a discrete sampling or grouping of the continuum of applications, while leaving the reader the freedom to filter or integrate this information based on the specificity of the application at hand. We summarize the requirements in Table III. For the purpose of mere *visualization*, only the existence of significant holes is generally deemed unacceptable – all other types of defects can often be neglected. This might be one of the reasons why the importance of the topic "mesh repair" is often underestimated. To achieve pleasing renderings of a certain visual quality, however, also noise, gaps, and chamfered features can be adverse. For *modeling* and deformation tools, connected surfaces without degeneracies are usually required – intersections are often acceptable in the case of surface-based methods. Singularities and topological noise do not cause problems for some methods, others require or prefer clean manifold meshes. For *rapid prototyping* purposes, the mesh model naturally needs to be convertible to a solid model, i.e. it has to well-define an interior and exterior volume. For this purpose the mesh definitely has to be closed and free of intersections and singular non-manifold configurations that would prevent an unambiguous volume classification. For many

Table III. Typical quality requirements of groups of downstream applications. For each group, the table reports the defects whose repairing is most often mandatory (X), optional (x), or unnecessary.

Application group	noise	holes	gaps	intersections	degeneracies	singularities	topological noise	chamfered features
Visualization	x	X	x					x
Modeling		X	X		X	x	x	
Rapid Prototyping		X	X	X		X		
Processing	X	X	X	x	X	X	x	x
Simulation	X	X	X	X	X	X	X	x

geometry processing applications, the input mesh is additionally required to be free of degeneracies and noise in order to allow for the computation of element properties and discrete differential quantities in a reasonable way. Aliasing effects like topological noise and chamfered features negatively affect and disturb several of these methods. In most cases, the *simulation* of real-world phenomena on digital models poses the highest requirements on the model's quality in order to be able to achieve reliable results.

5. STATE OF THE ART

In this section we describe, analyze, and classify a reasonably complete set of methods that have been proposed for mesh repairing tasks during the last two decades. On the highest level we distinguish between methods that use a local approach (Section 5.1) and methods that employ a global strategy (Section 5.2). Specifically, we say that an algorithm uses a *local* approach if it modifies the mesh only in the vicinity of the individual defects and flaws, whereas the remaining parts of the surface are kept completely unaltered. Conversely, *global* methods are typically based on a complete remeshing of the input (implied by the use of some intermediate data structure different from a polygon mesh): this allows to more easily achieve robustness and the global investigation can help to resolve the ambiguities that emerge during the repairing process in many cases in a more reasonable way – at the cost of unnecessarily impairing the accuracy in flawless regions.

As a rule of thumb, highly-detailed, feature-rich meshes with mainly isolated flaws should be fixed using local approaches to preserve as many details as possible. Conversely, highly corrupted or inconsistent meshes with multiple types of defects (e.g. polygon soups) would better be fixed through one of the global approaches described in Section 5.2 – especially if one needs a guarantee that the repair process succeeds. Global approaches typically are highly robust whereas local approaches are less invasive.

In the remainder of this section, algorithms targeting the same set of flaws are listed within tables where, for each method, the requirements of the input mesh are reported along with other useful information such as the request for input parameters, guarantees of success, accuracy of the results, and possible defects newly introduced by the repairing process itself.

It is worth mentioning that the table contents are not meant to be exhaustive and exact; they just allow to quickly identify possible algorithms, or sequences of algorithms, for a specific repairing task with its particular requirements. For example, the objective of the "parameters" column is to indicate whether the algorithm can be executed in a workflow without the need of a case-by-case user intervention. Hence it indicates only parameters whose setting is normally necessary because the user cannot rely on a suitable default preset. Other possible parameters are not indicated here, e.g. because they are just used for a fine tuning of the algorithms or because a default setting exists which is suitable for most of the practical cases.

5.1 Local Approaches

Local approaches to mesh repairing are suitable when input meshes have sparse defects that prevent their exploitation in downstream applications. An algorithm using such an approach locates each specific defect in the mesh and tries to fix it while leaving the remaining model unaltered.

5.1.1 *Manifold Connectivity.* Meshes with non-manifold connectivity, i.e. containing singular edges and vertices, can be categorized into two classes: those that bound so-called regular sets [Mäntylä 1988] and those that do not. Those that bound regular sets still well-define a solid volume – up to the singular contacts at the non-manifold edges and vertices. For such meshes algorithms that split the singular elements into multiple regular elements have been presented as described below. For the general case (e.g. meshes containing singular edges with an odd number of incident faces), posing significant ambiguities, specialized or global methods (cf. Section 5.2) are usually required.

The two algorithms proposed in [Guéziec et al. 2001] and [Rossignac and Cardoze 1999] convert meshes bounding non-manifold regular sets to sets of combinatorially manifold meshes. Strictly speaking, a closed surface mesh can represent a regular set only if it has no self-intersections. Nonetheless [Guéziec et al. 2001] and [Rossignac and Cardoze 1999] are focused on the connectivity only, and hence can process meshes with self-intersections as well. After having identified a singular edge having $2k$ incident faces, it is split into k manifold edges having exactly 2 incident faces each. Such an *edge-manifold* representation may still contain singular vertices that must be identified and duplicated properly.

While in [Rossignac and Cardoze 1999] a strategy is suggested to perform a minimum number of such duplications, [Guéziec et al. 2001] introduced additional operations to join the boundary edges of the mesh cut along the singular edges either by simply contracting boundary loops (pinching) or by stitching nearby boundary curves thus reducing the number of connected components (snapping).

Extensions of these works have been studied for higher dimensions in [De Floriani et al. 2003], where the proposal is to keep some *harmless* singularities as long as the model remains a so-called *initial quasi-manifold*, which is a weaker condition than manifoldness. For the particular case of three dimensions, [Attene et al. 2009] proposes two algorithms to remove singularities from tetrahedral meshes, one guaranteeing combinatorial manifoldness, and the other guaranteeing geometrical manifoldness. Any mesh without degeneracies that bounds a regular set can be

Table IV. Algorithms converting the input to a mesh with **manifold connectivity**. The table reports the type of defects fixed, the requirements on the input, and whether the results are in some sense provably optimal or not. All the algorithms listed here require no parameters and provide guarantees of success without introducing new defects in the mesh. However, algorithms that fix combinatorial singularities only do not remove the possibly corresponding geometric singularities which thus remain in the output.

Algorithm	fixed defects	input requirements	optimal
[Rossignac and Cardoze 1999]	combinat. singularities	no boundary	X
[Guéziec et al. 2001]	combinat. singularities	no boundary	
[Attene et al. 2009] (combinatorial)	combinat. singularities	no boundary,	
[Attene et al. 2009] (geometrical)	combinatorial and geometrical singularities	no boundary, degener., intersect.	

tetrahedralized and processed.

Table IV summarizes the main characteristics of the abovementioned algorithms.

5.1.2 *Gap Closing.* Gaps are usually found between connected components of a mesh, i.e. their boundary is formed by multiple disconnected chains of edges. It is reasonable to assume that such gaps are quite narrow since their most common sources are small tessellation, round-off, and conversion errors, as well as inaccurate trimming or modeling. Hence, gap closing methods usually match boundaries by considering their spatial proximity.

As a simplest variant Rock and Wozny [1992] merge vertices within a prescribed tolerance distance, which allows to re-unite actually equivalent vertices that are slightly displaced due to numerical round-off errors. Other researchers proceed systematically along the boundary edges found in the input mesh and thereby have better control over topological changes performed during the gap closing process. Sheng and Meier [1995] as well as Barequet and Kumar [1997] proceed on a per-edge basis and progressively merge, or "zipper", pairs of boundary edge chains. In order to resolve ambiguities that might be inherent in models with numerous gaps in close proximity, they start from those pairs with smallest distances. Turk and Levoy [1994] handle the special case of "negative gaps", i.e. overlapping patches, by clipping and merging.

Due to their pairwise processing, these methods do not introduce singular edges. Though often desired, this behavior leaves remaining gaps in situations that are only reasonably resolvable by producing a non-manifold mesh. To be more flexible in this regard, Borodin et al. [2002] allow the creation of edges incident to more than two faces, i.e. boundaries of more than two patches can be merged into a common singular edge chain. They further enhance the zippering process by adapting the mesh resolution using edge split operations where necessary before merging. This makes the processing less tessellation-dependent and reduces distortions introduced by edge merging. Patel et al. [2005] further introduce a threshold to choose between two different modes of gap closing: very narrow gaps are closed using zippering, whereas wider gaps are closed by "stitching", which inserts strips of new triangles to avoid shifting vertices too far.

Deviating from the progressive, greedy nature of these methods, Barequet and

Table V. Algorithms for **gap closing**. They all can only guarantee (reasonable) gap-free output if all gaps in the input are narrower than some specified threshold. Using high (or no) thresholds to obtain guaranteed success can result in arbitrarily implausible results.

Algorithm	input requirements	parameters	possible new flaws
[Rock and Wozny 1992]	very small gaps	gap width	intersections, degen., singularities
[Sheng and Meier 1995]	–	gap width	intersections, degen.
[Barequet and Kumar 1997]	–	gap width	intersections, degen.
[Turk and Levoy 1994]	overlapping parts	gap threshold	intersections, degen.
[Borodin et al. 2002]	–	–	intersections, degen., singularities
[Patel et al. 2005]	–	–	intersections, degen., singularities
[Barequet and Sharir 1995]	–	gap threshold	intersections, degen.
[Bischoff and Kobbelt 2005]	–	gap width, resolution	degeneracies

Sharir [1995] tackle the problem differently: first a globally consistent matching of (parts of) boundary curves is determined and, after that, stitching is performed. A good matching is found heuristically using partial curve matching in 3D applied to samplings of the boundary edge chains.

Since they rely on the detection of gaps by analyzing mesh boundaries, all these methods are only able to handle gaps between two (or more) patch boundaries, while gaps between a patch boundary and the interior of another patch as well as gaps between two mesh components without boundaries are not handled. Such gaps are harder to detect and harder to close since the input mesh needs more complex modifications than just merging of edges or vertices. Furthermore note that gaps, just as likely as being "empty space", can also occur in form of small overlaps and often come along with intersections. This hinders the described methods from being able to provide strong guarantees regarding the quality of the output – maybe except for that it contains no boundary edges anymore. Hence, especially for robustness reasons, global repair methods (cf. Section 5.2) might be preferrable for these more general cases. A hybrid variant has been presented by Bischoff and Kobbelt [2005]: gaps as well as intersections are located within a voxel grid. Only in defective voxels new flawless mesh parts are generated and integrated into the intact parts of the input mesh to replace the defective regions. One of the hole-filling methods described in the next section is also able to close gaps in an intersection-free manner [Podolak and Rusinkiewicz 2005].

Table V summarizes the main characteristics of the abovementioned algorithms.

5.1.3 *Hole Filling.* In contrast to gaps, holes usually correspond to missing surface parts. Therefore it is advisable to close them by inserting new geometry, i.e. additional triangles (or more general: polygons).

Early methods detect holes by finding closed loops of boundary edges. Filling is then performed by triangulating these boundary loop polygons. This can for instance be performed by incremental greedy strategies according to simple heuristics [Bøhn and Wozny 1992; Mäkelä and Dolenc 1993; Varnuska et al. 2005], or by more elaborate randomized triangulation methods, that however require the boundary to be parameterized over the plane [Roth and Wibowoo 1997]. Brunton et al. [2010]

propose a simulated annealing based boundary "unfolding" approach that increases the probability that such a planar parameterization can be obtained also for rather complex hole boundaries. Pfeifle and Seidel [1996] heuristically introduce additional vertices in the hole regions to establish a Delaunay-like triangulation.

These methods do not specifically pay attention to the geometric quality of the surface that is produced to fill the holes. Especially for highly non-planar, convolved hole boundaries this might result in very unintuitive results. Considering this, Barequet and Sharir [1995] employ a dynamic programming technique to find a minimum area triangulation for holes. Liepa [2003] further improved this approach by additionally considering discontinuity penalties along the hole boundaries and applying mesh fairing techniques to the constructed hole filling patches. Bac et al. [2008] built upon this and enhanced computational efficiency by performing filling and fairing interleaved in a multi-level procedure, Wei et al. [2010] describe a generalization that takes internal angles, dihedral angles, and areas of the created triangles into account simultaneously. The dynamic programming approach employed by these methods, however, is quite complex, such that the repair process can be extremely time-consuming for holes with hundreds of edges (which are not rare, e.g., in high resolution scans). Zhao et al. [2007] construct hole filling patches by an advancing-front mesh generation method, derive desirable triangle normals from the boundary regions of the input mesh and finally optimize their positions according to a Poisson equation to achieve smoothness. Other techniques that have been applied to derive intuitive geometry for hole regions are Radial Basis Function interpolation [Branch et al. 2006], NURBS fitting [Kumar et al. 2007], curvature energy minimization [Lévy 2003; Pernot et al. 2006], and Moving Least Squares projection [Wang and Oliveira 2007; Tekumalla and Cohen 2004].

The latter method [Tekumalla and Cohen 2004] also attempts to tackle the problem that all the aforementioned methods might very easily introduce self-intersections: potential new triangles are simply tested for intersection with the already existing mesh. However, due to the lack of a circumvention strategy, this might prevent holes from getting filled. For this purpose Wagner et al. [2003] applied a randomized optimization technique (simulated annealing) which can additionally remove triangles to potentially escape such deadlocks. Convergence (to some reasonable, predictable result) can, however, not be guaranteed.

Table VI summarizes the main characteristics of the abovementioned algorithms.

The described methods consider holes locally and fill them with patches with disc-topology. They do not automatically evaluate the spatial relationship of multiple boundary edge loops to come up with potentially better fitting hole patches connecting multiple boundary loops with patches with more complex topologies. This, for instance, leads to the behavior that holes with so-called islands are handled suboptimally: the outer hole is filled without considering the valuable geometric information provided by the interior islands, and the islands themselves are closed to some spurious, often intersecting, "blobs".

In contrast, the approach described by Podolak and Rusinkiewicz [2005] considers the problem globally to locally close holes (and also gaps). The input mesh is first augmented with a tetrahedral space partitioning which is aligned with its polygons. On account of this volumetric representation even extremely complex hole filling

Table VI. Algorithms for **hole filling**. Almost all of these might introduce new degenerate elements and self-intersections, except for the last three that explicitly check for that – all but the very last one, however, at the expense of possibly not being able to fill all holes. Also, for all but this last globally oriented method, holes should be rather simple: highly convolved boundaries result in implausible intersecting filling and islands result in spurious blobs.

Algorithm	input requirem.	parameters	intersect.-free
[Bøhn and Wozny 1992]	–	–	
[Mäkelä and Dolenc 1993]	–	–	
[Varnuska et al. 2005]	–	–	
[Roth and Wibowoo 1997]	roughly planar hole boundaries	–	
[Brunton et al. 2010]	simple boundary loops	–	
[Pfeifle and Seidel 1996]	–	–	
[Barequet and Sharir 1995]	–	–	
[Liepa 2003]	–	–	
[Zhao et al. 2007]	–	–	
[Branch et al. 2006]	–	–	
[Lévy 2003]	–	–	
[Pernot et al. 2006]	–	–	
[Wang and Oliveira 2007]	roughly planar hole boundaries	moving least squares radius	
[Tekumalla and Cohen 2004]	–	MLS radius	X
[Wagner et al. 2003]	–	sim. annealing parameters	X
[Podolak and Rusinkiewicz 2005]	no degen., intersect., singular.	–	X

patches (incorporating islands) can be obtained using graph-cut techniques while still being able to guarantee that they do not cause any intersections. The input is required to be already free of self-intersections, singularities, and degeneracies in order to be able to guarantee manifold output.

The global repair methods presented in Section 5.2 can also be used to handle complex holes reasonably and robustly by exploiting global relationship information, e.g. in a discrete spatial representation. Additionally, some of them are not restricted to holes that are identifiable by boundary edges. Note, however, that due to the fact that holes usually represent missing information (about the geometry as well as about the topology of the hole regions) even these methods can only work heuristically – semi-automatic, interactive approaches (cf. Section 6.2) can be useful to introduce higher-level knowledge about the object to be repaired into the repair process by involving an expert user.

5.1.4 *Mesh Completion.* The hole-filling methods presented in the last section mainly assume (locally) smooth surfaces and create (more or less) smooth hole-filling patches by some form of fairing or interpolation. For non-smooth and structured surfaces this of course leads to implausible fillings. To remedy this issue, more recently quite a number of *mesh completion* methods have been proposed. These in general try to create more plausible fillings, matching in structure, texture, and features, by examining and imitating the information found in the intact parts of the input or in additional example models in some form.

One class of methods proceeds by not actually repairing a given input mesh, but

Table VII. Algorithms for **mesh completion**. Input requirements, mandatory parameters, and possibly introduced flaws are specified. The algorithmic descriptions are often focused on the process of conceiving plausible geometry rather than integrating it into the input mesh. Hence strict statements about the robustness, guarantees, etc., cannot always be made. For the point-based methods of course a mesh reconstruction would have to follow, possibly introducing topological noise and feature chamfering.

Algorithm	input requirem.	parameters	possible new flaws
[Sharf et al. 2004]	– (point-based)	resolution	(top. noise, alias.)
[Bendels et al. 2005]	– (point-based)	# of scale levels	(top. noise, alias.)
[Breckon and Fisher 2005]	– (point-based)	window size	(top. noise, alias.)
[Park et al. 2006]	– (point-based)	resolution	(top. noise, alias.)
[Xiao et al. 2007]	– (point-based)	several	(top. noise, alias.)
[Nguyen et al. 2005]	roughly planar hole boundaries	approx. thresh., # of scale levels	degeneracies, intersections
[Xu et al. 2006]	roughly planar hole boundaries	model-aligned images	degeneracies, intersections
[Jia and Tang 2004]	roughly planar hole boundaries	tensor voting parameters	degeneracies, intersections
[Kraevoy and Sheffer 2005]	manifold with boundary	templates & correspondences	intersections
[Pauly et al. 2005]	–	model database, keywords	degeneracies, intersections

completely “replacing” it by a template model that is fitted to the input by some form of deformation or morphing. This approach of course constrains applicability to specific narrow classes of input meshes and has for instance been used for scans of human heads [Blanz and Vetter 1999; Kähler et al. 2002; Blanz et al. 2004], bodies [Allen et al. 2003; Anguelov et al. 2005], or teeth [Kähler et al. 2002; Savchenko and Kojekine 2002]. Most of these methods require additional information to facilitate the fitting, e.g. in form of user-specified correspondences or feature markers in the input data.

Other methods keep the input mesh and complete it by only filling the holes. Since the merging of hole filling patches with an input mesh is hard to achieve robustly, many of these methods work on point-sampled surfaces instead ([Sharf et al. 2004; Bendels et al. 2005; Breckon and Fisher 2005; Park et al. 2006; Xiao et al. 2007]), avoiding the need to explicitly deal with a mesh topology. Hence they are not *directly* applicable to the *mesh* repair problem: surface sampling and a complete remeshing are implied. In this sense they could be considered global methods (cf. Section 5.2) in the mesh repair context – but in principle it is also imaginable to retain the triangulation in the intact parts while meshing only the hole-filling parts of the completed point set in some way.

These methods can in general be distinguished by whether they consider intra-shape or inter-shape similarities. The first class is particularly useful for cases where the texture and local features of the input itself should be replicated in the hole regions. Algorithms of the latter class, exploiting inter-shape similarities, i.e. examining sets of similar example models, are particularly suitable to achieve a correct global structure and topology even in the case of extremely large amounts of missing data.

Sharf et al. [2004] proposed a method of the former class: inspired by context-based image completion methods they examine the intact parts of the input using a shape similarity measure to find suitable parts to fill hole regions in a coarse-to-fine fashion. A variant of this method is described by Park et al. [2006]. The search is performed in a manner which is discrete regarding location, rotation, and scale. Other methods search on a finer per-point basis [Bendels et al. 2005; Breckon and Fisher 2005]. Nguyen et al. [2005] – inspired by 2D texture synthesis – applied a synthesis approach to local gradient images of a mesh in hole regions. A global planar parameterization of the input mesh is required for this purpose. A similar method (however for point-sampled surfaces) also based on texture synthesis applied to encoded geometric detail is described by Xiao et al. [2007]. Xu et al. [2006] and Brunton et al. [2010] infer the missing geometry information by a shape-from-shading technique resp. a photoconsistency measure applied to photos taken from the real-world equivalent of a scanned object. Jia and Tang [2004] apply a tensor voting approach to derive hole filling geometry, but no detailed information on the actual patch merging is provided.

Kraevoy and Sheffer [2005] presented a method that uses a template mesh that is fit either globally or locally to the incomplete input mesh to then derive hole-filling patches from it. User intervention is needed to obtain correct base-meshes for cross-parameterization purposes in cases of complex topologies or non-trivial hole and gap configurations. Pauly et al. [2005] exploit multiple example models retrieved from shape databases to enhance the completion process. Suitable models are selected according to user-specified keywords and shape similarity. Parts from several models are then cut out in an overlapping manner and merged with the input using the approach described by Turk and Levoy [1994].

Table VII summarizes the characteristics of the mesh completion algorithms.

5.1.5 *Degeneracy Removal.* In [Botsch and Kobbelt 2001] a slicing technique is used to detect and remove degenerate triangles from a manifold mesh. To avoid numerical issues that would probably occur due to the degeneracies, the mesh slicing operator only uses robust predicates to split faces in a controlled manner. Along with a proper decimation scheme, this algorithm is able to remove the degenerate faces from typical meshes generated by tessellation units in CAD systems.

In a different scenario, the algorithm included in [Attene 2010] focuses on raw digitized meshes, and iteratively eliminates nearly degenerate faces using a combination of edge contractions and swaps. In this approach, as well as in [Botsch and Kobbelt 2001], a triangle is declared to be *degenerate* if it has an angle either too acute ($\leq \varepsilon$) or too obtuse ($\geq \pi - \varepsilon$). The algorithm is guaranteed to converge with success when removing only exact degeneracies, that is, when the user-controlled parameter ε is set to zero.

In both methods short edges are contracted to eliminate *needle-like* triangles, thus care must be taken in order to avoid non-manifold results: if a contraction is prevented to avoid such a possible new flaw, the algorithm cannot be guaranteed to eliminate all the degeneracies.

Table VIII summarizes the main characteristics of the abovementioned algorithms.

5.1.6 *Self-Intersection Removal.* The intersections between the various patches of a tessellated CAD model can be located and removed as described in [Bischoff

Table VIII. Algorithms eliminating **degenerate faces** and/or **self-intersections**. The table reports the type of defects fixed (D = degenerate facets, S = self-intersections, H = holes, G = gaps), the requirements on the input, possible parameters, guarantees of success (GS), accuracy of the results (approximated vs. exact). None of these algorithms should introduce new defects.

Algorithm	fixes	input requirem.	parameters	GS	accur.
[Botsch and Kobbelt 2001]	D	manifold	thr. angle ε		approx.
[Attene 2010]	D, S, H	–	thr. angle ε		approx.
[Bischoff and Kobbelt 2005]	S, G	manifold	tolerance ε_0 gap width γ_0	X	approx.
[Campen and Kobbelt 2010]	S	no boundary, no degeneracies	–	X	exact
[Granados et al. 2003]	S	–	–	X	exact

and Kobbelt 2005], where a spatial subdivision (i.e. a voxel grid) grants the efficiency, and accuracy is due to the locality of the approach. Modifications, indeed, take place only within the voxels containing the flaws. There a valid geometry is regenerated. The fixed model is guaranteed to stay within a user-prescribed distance ε_0 from the input. The algorithm also closes gaps that might separate nearby patches up to a user-specified distance γ_0.

The approach presented in [Attene 2010] integrates several repairing algorithms. It has been designed to fix raw digitized meshes and has no requirement on the input. After having converted the mesh to an oriented manifold, it closes the holes and removes degenerate faces. Then, it efficiently locates and repairs self-intersections. In this approach, again, a spatial subdivision is used for efficiency. In contrast to [Bischoff and Kobbelt 2005], however, the self-intersecting triangles are assumed to be small due to the nature of the targeted models, so they are simply removed and the resulting holes are filled. Clearly, this is not guaranteed to work in all the cases, but for the class of models targeted it rarely fails and in general introduces smaller distortions. This approach to self-intersection removal does not require any user-defined parameter.

In general, managing intersecting geometry is a delicate issue because the finite precision used to represent the intersections might not be enough to grant a robust calculation of geometric predicates. In [Campen and Kobbelt 2010] this problem has been avoided by converting the mesh to an intermediate plane-based BSP representation [Bernstein and Fussell 2009], such that the actual calculation of the intersections is unnecessary until the eventual production of the output. When the input is provided with a fixed numerical precision, this algorithm guarantees exactness and robustness while still achieving high performance and low memory consumption. The input mesh must be closed and free of truly degenerate faces that would hamper the computation of the plane-based representation.

An opposite approach is used by [Granados et al. 2003] where arbitrary precision arithmetic is used instead. This method is able to also treat models with open boundaries and dangling/isolated elements, but the approach employed is less efficient and has higher memory requirements.

Table VIII summarizes the main characteristics of the abovementioned algorithms.

5.1.7 *Sharp Feature Restoration.* An interactive approach for restoring corrupted sharp edges has been proposed in [Kobbelt and Botsch 2003]. The user is required to

Table IX. Algorithms that reconstruct corrupted **sharp features**. The table reports the requirements of the input, possible parameters, possible defects newly introduced by the repairing itself.

Algorithm	input requirements	parameters	possible new flaws
[Kobbelt and Botsch 2003]	manifold	interactive	self-intersections
[Attene et al. 2005]	manifold, no degeneracy	–	self-intersections, degeneracies
[Chen and Cheng 2008]	manifold, no degeneracy	–	self-intersections
[Wang 2006]	no niose, no degeneracy	two thresholds	self-intersections

construct a number of "fishbone structures" (spine and orthogonal ribs) which are automatically tessellated to replace the original chamfers. Though not automatic, this method is particularly suitable for simple models with few sharp edges and allows to sharpen the chamfers as well as to modify the swept profiles to produce blends or decorated edges.

The EdgeSharpener method [Attene et al. 2005] provides an automatic procedure for identifying and sharpening chamfered edges and corners. Based on the average dihedral angle at edges smooth regions are grown on the mesh, and the strips of triangles separating neighboring smooth regions are considered *aliasing artifacts* made of chamfer triangles. EdgeSharpener infers the original sharp edges and corners by intersecting planar extrapolations of the smooth regions. Then, chamfer triangles and edges are subdivided, and the newly inserted vertices are moved to the intersections so as to reconstruct the features.

In a different setting, Chen and Cheng [2008] consider the problem of recovering sharp features within smooth patches interpolating filled surface holes. The proposed sharpness-dependent filter is an iterative procedure where each step adjusts the face normals, and the corrupted sharp feature is not required to be represented by a single strip of triangles as in [Attene et al. 2005]. Similarly, motivated by the need to repair meshes produced by dynamic remeshing strategies, [Wang 2006] proposes another feature sharpening approach that allows the presence of vertices in the interior of the chamfers. Contrary to the previously described algorithms, in [Wang 2006] there is no obvious way to automatically deduce whether a patch between two smooth regions is an actual blend or a corrupted sharp feature, so the user is required to set a few parameters (two threshold values) used to distinguish between these cases.

Since all these algorithms *add material* along the corrupted sharp features to reconstruct them, and since there is no control that such new material does not intersect other parts of the surface, these methods can potentially produce self-intersecting meshes. Furthermore, in [Attene et al. 2005] the displacement of newly inserted vertices may also produce degenerate triangles, though this happens only rarely in practice.

Table IX summarizes the characteristics of the abovementioned algorithms.

5.1.8 *Mesh Denoising.* The removal of noise from meshes is a widely studied problem, and a comprehensive discussion of all the existing algorithms in this area would lead us too far from the scope of this survey. Thus, in this section we describe a subset of particularly significant works that well represent the various approaches proposed in this research domain. For a more comprehensive list of existing algorithms we point the reader to the state-of-the-art section in the recent work of Fan et al. [2010].

A very simple iterative approach to mesh denoising is the so-called *Laplacian smoothing* where, at each iteration, all the vertices are moved to the center of mass of their neighbors. This method, however, tends to *shrink* the object while removing the noise. A pioneering contribution preventing this shrinkage is due to Taubin [1995] whose $\lambda|\mu$ algorithm alternates an inward diffusion (controlled by a parameter λ) and an outward diffusion step (controlled by μ). Alternatively, in [Vollmer et al. 1999] the key idea to avoid the shrinking is to push the vertices obtained after each iteration of the Laplacian smoothing back into the direction of the original vertices.

Though being able to successfully reduce the noise, possibly even without shrinking, the aforementioned algorithms are not able to distinguish between a region which is intended to be smooth from a region which is rich of morphological features; consequently such features (which include sharp edges and corners) are smoothed just like all the other regions. To cope with this problem, [Fleishman et al. 2003] adapted the bilateral filtering techniques used in image processing to the case of meshes, thus introducing a first *feature preserving* smoothing algorithm. Besides specifying the number of iterations, the user must tune the bilateral filter based on two parametrs σ_c and σ_s. Similarly, but using a different approach, Jones et al. [2003] introduced a feature-preserving smoothing algorithm that (1) is non-iterative and (2) does not have strict requirements on the mesh connectivity as it can treat also so-called *polygon soups*. Though this method is based on two parameters, their optimal setting can be computed based on the variance of the noise σ_{noise} on the target mesh.

Based on an anisotropic mean curvature flow filtering, the algorithm presented in [Hildebrandt and Polthier 2004] goes one step further, as it is able to actually *sharpen* feature lines while removing the noise in the other regions of the mesh. A slightly more accurate and more efficient approach which is able to better sharpen the features is presented in [Sun et al. 2007].

Finally, by assuming that the surface being modeled by the noisy mesh is piecewise smooth, the method introduced in [Fan et al. 2010] is able to accurately reconstruct all the sharp features lying at the intersection of multiple smooth regions, while removing the noise in all the other parts. This algorithm is particularly suitable to reverse engineering applications dealing with man-made and mechanical objects.

Note that the mesh denoising algorithms described in this section move vertices to new positions, in general without any control that this displacement does not cause the creation of degenerate faces or self-intesecting surface regions.

Table X summarizes the main characteristics of the abovementioned algorithms.

5.1.9 *Topological Noise Removal.* Dealing with topological defects is unavoidable in several applications. For some specific and well-defined cases, prior knowledge about the topology of the mesh being reconstructed can be used to avoid the production of unwanted handles or tunnels. A prominent example of such cases

Table X. Mesh **denoising** algorithms. The table reports the type of repairing (N = noise removal, F = feature preservation, S = feature sharpening), the requirements on the input, and possible parameters (for iterative algorithms, n indicates the number of iterations). All these algorithms are guaranteed to converge with success, though there is no way to prove that the results are satisfactory in general. Also, all of them might generate meshes with degenerate faces and self-intersections.

Algorithm	fixed defects	input requirements	parameters
[Taubin 1995]	N	closed manifold	λ, μ, and n
[Fleishman et al. 2003]	N, F	manifold	σ_c, σ_s, and n
[Jones et al. 2003]	N, F	–	σ_{noise}
[Hildebrandt and Polthier 2004]	N, S	manifold	λ and r
[Fan et al. 2010]	N, S	manifold	n

is the reconstruction of the human brain's cortex from MRI data [Xu et al. 2002], where the desired model is known to be a genus zero surface a priori. For more complex models with handles, tunnels and disconnected components, exploiting prior knowledge to reconstruct a correct topology is more challenging, and methods based on this paradigm may produce models whose topology is correct in a global sense but is not the expected one (e.g. because a handle has been introduced in the wrong part of the model). In some other cases possible ambiguities in the reconstruction of a correct topology can be resolved by involving the user as suggested in [Sharf et al. 2007a]. In many applications, however, this kind of interaction is infeasible and a prior knowledge of the topology is either unavailable or cannot be exploited for a correct reconstruction, such that potential topological errors must be removed in a successive repairing step.

Some topology-correction algorithms work directly on the mesh, whereas many others are designed to treat digital 3D images (i.e. voxel-based representations). This latter class of methods can be used to fix the topology of meshes, too, but only after a voxelization step, which consequently alters the geometry also where it is not strictly necessary. Note that the request for a voxelization implicitly casts these algorithms to the category of the global approaches handled in Section 5.2, but we list them here amongst the others because they are specifically focused on the repair of topological defects.

Mesh-based topology correction With the objective of removing all the handles from a brain's cortex mesh, Fishl et al. [2001] *inflate* the input mesh by alternating steps of Laplacian smoothing and radial projection, so as to map the original surface onto a sphere. Handles in the input are then located by looking for big folds on the mapped surface. The part of the mesh surrounding each handle is removed and the resulting holes are filled using disk-like patches.

Using a local wave front traversal, the algorithm proposed in [Guskov and Wood 2001] discovers the local topology of an oriented manifold mesh and identifies small tunnels and handles. Then non-separating cuts are identified and the mesh is cut and sealed along them, reducing the genus and thus the topological complexity of the mesh. The size of the tunnels and handles to remove is controlled by a user-defined threshold value. A more efficient algorithm tailored to models obtained by digitization is presented in [Attene and Falcidieno 2006]. The gain in speed is based on the observation that, for this class of models, the length of edges does not vary too much across the mesh. Consequently, by considering a depth-first region growing, it is possible to locate handles in the vicinity of splitting points of the region's front. Again, the size of the tunnels and handles to remove is controlled by a user-defined threshold value.

Earlier works in this area include [El-Sana and Varshney 1997], where the idea of α-hulls over point sets was extended to polygon meshes. Intuitively, the idea underlying the algorithm is to simplify the genus of a polygonal mesh by rolling a sphere of radius α over it and filling up all the regions that are not accessible to the sphere. This assumes that the mesh does not have boundaries. Note that besides filling handles, this approach "simplifies" several other morphological features of the surface (e.g. concave sharp edges) which are not accessible to the sphere. This algorithm appears to be extremely difficult to implement and robustness issues probably render it inappropriate for many practical cases.

All these algorithms may *add material* to fill the handles/tunnels, hence the resulting meshes may self-intersect. However, it is worth noticing that in practice undesired handles are typically small, therefore the added material is not overly likely to intersect other parts of the surface.

Voxel-based topology correction With the input being a discrete volumetric representation, the algorithm presented by Wood et al. [2004] performs an axis-aligned sweep through the volume to locate handles, computes their sizes, and selectively removes them. Handles are found by incrementally constructing and analyzing a Reeb graph, and their size is measured by a short non-separating cycle. Their removal is performed on the volume data and the modifications are spatially restricted in order to preserve geometrical detail. The method presented in [Szymczak and Vanderhyde 2003] has the same objective but, instead of computing a Reeb graph, it is based on a simpler morphological operation called *topology-sensitive carving* that makes the method more appropriate to process models with numerous handles – at the expense of typically being less precise. The user sets the maximum number T of allowed topology-altering operations to stop the topological simplification at the desired level.

By using an adaptive grid structure, the method of [Zhou et al. 2007] is capable of processing huge models efficiently at very high resolutions (e.g. 4096^3, processed in a few minutes on a standard PC). In contrast to [Wood et al. 2004], where a Reeb graph is computed, they employ a discrete curve skeleton whose elements are associated with solid parts of the model. This association is performed in a way that the breaking of handles and the filling of tunnels by removing such parts provably does not introduce new unwanted handles. The user is required to specify the processing resolution and two thresholds ε and $\overline{\varepsilon}$ controlling the size of *ring-like* handles to cut and *tunnel-like* handles to fill, respectively.

The algorithm introduced in [Ju et al. 2007] is based on similar principles but goes one step further, as it allows not only to simplify excess topology, but also to actually edit the topology of an object so as to make it equivalent to that of a given target shape. This algorithm can easily remove or add various topological features (e.g., handles, tunnels, or cavities) with minimal modifications of the input, but of course, just like with all the other voxel-based methods, an additional deviation from the input geometry due to the voxelization and meshing steps must be taken into account when dealing with polygon meshes.

Table XI. Algorithms that treat **topological defects**. The table reports the requirements on the input mesh, possible parameters and defects possibly introduced by the repairing itself. Note that [Fischl et al. 2001], [Han et al. 2002] and [Shattuck and Lehay 2001] do not require parameters but are meant to repair only models with zero desired handles (with the distinction of foreground/backgroud handles in [Han et al. 2002]). The lower six methods work on a voxelization, thus the input mesh is required not to have substantial surface holes that would jeopardize the inside/outside classification – furthermore, if their output is meshed again, original features might be chamfered due to the discrete intermediate voxel representation.

Algorithm	input reqs.	parameters	possible new flaws
[El-Sana and Varshney 1997]	no boundary	radius α	self-intersections chamfered feat.
[Guskov and Wood 2001]	oriented manifold	one threshold	self-intersections
[Fischl et al. 2001]	oriented manifold	–	self-intersections
[Attene and Falcidieno 2006]	–	one threshold	self-intersections
[Shattuck and Lehay 2001]	no large holes	–	(chamf. features)
[Han et al. 2002]	no large holes	–	(chamf. features)
[Szymczak and Vanderhyde 2003]	no large holes	one threshold	(chamf. features)
[Wood et al. 2004]	no large holes	one threshold	(chamf. features)
[Zhou et al. 2007]	no large holes	two thresholds	(chamf. features)
[Ju et al. 2007]	no large holes	target "shape"	(chamf. features)

Earlier works in this area include [Shattuck and Lehay 2001], where the segmented MRI image of the brain cortex is processed to remove all the handles and tunnels, and [Han et al. 2002], where the same problem is solved through a more flexible approach which is not limited to the production of genus zero models.

Table XI summarizes the main characteristics of the abovementioned algorithms.

5.2 Global Approaches

The methods described in section 5.1 remove single defects like gaps, holes, singularities, and self-intersections, mainly individually. The absence of these defects, however, is usually not required for their own sake, but as part of (necessary condition for) the greater requirement for manifoldness that is imposed by many downstream applications. While it is relatively easy to check a mesh for combinatorial and geometrical manifoldness, establishing it by these individual, local operations is complicated by the fact that, e.g., the filling of holes or gaps might easily introduce new self-intersections. Furthermore, the ambiguities that emerge (*how to fill a hole, how to connect patch boundaries, how to break singularities and intersections?*) are hard to resolve in a consistent and plausible manner when only performing individual, local investigations.

To account for these issues, methods that repair meshes in a global manner, considering also the mutual relations of individual defects, have been conceived. Since even manifoldness is often not requested for its own sake, but as a condition for the mesh representing the boundary of some solid volume, many of these global repairing methods use some kind of intermediate *volumetric* object representation instead of the polygon mesh *boundary* representation. This enables the use of more meaningful disambiguation rules and allows to easily guarantee correctness by using contouring methods which can guarantee robust re-conversion from the

Table XII. **Global repair algorithms**. The table reports the requirements on the input as well as the general methodology employed for making inside/outside decisions. All algorithms are in principle able to guarantee producing defect-free output (although not necessarily plausible and meaningful) – depending on the final mesh extraction approach employed, however, (nearly) degenerate elements and singularities might be generated, but this can rather easily be prevented. Requirements listed in brackets are not mandatory in order to be able to obtain some manifold output, but hole/gap filling behavior is rather unlikely to be plausible if these are not met.

Algorithm	input requirements	signing method
[Oomes et al. 1997]	no significant holes/gaps	flood-filling
[Andújar et al. 2002]	no significant holes/gaps	flood-filling
[Curless and Levoy 1996]	oriented range meshes	line-of-sight
[Furukawa et al. 2007]	oriented range meshes	line-of-sight
[Davis et al. 2002]	oriented	normals & diffusion
[Guo et al. 2006]	oriented	normals & diffusion
[Masuda 2004]	oriented	normals & diffusion
[Sagawa and Ikeuchi 2008]	oriented	normals & area minimization
[Shen et al. 2004]	oriented	normals & MLS interpolation
[Verdera et al. 2003]	oriented	normals & inpainting
[Ju 2004]	(no significant gaps)	parity-counting
[Nooruddin and Turk 2003]	–	parity-counting, ray-stabbing
[Bischoff et al. 2005]	–	morphology & flooding
[Hétroy et al. 2011]	–	membrane shrinking
[Hornung and Kobbelt 2006]	–	morphology & graph cut
[Spillmann et al. 2006]	–	parity-counting
[Murali and Funkhouser 1997]	(no significant holes)	global sign optimization

intermediate volumetric representation to a consistent manifold mesh free of gaps, holes, degeneracies, and intersections. Hence, the repairing task in such a volumetric setting in its core boils down to deciding which volume parts are inside and which are outside of the represented object (e.g. by assigning signs to a distance function representation). On the downside, often discrete voxel representations are used as intermediate data structures and this might lead to aliasing, i.e. chamfered features and possibly topological noise, in the result – not impairing manifoldness but limiting the quality and accuracy of the output.

It is typical for these methods to implicitly address multiple types of defects. Hence, we do not categorize them by the type of defect treated like the local approaches but by their input requirements. Note that in many cases these requirements are not strict: several of the methods will always output a manifold mesh by their very design, no matter what the input – but the result is (much) less likely to be reasonable if the "requirements" are not fulfilled by the input mesh.

Table XII provides a succinct overview of the global methods surveyed in the following.

5.2.1 *Input without Gaps/Holes.* Early methods simply convert the input mesh to a discrete volumetric representation (the samples either describing a distance function or a binary solid/empty voxel classification) and re-convert it to a polygon mesh. As mentioned above, the crucial step is the sign assignment. When assuming hole-free input (small gaps and holes below voxel size are feasible), this can be done by a simple flood-filling process from seeds known to be inside or outside [Oomes et al. 1997]. Without such information provided additionally, one can at least start from a seed outside of the input's bounding box. This results in the *outer hull* of the input object being obtained – any internal structures (possibly including intentional voids) are discarded [Andújar et al. 2002]. Due to the discretization procedure, as a side effect high frequency detail is suppressed, removing also geometrical and topological noise. This behavior is further strengthened by He et al. [1996] by explicitly applying a low-pass filter.

When a polygon mesh, however, contains significant holes and gaps additional measures need to be taken in order to plausibly assign signs. A variety of techniques for sign assignments has been proposed for such cases. They are presented in the following sections.

5.2.2 *Input with Known Orientation.* Probably the most common source of holes in polygon models is missing information in scans of real-world objects (due to occlusions, etc.). Hence many global mesh repair methods with hole-filling capabilities are tailored to the completion of incomplete scans and the merging of range images. Due to the very nature of the acquisition process, the orientation of the polygons is known. This is often exploited by the methods as valuable initial information for the sign assignment process.

In an early work, Curless and Levoy [1996] exploit line-of-sight information of the scanning process to classify voxels as either "definitely outside" or "possibly inside". This conservative classification closes all holes but leads to implausible excess geometry in larger regions of missing information. Furukawa et al. [2007] employ an additional heuristic and furthermore exploit "line-of-light" information for improvement.

In order to achieve more plausible hole filling behavior Davis et al. [2002] propose a diffusion process. The input is converted to a signed distance field in the vicinity of the surface and this field is iteratively diffused away from the initial surface, effectively closing gaps and holes up to a given width. Guo et al. [2006] and Masuda [2004] describe variants of this approach with different diffusion behavior, specifically suited for non-smooth features.

These methods iteratively shrink holes and gaps. Their boundary sides grow independently and merge arbitrarily. During further iterations they are then implicitly smoothed out. Other methods consider a hole region globally and might thus be able to obtain more plausible results. Sagawa and Ikeuchi [2003; 2008] initially classify voxels as inside or outside depending on the input's normals. Then voxels along the interfaces in hole regions are iteratively reclassified so as to locally minimize the interface area. Shen et al. [2004] describe an implicit surface construction based on a moving least squares interpolation of the input polygon mesh. Holes and gaps up to a given width are globally bridged. Since evaluation of the implicit function is expensive, a hierarchical evaluation and caching scheme is proposed. Verdera et al. [2003] apply partial differential equations borrowed from image inpainting approaches to hole regions in order to obtain smooth fillings. Unfortunately, setting up the necessary boundary conditions can be quite involved for non-simple hole and gap constellations.

5.2.3 *Arbitrary Input.* Ju [2004] describes a method to patch holes in a voxelized version of an input mesh. The output is guaranteed to be a closed mesh, but due to the explicit tracing of hole boundary loops for filling, gaps whose boundary does not form a closed loop are unlikely to get bridged. Furthermore, notice that so far we were mainly concerned with *missing* information in form of holes and gaps. However, models can as well contain excess geometry that needs to be discarded to obtain a proper manifold. For instance a model might have been constructed by simply putting several models together – resulting in double walls or interpenetrations. These internal structures and non-manifold configurations might result in spurious geometry and can disturb the reconstruction, e.g. when the popular dual contouring method is applied the output could become a non-manifold regular set [Mäntylä 1988].

One possibility to deal with internal structures as well as non-manifold configurations is to simply disregard them by focusing only on the *outer hull* of the object. Nooruddin and Turk [2003] do this by considering all voxels between the first and the last intersections of numerous rays with the input as inside. In case of holes and gaps this strategy of course leads to misclassifications. Therefore they perform this *ray-stabbing* for several ray directions. This leads to plausible classifications except for regions very close to holes and gaps where the resulting geometry can be rather unintuitive.

Bischoff et al. [2005] also focus on the outer hull but handle holes and gaps using morphological closing operations applied to the voxelized input. The outer hull can afterwards be found by flood-filling the outside. Computational efficiency is achieved by utilizing an adaptive octree-based voxelization scheme in conjunction with hole and gap boundary detectors to spatially restrict morphology. Discrete membrane shrinking can also be employed to obtain the outer hull [Hétroy et al. 2011]. In this work, however, the morphological operators are employed globally, altering the surface also in concave intact regions. Hornung and Kobbelt [2006] also (roughly) determine the outer region by morphological operations and flood-filling, but the actual surface is then found by a global graph-cut approach resulting in minimal-area hole filling patches. This approach is especially useful when dealing with uncertain data, e.g. noise-ridden input or imperfectly registered partial scans.

These methods, of course, remove not only redundant internal geometry but also voids that might be intentional. Methods based on parity counting [Nooruddin and Turk 2003; Spillmann et al. 2006] are able to correctly preserve internal voids – but they only lead to reasonable results if all internal structures are intentional.

The approach presented by Murali and Funkhouser [1997] is quite different from all others presented in this section. First of all, it does not affect the original geometry in intact mesh regions (however, the tessellation is altered). This is achieved by employing a volumetric space partitioning into arbitrary polyhedral cells aligned with the input polygons (using a BSP-tree) instead of regular voxels. Then, it also is able to meaningfully deal with input with internal structures without just discarding them or requiring that they form only consistent voids. This is achieved by, in a global optimization process, establishing a sign assignment for the volume cells such that the resulting output (the interface between positive and negative cells) conforms with the input as well as possible. The geometry of the hole filling patches is, however, highly dependent on the structure of the employed BSP-tree and can be very unpleasing in cases of large holes that are to be filled.

6. DISCUSSION AND OPEN PROBLEMS

After the overview of defects and algorithms presented in this survey, it is evident that some repair tasks are significantly more challenging than others. While, e.g., the problems of converting a regular set to a manifold configuration or consistently orienting faces can be easily formalized and solved, providing robust and intelligent algorithms for, e.g., hole filling, gap closing, and intersection removal requires a non-trivial interpretation of the problem and an elaborate case-by-case study. Especially for these *difficult* defect types existing methods, despite their vast number, leave room for future investigation of the underlying problem, possibly from novel perspectives following different paradigms. The research directions we in particular deem important and worthwhile following are outlined in this section.

6.1 Guaranteeing and Global, yet Minimally Invasive

For the *difficult* tasks, most of the existing local approaches are characterized by a lack of guarantees and possible new flaws introduced by the repairing itself. On the other hand, the field of global strategies provides much more robust algorithms with output of guaranteed correctness – at the cost of impairing also the intact mesh parts due to global conversions and remeshing. This rule of thumb, however, has noticeable exceptions such as [Podolak and Rusinkiewicz 2005] and [Murali and Funkhouser 1997] which are able to guarantee manifold output while (at least geometrically) leaving the intact parts unchanged. This comes with limitations, e.g. the input is required to be oriented consistently and free of degeneracies, singularities and self-intersections [Podolak and Rusinkiewicz 2005], or the patches filling holes are shaped rather arbitrarily and robust implementation is involved [Murali and Funkhouser 1997]. Nevertheless, these two works show that it is possible to devise repairing algorithms that can guarantee a successful processing while modifying the input only where strictly necessary. In this context it is also worth mentioning hybrid algorithms (e.g. [Bischoff and Kobbelt 2005]) that employ a discrete voxel structure (like most global approaches) to easily achieve robustness and guarantees but perform the conversion and reconstruction only locally where defects exist. Since guarantees are highly desirable and a global remeshing implies several undesirable disadvantages for various application scenarios, we believe that this field of methods that can guarantee defect-free output while impairing the input as little as possible is an important area for further research.

6.2 High-Level Interaction incorporating Meta-Knowledge

No matter how robust, accurate and intelligent these *automatic* methods will become, missing or excess geometry always implies ill-posedness of the underlying repair problem that then bears several ambiguities: holes can be filled by patches with different geometry, multiple holes be filled and connected by patches of different topology, self-intersections be removed in one or the other way, and so on. Although a fully automatic repairing of mesh models is highly desirable, this goal is thus hard to achieve – at least if by “repairing” we mean more than just the resolution of all the abovementioned defects and flaws in an *arbitrary* way.

All the automatic repair methods presented in the previous section use some heuristics, tie-breaking rules, random decisions, or some restricted problem specification to disambiguate these cases. While this might be acceptable in some applications, in others the repaired model has to adhere to some higher-level semantics to be acceptable. These can be hard to formalize and therefore hard to model and integrate into a specialized repairing algorithm, so that in such cases today the most realistic approach to solve these problems is to incorporate a human observer, which has qualified knowledge and understanding of the underlying semantics and requirements, into the repairing process by allowing to interactively resolve the ambiguities that arise. This is particularly true when a method is to be designed to cope with a wide spectrum of types of models. Of course, on one extreme, if the algorithm can assume that the input model belongs to a specific class that can be clearly defined, then one may justly expect that the repairing proceeds automatically and successfully in any case. In practice, however, often flexibility is desired and repairing algorithms are expected to deal with wider spectra of input models. Therefore, we believe that it is worth designing new algorithms which are as automatic as possible, but also incorporate interaction metaphors to allow intervention where disambiguation is necessary.

First steps in this direction have been undertaken in the past. Morvan and Fadel [1996], Barequet et al. [1998], and Attene and Falcidieno [2006] describe model repair systems incorporating several of the surveyed repair techniques that allow the user to explicitly select boundaries to be connected, intersections to be resolved, etc., so as to resolve the inherent ambiguities by exploiting the user's meta-knowledge of the object under consideration. This process is partly supported by error visualization and automatic zooming techniques. Kobbelt and Botsch [2003] describe an approach providing the user with the possibility to manually define elaborate blends between smooth surfaces as well as surface features to be reconstructed. In these methods, the interaction happens on a rather low level: the user is required to inspect individual defects, possibly deal with the selection of individual edges or vertices, and so on. This might lead to a rather tedious and costly repair process – especially when large models containing millions of faces and thousands of small defects and flaws are dealt with.

Interaction on a somewhat higher level is employed, e.g., in the template-based mesh completion method of Kraevoy and Sheffer [2005] where the user specifies some correspondences between (part of) the incomplete input mesh and (some part of) a template model, or in Hétroy et al. [2011] where the user is presented with several morphologically altered variants of the input differing topologically to make choices.

In the field of surface reconstruction from point clouds, Sharf at el. [2007b] presented an interesting approach that analyzes the local topological stability of the reconstruction and prompts the user to make a qualified decision in unstable regions. This is supported by presenting the region of interest in an optimally perceivable way and allowing the user to provide input very easily in form of simple "scribbling" on a 2D plane. The extension and incorporation of such or similar analysis and visualization techniques in conjunction with intuitive high-level selection and modification metaphors into mesh repair approaches could significantly ease the process of turning raw models into, not only syntactically correct, but also semantically valid and acceptable output.

6.3 Vertical Integration to Repair Workflows

Another important direction for future research concerns the creation of accurate repairing workflows for specific scenarios. If there is the need to keep high accuracy in intact regions, one will usually prefer to rely on local repair approaches. Most of these methods, however, are tailored to fix one specific type of defect, usually require the input to satisfy several requirements, and often produce output with potential new problems. Thus, using several algorithms in sequence to produce a repairing workflow may not be as easy as expected because there might be multiple compatibility gaps between the output produced by one algorithm and the input of its successor in the pipeline. Interactive repair systems [Barequet et al. 1998; Attene and Falcidieno 2006] face this verticalization or serialization problem by basically letting the user choose the order of processing, possibly implying several iterations until finally a model is obtained that fulfills the desired quality criteria. For the particular case of repairing raw digitized meshes this problem has been studied in [Attene 2010], where the pipeline may have some "loops" to account for more attempts when the output quality of one of the algorithms is not sufficient for its successor. We believe that the workflow-based analysis is worth further investigation to account for the many other scenarios that may require, today or in the future, a non-trivial mesh repairing process.

7. AVAILABLE REPAIR TOOLS

Several mesh repair tools implementing one or multiple of the surveyed methods are freely available – some in open-source, some in closed-source form. To ensure up-to-dateness we provide information and references to these software tools on the accompanying website `http://www.meshrepair.org`.

Acknowledgements

This work was partly supported by the EU project Visionair (grant agreement N. 262044) and by the CNR activity N. ICT.P10.009 (Advanced Techniques for Digital 3D Shape Analysis and Synthesis – "Multimodal and Multidimensional Content and Media" project). Part of this work was also funded by the DFG Cluster of Excellence *UMIC* (DFG EXC 89).

REFERENCES

Allen, B., Curless, B., and Popović, Z. 2003. The space of human body shapes: reconstruction and parameterization from range scans. *ACM Transactions on Graphics (Proc. SIGGRAPH) 22,* 3, 587–594.

Alliez, P., Ucelli, G., Gotsman, C., and Attene, M. 2008. Recent advances in remeshing of surfaces. In *Shape Analysis and Structuring*, L. De Floriani and M. Spagnuolo, Eds. Springer, Chapter 2, 53–82.

Andújar, C., Brunet, P., and Ayala, D. 2002. Topology-reducing surface simplification using a discrete solid representation. *ACM Transactions on Graphics 21,* 2, 88–105.

Anguelov, D., Srinivasan, P., Koller, D., Thrun, S., Rodgers, J., and Davis, J. 2005. Scape: shape completion and animation of people. *ACM Transactions on Graphics (Proc. SIGGRAPH) 24,* 3, 408–416.

Attene, M. 2010. A lightweight approach to repair polygon meshes. *The Visual Computer*, 1393–1406.

Attene, M. and Falcidieno, B. 2006. ReMESH: An interactive environment to edit and repair triangle meshes. In *Shape Modeling and Applications*. 271–276.

Attene, M., Falcidieno, B., Rossignac, J., and Spagnuolo, M. 2005. Sharpen&Bend: Recovering curved sharp edges in triangle meshes produced by feature-insensitive sampling. *IEEE Trans. Vis. and Comp. Graph. 11,* 2, 181–192.

Attene, M., Giorgi, D., Ferri, M., and Falcidieno, B. 2009. On converting sets of tetrahedra to combinatorial and pl manifolds. *Computer-Aided Geometric Design 26,* 8, 850–864.

Bac, A., Tran, N.-V., and Daniel, M. 2008. A multistep approach to restoration of locally undersampled meshes. In *Advances in Geometric Modeling and Processing*. 272–289.

Barequet, G., Duncan, C., and Kumar, S. 1998. RSVP: A geometric toolkit for controlled repair of solid models. *IEEE Transactions on Visualization and Computer Graphics 4,* 2, 162–177.

Barequet, G. and Kumar, S. 1997. Repairing CAD models. In *VIS '97: Proceedings of the 8th conference on Visualization '97*. IEEE Computer Society Press, Los Alamitos, CA, USA, 363–370.

Barequet, G. and Sharir, M. 1995. Filling gaps in the boundary of a polyhedron. *Computer-Aided Geometric Design 12,* 2, 207–229.

Bendels, G. H., Schnabel, R., and Klein, R. 2005. Detail-preserving surface inpainting. In *The 6th International Symposium on Virtual Reality, Archaeology and Cultural Heritage (VAST)*. Eurographics Association, 41–48.

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

Bernstein, G. and Fussell, D. 2009. Fast, exact, linear booleans. *Computer Graphics Forum 28,* 5, 1269–1278.

Bischoff, S. and Kobbelt, L. 2005. Structure preserving CAD model repair. *Computer Graphics Forum 24,* 3, 527–536.

Bischoff, S., Pavic, D., and Kobbelt, L. 2005. Automatic restoration of polygon models. *ACM Transactions on Graphics 24,* 4, 1332–1352.

Blanz, V., Mehl, A., Vetter, T., and Seidel, H.-P. 2004. A statistical method for robust 3D surface reconstruction from sparse data. In *2nd International Symposium on 3D Data Processing, Visualization and Transmission (3DPVT 2004)*. 293–300.

Blanz, V. and Vetter, T. 1999. A morphable model for the synthesis of 3D faces. In *SIGGRAPH '99: Proceedings of the 26th annual conference on Computer graphics and interactive techniques*. ACM, New York, NY, USA, 187–194.

Bøhn, J. H. and Wozny, M. J. 1992. A topology-based approach for shell-closure. In *Selected and Expanded Papers from the IFIP TC5/WG5.2 Working Conference on Geometric Modeling for Product Realization*. North-Holland Publishing Co., Amsterdam, The Netherlands, 297–319.

Borodin, P., Novotni, M., and Klein, R. 2002. Progressive gap closing for mesh repairing. In *Advances in Modelling, Animation and Rendering*. 201–213.

Botsch, M. and Kobbelt, L. 2001. A robust procedure to eliminate degenerate faces from triangle meshes. In *Vision, Modeling and Visualization*. 283–290.

Branch, J., Prieto, F., and Boulanger, P. 2006. Automatic hole-filling of triangular meshes using local radial basis function. In *3DPVT '06: Proceedings of the Third International Symposium on 3D Data Processing, Visualization, and Transmission (3DPVT'06)*. IEEE Computer Society, Washington, DC, USA, 727–734.

Breckon, T. P. and Fisher, R. B. 2005. Non-parametric 3D surface completion. In *3DIM '05: Proceedings of the Fifth International Conference on 3-D Digital Imaging and Modeling*. IEEE Computer Society, Washington, DC, USA, 573–580.

Brunton, A., Wuhrer, S., Shu, C., Bose, P., and Demaine, E. 2010. Filling holes in triangular meshes using digital images by curve unfolding. *International Journal of Shape Modeling 16,* 1–2, 151–171.

Campen, M. and Kobbelt, L. 2010. Exact and robust (self-)intersections for polygonal meshes. *Computer Graphics Forum 29,* 2, 397–406.

Chen, C.-Y. and Cheng, K.-Y. 2008. A sharpness dependent filter for recovering sharp features in repaired 3D mesh models. *IEEE Trans. Vis. and Comp. Graph. 14,* 1, 200–212.

Curless, B. and Levoy, M. 1996. A volumetric method for building complex models from range images. In *SIGGRAPH '96: Proceedings of the 23rd annual conference on Computer graphics and interactive techniques*. ACM, New York, NY, USA, 303–312.

Curless, B. and Seitz, S. 2000. 3D Photography. *Course Notes for SIGGRAPH 2000*.

Davis, J., Marschner, S. R., Garr, M., and Levoy, M. 2002. Filling holes in complex surfaces using volumetric diffusion. In *1st International Symposium on 3D Data Processing Visualization and Transmission (3DPVT 2002)*. 428–438.

De Floriani, L., Morando, F., and Puppo, E. 2003. Representation of non-manifold objects through decomposition into nearly manifold parts. In *ACM Solid Modeling*. 304–309.

El-Sana, J. and Varshney, A. 1997. Controlled simplification of genus for polygonal models. In *IEEE Visualization 97 proceedings*. 403–410.

Fan, H., Yu, Y., and Peng, Q. 2010. Robust feature-preserving mesh denoising based on consistent subneighborhoods. *Visualization and Computer Graphics, IEEE Transactions on 16,* 2, 312–324.

Farouki, R. 1999. Closing the gap between CAD model and downstream application. *SIAM news 32,* 5, 303–319.

Fischl, B., Liu, A., and Dale, A. M. 2001. Automated manifold surgery: Constructing geometrically accurate and topologically correct models of the human cerebral cortex. *IEEE Transactions on Medical Imaging 20,* 1, 70–80.

Fleishman, S., Drori, I., and Cohen-Or, D. 2003. Bilateral mesh denoising. *ACM Transactions on Graphics (Proc. SIGGRAPH) 22,* 3, 950–953.

Furukawa, R., Itano, T., Morisaka, A., and Kawasaki, H. 2007. Improved space carving method for merging and interpolating multiple range images using information of light sources of active stereo. In *ACCV 2007, 8th Asian Conference on Computer Vision*. 206–216.

Granados, M., Hachenberger, P., Hert, S., Kettner, L., Mehlhorn, K., and Seel, M. 2003. Boolean operations on 3D selective Nef complexes: Data structure, algorithms, and implementation. *Algorithms-ESA 2003*, 654–666.

Guéziec, A., Taubin, G., Lazarus, F., and Horn, B. 2001. Cutting and stitching: Converting sets of polygons to manifold surfaces. *IEEE Transactions on Visualization and Computer Graphics 7,* 2, 136–151.

Guo, T.-Q., Li, J.-J., Weng, J.-G., and Zhuang, Y.-T. 2006. Filling holes in complex surfaces using oriented voxel diffusion. In *Proc. 2006 International Conference on Machine Learning and Cybernetics*. 4370–4375.

Guskov, I. and Wood, Z. 2001. Topological noise removal. In *Proceedings of Graphics Interface*. 19–26.

Han, X., Xu, C., Braga-Neto, U., and Prince, J. L. 2002. Topology correction in brain cortex segmentation using a multiscale, graph-based algorithm. *IEEE Transactions on Medical Imaging 21,* 2, 109–121.

He, T., Hong, L., Varshney, A., and Wang, S. W. 1996. Controlled topology simplification. *IEEE Transactions on Visualization and Computer Graphics 2*, 171–184.

Hétroy, F., Rey, S., Andujar, C., Brunet, P., and Vinacua, A. 2011. Mesh repair with user-friendly topology control. *Computer-Aided Design 43*, 101–113.

Hildebrandt, K. and Polthier, K. 2004. Anisotropic filtering of non-linear surface features. *Computer Graphics Forum 23*, 391–400.

Hornung, A. and Kobbelt, L. 2006. Robust reconstruction of watertight 3D models from non-uniformly sampled point clouds without normal information. In *Proc. Eurographics Symposium on Geometry Processing*. 41–50.

Igarashi, T., Matsuoka, S., and Tanaka, H. 1999. Teddy: a sketching interface for 3D freeform design. In *SIGGRAPH '99: Proceedings of the 26th annual conference on Computer graphics and interactive techniques*. ACM New York, NY, USA.

Jia, J. and Tang, C.-K. 2004. Inference of segmented color and texture description by tensor voting. *IEEE Trans. Pattern Anal. Mach. Intell. 26,* 6, 771–786.

JONES, T. R., DURAND, F., AND DESBRUN, M. 2003. Non-iterative, feature-preserving mesh smoothing. *ACM Transactions on Graphics 22*, 943–949.

JU, T. 2004. Robust repair of polygonal models. *ACM Transactions on Graphics (Proc. SIGGRAPH) 23,* 3, 888–895.

JU, T. 2009. Fixing geometric errors on polygonal models: A survey. *Computer Science and Technology 24,* 1, 19–29.

JU, T., ZHOU, Q.-Y., AND HU, S.-M. 2007. Editing the topology of 3D models by sketching. *ACM Transactions on Graphics (Proc. SIGGRAPH) 26,* 3, 42–1–42–9.

KÄHLER, K., HABER, J., YAMAUCHI, H., AND SEIDEL, H.-P. 2002. Head Shop: generating animated head models with anatomical structure. In *SCA '02: Proceedings of the 2002 ACM SIGGRAPH/Eurographics Symposium on Computer Animation.* ACM, New York, NY, USA, 55–63.

KOBBELT, L. AND BOTSCH, M. 2003. Feature sensitive mesh processing. In *SCCG 03: Proceedings of the 19th Spring Conference on Computer Graphics.* 17–22.

KRAEVOY, V. AND SHEFFER, A. 2005. Template-based mesh completion. In *Eurographics Symposium on Geometry Processing.* 13–22.

KUMAR, A., SHIH, A. M., ITO, Y., ROSS, D. H., AND SONI, B. K. 2007. A hole-filling algorithm using non-uniform rational b-splines. In *Proc. 16th International Meshing Roundtable.* 169–182.

LÉVY, B. 2003. Dual domain extrapolation. *ACM Transactions on Graphics 22,* 3, 364–369.

LIEPA, P. 2003. Filling holes in meshes. In *Proc. Eurographics Symposium on Geometry Processing.* 200–205.

LUEBKE, D. 2001. A developer's survey of polygonal simplification algorithms. *Computer Graphics and Applications 21,* 3, 24–35.

MÄKELÄ, I. AND DOLENC, A. 1993. Some efficient procedures for correcting triangulated models. In *Proc. Symp. Solid Freeform Fabrication.* 126–134.

MÄNTYLÄ, M. 1988. *Introduction to Solid Modeling.* WH Freeman & Co. New York, NY, USA.

MASUDA, T. 2004. Filling the signed distance field by fitting local quadrics. In *3DPVT '04: Proceedings of the 3D Data Processing, Visualization, and Transmission, 2nd International Symposium.* IEEE Computer Society, Washington, DC, USA, 1003–1010.

MORVAN, S. M. AND FADEL, G. M. 1996. IVECS, interactively correcting STL files in a virtual environment. In *Solid Freeform Fabrication Symposium.* 491–498.

MURALI, T. AND FUNKHOUSER, T. 1997. Consistent solid and boundary representations from arbitrary polygonal data. In *Proceedings of Symposium on Interactive 3D Graphics.* 155–162.

NGUYEN, M. X., YUAN, X., AND CHEN, B. 2005. Geometry completion and detail generation by texture synthesis. *The Visual Computer 21,* 8-10, 669–678.

NIELSON, G., HOLLIDAY, D., AND ROXBOROUGH, T. 1999. Cracking the cracking problem with Coons patches. In *IEEE Visualization '99 proceedings.* 285–290.

NOORUDDIN, F. AND TURK, G. 2003. Simplification and repair of polygonal models using volumetric techniques. *ACM Transactions on Visualization and Computer Graphics 9,* 2, 191–205.

OOMES, S., SNOEREN, P., AND DIJKSTRA, T. 1997. 3D shape representation: Transforming polygons into voxels. In *Proc. 1st Intnl. Conf. on Scale-Space Theory in Computer Vision.* 349–352.

PARK, S., GUO, X., SHIN, H., AND QIN, H. 2006. Surface completion for shape and appearance. *Vis. Comput. 22,* 3, 168–180.

PATEL, P. S., MARCUM, D. L., AND REMOTIGUE, M. G. 2005. Stitching and filling: Creating conformal faceted geometry. In *Procs of 14th International Meshing Roundtable.* 239–256.

PAULY, M., MITRA, N. J., GIESEN, J., GROSS, M., AND GUIBAS, L. J. 2005. Example-based 3D scan completion. In *Eurographics Symposium on Geometry Processing.* 23–32.

PERNOT, J. P., MORARU, G., AND VERON, P. 2006. Filling holes in meshes using a mechanical model to simulate the curvature variation minimization. *Comput. Graph. 30,* 6, 892–902.

PFEIFLE, R. AND SEIDEL, H.-P. 1996. Triangular b-splines for blending and filling of polygonal holes. In *GI '96: Proceedings of the conference on Graphics interface '96.* Canadian Information Processing Society, Toronto, Ontario, Canada, 186–193.

Podolak, J. and Rusinkiewicz, S. 2005. Atomic volumes for mesh completion. In *Eurographics Symposium on Geometry Processing*. 33–42.

Rock, S. and Wozny, M. J. 1992. Generating topological information from a bucket of facets. In *Solid freeform fabrication symposium proceedings*. 251–259.

Rossignac, J. and Cardoze, D. 1999. Matchmaker: manifold BReps for non-manifold r-sets. In *Procs of 5th ACM symposium on Solid modeling and applications*. 31–41.

Roth, G. and Wibowoo, E. 1997. An efficient volumetric method for building closed triangular meshes from 3-d image and point data. In *Proceedings of the conference on Graphics interface '97*. Canadian Information Processing Society, Toronto, Ont., Canada, Canada, 173–180.

Sagawa, R. and Ikeuchi, K. 2003. Taking consensus of signed distance field for complementing unobservable surface. *International Conference on 3D Digital Imaging and Modeling*, 410–417.

Sagawa, R. and Ikeuchi, K. 2008. Hole filling of a 3D model by flipping signs of a signed distance field in adaptive resolution. *IEEE Trans. Pattern Anal. Mach. Intell. 30,* 4, 686–699.

Savchenko, V. and Kojekine, N. 2002. An approach to blend surfaces. In *Advances in Modeling, Animation and Rendering*. 139–150.

Seitz, S. M., Curless, B., Diebel, J., Scharstein, D., and Szeliski, R. 2006. A comparison and evaluation of multi-view stereo reconstruction algorithms. *Proceedings of the 2006 IEEE Computer Society Conference on Computer Vision and Pattern Recognition*, 519–526.

Sharf, A., Alexa, M., and Cohen-Or, D. 2004. Context-based surface completion. *ACM Transactions on Graphics (Proc. SIGGRAPH) 23,* 3, 878–887.

Sharf, A., Lewiner, T., Shklarski, G., Toledo, S., and Cohen-Or, D. 2007a. Interactive topology-aware surface reconstruction. *ACM Transactions on Graphics (Proc. SIGGRAPH) 26,* 3, 43–1–43–9.

Sharf, A., Lewiner, T., Shklarski, G., Toledo, S., and Cohen-Or, D. 2007b. Interactive topology-aware surface reconstruction. *ACM Transactions on Graphics (Proc. SIGGRAPH) 26,* 3.

Shattuck, D. W. and Lehay, R. M. 2001. Automated graph based analysis and correction of cortical volume topology. *IEEE Transactions on Medical Imaging 20,* 11, 1167–1177.

Shen, C., O'Brien, J. F., and Shewchuk, J. R. 2004. Interpolating and approximating implicit surfaces from polygon soup. *ACM Transactions on Graphics (Proc. SIGGRAPH) 23,* 3, 896–904.

Sheng, X. and Meier, I. R. 1995. Generating topological structures for surface models. *IEEE Comput. Graph. Appl. 15,* 6, 35–41.

Shewchuk, J. R. 2002. Delaunay refinement algorithms for triangular mesh generation. *Computational Geometry: Theory and Applications 22,* 1-3, 21–74.

Spillmann, J., Wagner, M., and Teschner, M. 2006. Robust tetrahedral meshing of triangle soups. In *Proc. Vision, Modeling, Visualization (VMV)*. 9–16.

Sun, X., Rosin, P., Martin, R., and Langbein, F. 2007. Fast and effective feature-preserving mesh denoising. *IEEE transactions on visualization and computer graphics 13,* 5, 925–938.

Szymczak, A. and Vanderhyde, J. 2003. Extraction of topologically simple isosurfaces from volume datasets. In *IEEE Visualization*. 67–74.

Taubin, G. 1995. A signal processing approach to fair surface design. In *SIGGRAPH '95: Proceedings of the 22nd annual conference on Computer graphics and interactive techniques*. ACM, New York, NY, USA, 351–358.

Tekumalla, L. S. and Cohen, E. 2004. A hole-filling algorithm for triangular meshes. Tech. rep., School of Computing, University of Utah.

Turk, G. and Levoy, M. 1994. Zippered polygon meshes from range images. In *SIGGRAPH '94: Proceedings of the 21st annual conference on Computer graphics and interactive techniques*. ACM, New York, NY, USA, 311–318.

Turk, G. and O'Brien, J. 2002. Modelling with implicit surfaces that interpolate. *ACM Transactions on Graphics 21,* 4, 855–873.

Varnuska, M., Parus, J., and Kolingerova, I. 2005. Simple holes triangulation in surface reconstruction. In *Proc. Algorithmy*. 280–289.

VERDERA, J., CASELLES, V., BERTALMIO, M., AND SAPIRO, G. 2003. Inpainting surface holes. In *Int. Conference on Image Processing*. 903–906.

VOLLMER, J., MENCL, R., AND MUELLER, H. 1999. Improved Laplacian smoothing of noisy surface meshes. *Computer Graphics Forum 18,* 3, 131–138.

WAGNER, M., LABSIK, U., AND GREINER, G. 2003. Repairing non-manifold triangle meshes using simulated annealing. In *Proc. of The 4th Israel-Korea Bi-National Conference on Geometric Modeling and Computer Graphics*. 88–93.

WANG, C. 2006. Incremental reconstruction of sharp edges on mesh surfaces. *Computer-Aided Design 38,* 6, 689–702.

WANG, J. AND OLIVEIRA, M. M. 2007. Filling holes on locally smooth surfaces reconstructed from point clouds. *Image Vision Comput. 25,* 1, 103–113.

WEI, M., WU, J., AND PANG, M. 2010. An integrated approach to filling holes in meshes. In *Proceedings of the 2010 International Conference on Artificial Intelligence and Computational Intelligence*. IEEE Computer Society, Washington, DC, USA, 306–310.

WOOD, Z., HOPPE, H., DESBRUN, M., AND SCHROEDER, P. 2004. Removing excess topology from isosurfaces. *ACM Transactions on Graphics 23,* 2, 190–208.

XIAO, C., ZHENG, W., MIAO, Y., ZHAO, Y., AND PENG, Q. 2007. A unified method for appearance and geometry completion of point set surfaces. *Vis. Comput. 23,* 6, 433–443.

XU, C., PHAM, D., RETTMANN, M., YU, D., AND PRINCE, J. 2002. Reconstruction of the human cerebral cortex from magnetic resonance images. *IEEE Transactions on Medical Imaging 18,* 6, 467–480.

XU, S., GEORGHIADES, A., RUSHMEIER, H., DORSEY, J., AND MCMILLAN, L. 2006. Image guided geometry inference. In *3DPVT '06: Proceedings of the Third International Symposium on 3D Data Processing, Visualization, and Transmission.* IEEE Computer Society, Washington, DC, USA, 310–317.

ZHANG, R., TSAI, P., CRYER, J., AND SHAH, M. 1999. Shape-from-Shading: A survey. *IEEE Transactions on Pattern Analysis and Machine Intelligence 21,* 8, 690–706.

ZHANG, Y., ROHLING, R., AND PAI, D. 2002. Direct surface extraction from 3D freehand ultrasound images. In *IEEE Visualization*. 45–52.

ZHAO, W., GAO, S., AND LIN, H. 2007. A robust hole-filling algorithm for triangular mesh. *The Visual Computer 23,* 12, 897–997.

ZHOU, Q., JU, T., AND HU, S. 2007. Topology repair of solid models using skeletons. *IEEE Transactions on Visualization and Computer Graphics 13,* 4, 675–685.