Pacific Graphics (2019)
N. N, N. N, and N. N (Editors)

Connectivity-preserving Smooth Surface Filling with Sharp Features

Thibault Lescoat[1,2] Pooran Memari[2,3] Jean-Marc Thiery[1] Maks Ovsjanikov[2] Tamy Boubekeur[1]
[1] LTCI, Télécom Paris Institut Polytechnique de Paris
[2] LIX, École Polytechnique Institut Polytechnique de Paris
[3] CNRS

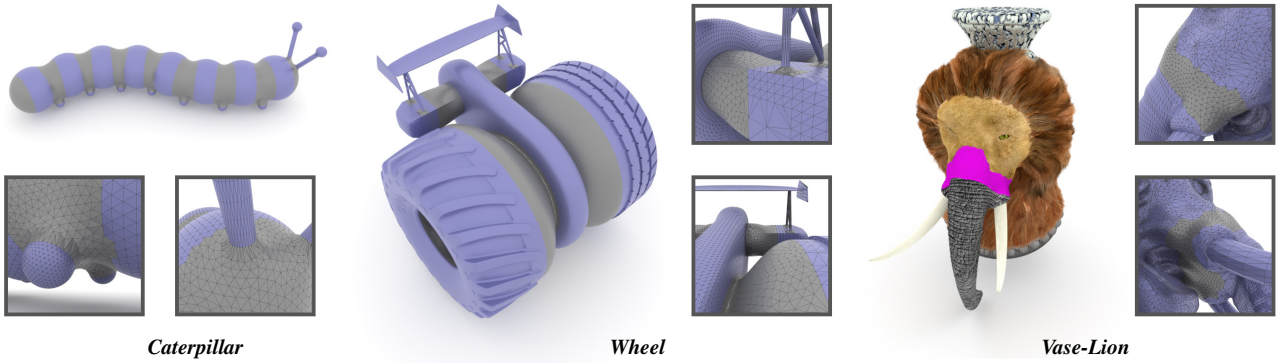


Figure 1: *Starting with an arrangement of mesh parts (in blue) with open boundaries, we stitch them together by only adding new triangles (in gray), so that inputs are contained in the output. We propagate feature-lines and adjust the surface in between to minimize the variation of curvature, making our method suitable for both organic and man-made shapes. The automatic completion of the vase-lion and an elephant trunk (right, generated surface in pink) requires strict connectivity preservation, as all components come with multiple per vertex UV coordinates and per face materials.*

Abstract

We present a method for constructing a surface mesh filling gaps between the boundaries of multiple disconnected input components. Unlike previous works, our method pays special attention to preserving both the connectivity and large-scale geometric features of input parts, while maintaining efficiency and scalability w.r.t. mesh complexity. Starting from an implicit surface reconstruction matching the parts' boundaries, we first introduce a modified dual contouring algorithm which stitches a meshed contour to the input components while preserving their connectivity. We then show how to deform the reconstructed mesh to respect the boundary geometry and preserve sharp feature lines, smoothly blending them when necessary. As a result, our reconstructed surface is smooth and propagates the feature lines of the input. We demonstrate on a wide variety of input shapes that our method is scalable to large input complexity and results in superior mesh quality compared to existing techniques.

CCS Concepts

• ***Computing methodologies*** → ***Shape modeling;*** *Mesh models; Mesh geometry models;*

1. Introduction

High-quality 3D content is central to a wide range of applications in both computer graphics and other related areas, including virtual reality, special effects and gaming to name a few. However, while consumption of 3D content is enabled by more powerful hardware and processing techniques, 3D content *creation* remains difficult, and is typically reserved to highly trained professionals. In recent years, novel modeling paradigms have been introduced, significantly expanding the possibilities for content creation, both by automatically synthesizing novel 3D shapes using e.g. parametric models, and modeling by example, introduced in the pioneering work of Funkhouser et al. [FKS*04]. This latter approach is especially appealing since it allows to combine parts of 3D shapes found in rich content repositories to quickly create new content.

Motivation. Despite significant improvements over the past decade for both suggesting appropriate geometric parts to compose and synthetizing new shapes, stitching multiple geometric parts into a single coherent object is still a core challenge. Unfortunately, this step usually involves remeshing a significant part of the input

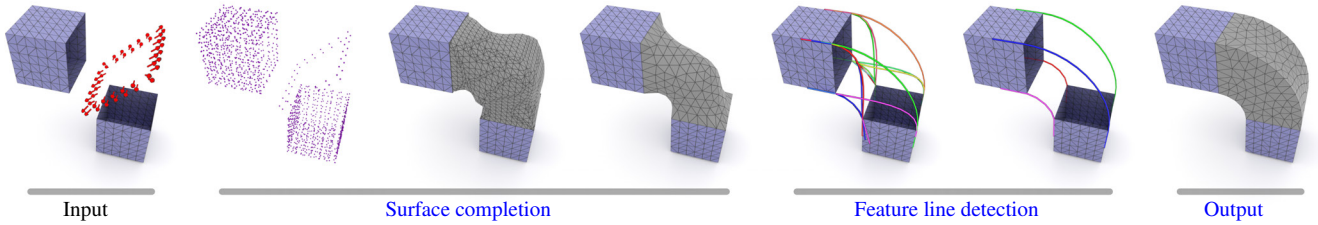


Figure 2: ***Overview***. *Starting from input mesh parts assembled as one triangular mesh, we first sample it for the initial surface reconstruction, then mesh the resulting Poisson implicit surface with our mesh-aware Dual Contouring and remesh the result to prevent numerical instabilities in the laplacian reconstruction (Section 4). Next, we determine salient points and candidates feature lines, keeping only those minimizing a tri-harmonic power cost (Section 5.1). Last, we stretch the surface between the feature lines (Section 5.3).*

e.g., with standard reconstruction techniques, and often distorts the input geometry for the sake of robustness and smoothness. This severely limits the utility of such techniques, as input shapes are often carefully modeled and thus their connectivity and key features should be preserved during stitching. For instance, aside from per-vertex UV coordinates, hair/fur is also connectivity-dependent as it is often modeled as particles emitted from a subset of faces (Figure 1 right).

In this paper, we aim at synthesizing a high quality surface filling the space in-between at set of existing parts, propagating dominant feature lines while delaying other form of high frequency signal propagation to map-based methods e.g., normal/displacement map in-painting.

Contributions. We introduce a novel technique for filling gaps between the boundaries of several non-overlapping input meshes. Our approach is inspired by implicit surface reconstruction techniques – notably, we leverage Screened Poisson Reconstruction [KH13] – but we introduce several crucial modifications that significantly improve the resulting mesh quality. Namely,

- a **mesh-aware dual contouring algorithm**, that can stitch an implicit surface contour to existing mesh components while preserving their connectivity,
- a **feature-sensitive mesh deformation mechanism** to comply with the boundary geometry and preserve sharp feature lines, smoothly blending them when necessary.

2. Related work

Mesh fusion. Wuttke et al. [WPM12] show how to zip non-overlapping boundaries by only adding triangles, provided gaps are small and triangles have similar sizes. For artistic creation, *Snap-Paste* [SBSCO06] consists of series of global-to-local transformations for positioning overlapping objects using *Soft-ICP*. While very effective for some configurations, the requirement for overlaps still strongly restricts the space of possible input arrangements, e.g. forbidding long range connections. Schmidt and Brochu [SB16] provide a robust algorithm to fuse meshes with boolean operations, which does not, however, fill gaps between open boundaries, nor preserve connectivity. Similarly, Yu et al. [YZX*04] merge and deform meshes by solving a Poisson system, but are limited to near-overlapping objects. Reducing mesh completion to a boolean operation (optionally with Poisson-based deformations) would require to align components, which is generally an ill-posed problem.

Point-cloud in-painting. Self-similarity helps filling holes: Sharf et al. [SACO04] complete holes in a point cloud with parts of the same point cloud in a multiresolution manner. By enforcing spatial-coherency [HTG14], filled holes mimic regions of the shape and fade-in seamlessly. However, these methods handle point clouds with no connectivity and cannot reconstruct long sharp features.

Curve networks. One could propagate feature lines and then fill the resulting curve-delimited patches [ZJC13, PLS*15, SHBSS16]. Speed-wise, triangulating curves [ZJC13] depends on the topology of the input holes, restricting the complexity of the input arrangement (e.g. it failed for several shapes of this article), while not addressing the surface geometry. Recent surfacing methods [PLS*15, SHBSS16] require the input curve network to be closed and connected and can result in the loss of input connectivity. One could also rigidly transform mesh parts and align them for completion [HGCO*12], before repositioning them at their input location and adjusting the stitching surface. However, feature lines found this way would disregard user placement, and under certain rotations they could intersect the surface (as in Figure 6 middle).

Stitching via parameterization. Dual Domain Extrapolation [Lév03] can be used to stitch components, by parameterizing the input parts into the same parameter, completing the parts in this space, and then updating the 3D geometry. However, stitching multiple topological disks into a single genus-0 surface requires extra seams, inducing tedious user intervention. Instead, we do not need user intervention, although interactive control is permitted.

Stitching with implicit surfaces. Implicit surfaces, based on *Radial Basis Functions* [JLW*06, LJWH08] or *Screened Poisson Reconstruction* [CPS15, CS18], have been used to stitch shapes. Such methods are oblivious to the topology of the gaps to fill and mesh the relevant portions of the implicit surfaces using off-the-shelve Marching Cubes, Marching Triangles [HSIW96] or Dual Contouring [JLSW02, SW03, SJW07]. One could also cut and zip the resulting mesh to the input; in contrast, we introduce a mesh-aware Dual Contouring that does not need such post-processing.

3. Overview

Our method composes and stitches *non-overlapping* triangular meshes $\mathcal{M}_1 ... \mathcal{M}_n$, completing the gaps between their boundaries, without modifying the connectivities and geometry embeddings. In the following, the input mesh is $\mathcal{M} = \cup_{i=1}^{n} \mathcal{M}_i$ i.e., meshes to compose are the disconnected components of $\mathcal{M}$. We also require the boundaries to be 1-manifold, although we do not require the rest of the mesh to be 2-manifold. Our method outputs a triangular mesh and can be decomposed into three main stages (see Figure 2):

1. components are stitched by reconstructing a surface between the boundaries while preserving the input connectivity (Section 4),
2. feature lines are determined using spatial reasoning to associate detected salient points (Sections 5.1 & 5.2),
3. the created surface is smoothly optimized in-between the feature lines (Section 5.3).

4. Surface completion

The role of this stage is to reconstruct a mesh $\mathcal{M}' = \mathcal{M} \cup \mathcal{Z}$ such that $\mathcal{Z}$ is manifold and covers the gaps between the boundaries of $\mathcal{M}$. We start by resampling $\mathcal{M}$ for a faithful Screened Poisson Reconstruction [KH13], yielding an implicit surface $\mathcal{S}$ and an octree, that we re-use to mesh only the relevant zones of $\mathcal{S}$, while stitching it to $\mathcal{M}$. Lastly, we clean $\mathcal{Z}$. At this stage, the embedding of $\mathcal{Z}$ is not critical and will be optimized later.

4.1. Sampling and implicit shape completion

We exploit the implicit surface $\mathcal{S}$ generated by Screened Poisson Reconstruction [KBH06, KH13], and structured in an octree of depth D, to sustain the smooth completion of disjoint parts. For a correct meshing, $\mathcal{S}$ has to be close to the input, thus we randomly sample points from the input triangles and merge them in the cells of a 2^D-side regular grid. This also makes $\mathcal{S}$ independent of the input density, which can vary greatly in artist-authored meshes (Figure 1 right). Optionally, we can enrich this set with *user-specified guiding samples*, to favor or prevent parts from getting connected during the Poisson reconstruction (see Figures 2 & 8, red arrows).

4.2. Connectivity-preserving dual-contouring

Equipped with $\mathcal{S}$, we mesh only the portion filling the gaps using a new variant of Dual Contouring [JLSW02] designed specifically to preserve the original connectivity of $\mathcal{M}$, and which also re-uses the octree generated by the Poisson reconstruction. In Dual Contouring, octree leaves are associated with (at most) 1 vertex, created

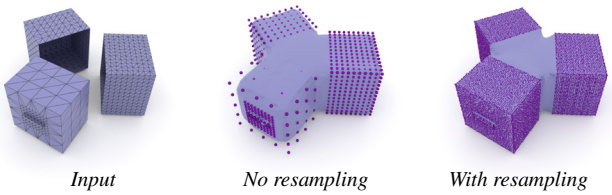


Figure 3: *(left) input with varying vertex density, (middle & right) surface from Poisson reconstruction, samples in purple.*

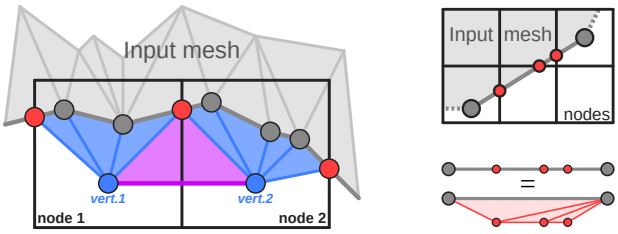


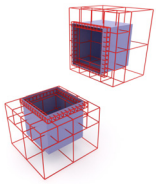
Figure 4: *With boundary vertices in gray, node vertices in blue, and intersection vertices in red, we add **blue triangles** inside nodes, and **purple triangles** between nodes sharing a vertex, and "subdivide" edges by adding 0-area triangle fans (right).*

on-the-fly, and only *minimal edges* of the octree where the sign of $\mathcal{S}$ changes yield a polygon: these are node edges that do not contain any other smaller edge of the octree. They are shared between 3 or 4 nodes, and can be found using a top-down traversal.

Our **mesh-aware dual contouring** (see Algorithm 1) enriches the original algorithm with two preprocessing stages to form a connectivity-preserving filling. We detail them in the following.

Extending the boundaries of $\mathcal{M}$. Here we link the boundaries of $\mathcal{M}$ to the vertices of the octree leaves. We denote by $\mathbf{vertex}(n)$ the vertex associated to a node n. First, for each leaf node n and boundary edge e whose intersection with n is a segment (v_1, v_2), with the same orientation as e and $v_1 \neq v_2$, we create the triangle $(\mathbf{vertex}(n), v_2, v_1)$ (Figure 4, blue triangles). Instead of subdividing boundary edges, we add new vertices and link them to the edge extremities via a triangle fan (Figure 4 right). To avoid ambiguities, we ensure that at least one point of (v_1, v_2) lies in $[n_l, n_h[$, where $n_l, n_h \in \mathbb{R}^3$ are the min and max coordinates of the cube of n. While some 0-area triangles may arise near boundaries, the connectivity is valid and the input has been preserved. Second, for each vertex v included in exactly 2 triangles $t_1 = (\mathbf{vertex}(n_1), v, ...)$ and $t_2 = (\mathbf{vertex}(n_2), ..., v)$, we add $(\mathbf{vertex}(n_1), \mathbf{vertex}(n_2), v)$, which is improducible by usual dual contouring (Figure 4 purple triangles). All added triangles are correctly oriented, and the new boundary can be stitched using standard dual contouring.

Pruning minimal edges. A straightforward dual contour would contain redundant elements near the input and non-manifold vertices at the new boundary. Instead, we exclude nodes that intersect $\mathcal{M}$ but not its boundary edges, as well as nodes with an excluded ancestor (in red in the inset figure). No minimal edge contained in these nodes will be considered when meshing $\mathcal{S}$.



4.3. Reaching manifoldness

For the cover mesh $\mathcal{Z}$ to be manifold, three conditions must be met: **(i)** $\mathcal{S}$ should be smooth enough that there are no nodes with several possible meshings; this is generally the case since we re-use the octree of the Poisson reconstruction; **(ii)** there should be at most 1 boundary edge crossing any face of a leaf node, so that cross-nodes edges from the boundary extension (Figure 4, purple)

are in 1 face before the usual dual-contouring; this is mostly the case as we are dealing with artist-authored meshes with relatively clean boundaries. **(iii)**, nodes intersecting $\mathcal{M}$ should also intersects $\mathcal{S}$, so that the dual-contouring connect $\mathcal{Z}$ to $\mathcal{M}$. This condition is rarely verified near sharp edges, thus we enforce it by simplifying the octree (Figure 5) before pruning minimal edges, in a bottom-up fashion: if one of the children of n violates this condition, we remove them and merge their vertices into **vertex**(n); we do not simplify past a minimum depth or if the topology of the intersection between border and nodes would change.

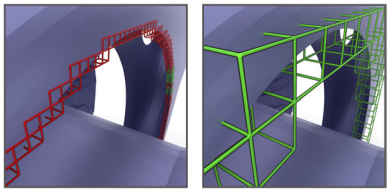


Figure 5: *Octree nodes intersecting a boundary loop, before (left) and after (right) the simplification. Nodes in red do not intersect $\mathcal{S}$ while those in green do.*

Finally, we repeatedly remove non-manifold edges and vertices from $\mathcal{Z}$ (if any) and fill the resulting holes using triangle fans around their center. All our test models were manifold at this stage.

4.4. Smooth remeshing

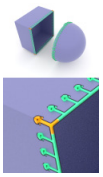
While manifold, $\mathcal{Z}$ now typically includes slivers, which would incur numerical instabilities in the remaining stages of our method. Thus, we remesh it via the iterative scheme of Botsch and Kobbelt [BK04], using the average boundary edge length as target length. This gives $\mathcal{Z}$ a regular connectivity while smoothly connecting the components of $\mathcal{M}$; here only the new vertices of $\mathcal{Z}$ can move, removing all degenerate triangles introduced previously.

5. Piecewise smooth surface

To reach a natural aesthetic completion, we minimize the curvature variation of $\mathcal{Z}$ by minimizing the surface tri-laplacian. To avoid smoothing out the sharp features near the boundaries of $\mathcal{M}$ during this minimization, we first propagate them onto $\mathcal{Z}$. To do so, we detect and process feature lines differently from the rest of $\mathcal{Z}$ and perform the geometric regularization in a piecewise fashion.

5.1. Candidate feature lines

Feature points. We first detect feature points on $\mathcal{M}$: these are the boundary vertices v where the surface forms a sharp edge, i.e., the angle between adjacent boundary faces is greater than a user-specified threshold. We also compute the direction of sharpness d_v (see inset), and τ_v, the direction toward the interior.



Associations. We now make candidate associations for these feature points, forming the endpoints of candidate feature lines, and define *full* associations $(v \to w)$ with $v \neq w$, and *partial* ones $(v \to \emptyset)$, from v to the closest point on another component of $\mathcal{M}$. In particular, we forbid "inverted" associations, i.e. where $d_v \cdot d_w \leq 0$ and $\tau_v \cdot \tau_w \leq 0$ to avoid self-intersections (Figure 6)

Algorithm 1 Connectivity-preserving Dual Contouring

Inputs: mesh $\mathcal{M}$, octree $\mathcal{O}$, implicit surface $\mathcal{S}$
Output: mesh $\mathcal{M}' = \mathcal{M} \cup \mathcal{Z}$
Extend boundaries of $\mathcal{M}$ into $\mathcal{O}$
(optional) Simplify $\mathcal{O}$ to improve quality
Prune minimal edges of $\mathcal{O}$
Dual-contouring on $\mathcal{O}$, generating $\mathcal{Z}$
Clean resulting mesh

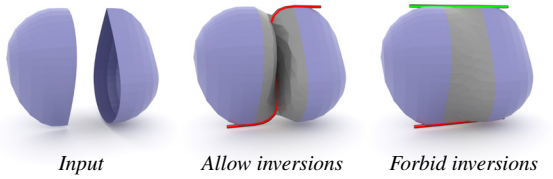


Figure 6: *Allowing "inversions" can lead to self-intersections.*

Chains. Feature lines are defined by the list of adjacent vertices ("chain") composing it. For each association, we compute these chains using Dijkstra's algorithm over $\mathcal{Z}$, and adjust their geometry with a tri-harmonic reconstruction, using their endpoints position and sharp edge direction as boundary conditions. The cost of a chain is the sum of the squared tri-laplacian at each of its vertices.

5.2. Selection of feature lines

Armed with a set of candidate chains, each with its associated cost, we want to select a subset that minimizes the sum of costs, maximizes the number of included feature points and with no shared vertices. These criteria help reducing ambiguity in the chains selection process while conforming as much as possible to the smoothness that will be imposed on the rest of $\mathcal{Z}$. No matter the cost, covering more feature points is more important than having a lower cost.

Let us consider the graph where nodes represent candidate chains and edges represent the absence of intersection i.e., an edge means that the candidate chains associated to its nodes do not intersect. It follows that the desired subset of chains is the maximal clique which covers the most feature points with the lowest cost possible. We use the Bron-Kerbosch algorithm [BK73] to find the cliques, in parallel on all components of $\mathcal{Z}$.

5.3. Smooth surface

The final geometric embedding of $\mathcal{Z}$ is obtained by iteratively remeshing (2 iterations are usually sufficient) and adjusting X, the vertex positions of $\mathcal{Z}$, via a tri-harmonic reconstruction (which we found to give the best tradeoff between smoothing and preserving low frequency details). As anchors, we use the vertices of $\mathcal{M}$ (up to a certain topological distance from $\mathcal{Z}$) and the anchors used for the chain costs:

- Remesh $\mathcal{Z}$ (except anchors) using iterative remeshing [BK04]

- Adjust positions by solving $HX = 0$ under the constraint that the position of the anchors do not change.

The operator $H = (WL)^3 + (I - W)C^3$ allows to reconstruct both feature lines and surface at the same time, with

- W, $W_{i,j} = 1$ only if $i = j$ and vertex i is not on a chain,
- C, uniform discretization of the 1D laplacian on chains,
- L, cotangent discretization of the 2D laplacian on the surface,

Note that multiplying W by L before the exponentiation prevents the variation of curvature to be minimized across features lines, effectively making this regularization piecewise.

6. Results

Our method can handle complex input arrangements, thanks to the use of a volumetric reconstruction method, and is suitable for both organic and CAD-like shapes because of the minimization of the curvature variation and the propagation of hard edges. We implemented our method in C++ and report timings in Table 1, measured on an Intel Core-i7 at 3.60Ghz with 8GB of memory. We also evaluate our method in comparison with state-of-the-art mesh completion technique [CS18], see Figure 7.

6.1. Performance

We can observe from Table 1 that, in general, the computation is dominated by the initial filling step, especially by the Poisson reconstruction, and reconstructing feature lines and final surface only take a small fraction of the total time. The computation of the feature lines can take a significant portion of the run-time when many salient points are detected (typical in man-made shapes); moreover, the geometry adjustment of $\mathcal{Z}$ is close to linear in the number of added vertices, which tend to increase when the gap to complete is large. The completion of Lin et al. [LJWH08] uses marching cubes on a grid, which quickly degrades in performance and memory when the resolution is increased to cope with finer details. Although we do not have timings on our examples for the method of Centin and Signoroni [CS18], they show that their runtime is dominated by the Poisson reconstruction in presence of simple boundaries, and by the mesher (a modified marching triangles) for complex boundaries. Since we use Poisson only for filling the gaps, we can use it at coarse scale as long as it correctly captures the finest gaps. Thus, shapes with coarse boundary geometry are fast to complete. Last, the completed surface can be cached, for improved interactivity.

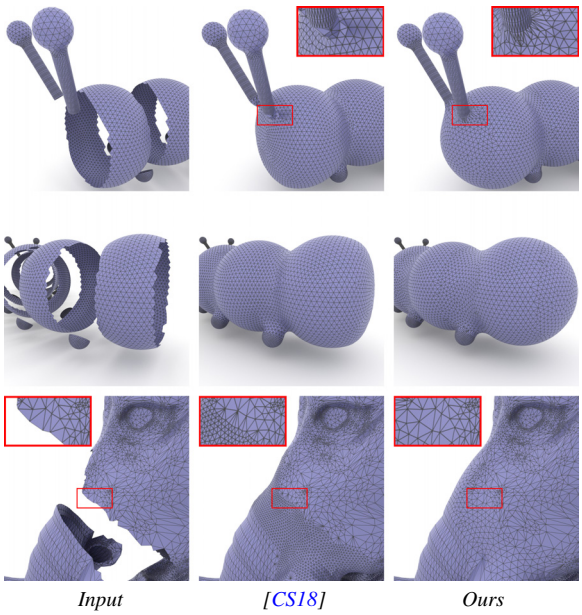


Figure 7: *Comparing our method to [CS18] for complex shapes.*

6.2. User control

Users interact with our operator mainly by positioning in 3D space the different parts, and while our method treats them as a unique mesh with separate components, their actual placement can be easily done with any 3D modeler. Both compared methods treat the input similarly, although Lin et al. [LJWH08] also use guiding primitives (such as spheres, cones...) for a faster modeling process.

In our framework, users can optionally *guide* the way the components are connected, by introducing additional oriented points in the Poisson reconstruction input (Section 4). Indeed, oriented points near a hole with outward normals will prevent its connection to the rest of the shape, and distant components can be connected by adding oriented points coarsely following the desired surface (Figure 8). The same control is compatible with the method of Centin et Signoroni [CS18] as well as with the naïve method consisting in using Poisson reconstruction only, while Lin et al. [LJWH08] propose a sketch-based control mechanism. Unless explicitly specified, all our figures are made without these additional control points.

Our framework also allows to control the reconstructed feature lines, letting users select the feature lines to reconstruct among the candidate pool or even drawing them; users can also bias the selection toward full or partial associations via a multiplicative factor on the cost of the partial associations (Figure 9).

6.3. Connectivity preservation

When fusing shapes, naïvely meshing the implicit surface from the Poisson reconstruction will remesh the input, while our method and the one of Lin et al. [LJWH08] preserve the input mesh. For

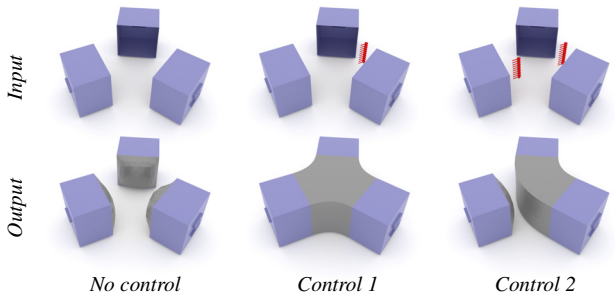


Figure 8: *The three cubes are too distant to be connected (left), but this can be fixed by adding control points (red arrows, middle). By including such points elsewhere, one can force a separation (right).*

Example		Input			Timings (seconds)				Output	
		Vertices	Triangles	*D*	Filling	Chains	Surface	Total	Vertices	Triangles
Overview	*(Fig. 2)*	434	720	5	0.400	0.003	0.034	0.437	595	1186
Tri-control	*(Fig. 8)*	1209	2272	6	0.582	0.006	0.068	0.656	1552	3096
Cube-sphere	*(Inset. 6.4)*	2374	4588	5	0.429	0.008	0.148	0.585	3088	6172
Cube-offset5	*(Fig. 9)*	5602	10880	5	0.590	0.040	1.236	1.866	10190	20376
Caterpillar	*(Fig. 1)*	15534	29084	8	2.497	0.011	2.642	5.150	22891	45778
Wheel	*(Fig. 1)*	63142	124865	7	3.737	1.080	3.569	8.386	72662	145280
Nightmare	*(Fig. 10)*	80318	159740	7	2.227	0.063	2.675	4.964	86329	172666
Crab-plane	*(Fig. 10)*	142617	284471	6	1.954	0.105	2.903	4.962	149576	299200
Vase-lion	*(Fig. 1)*	203669	407092	6	2.239	0.138	0.340	2.717	204564	409124

Table 1: *Completion timings in seconds. The octree depth D is user-controlled and greatly effects the completion speed (Section 4).*

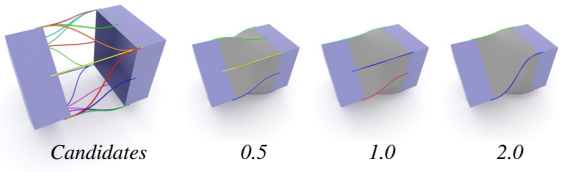
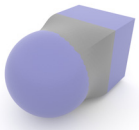


Figure 9: *The cost of partial chains is multiplied with a user parameter, to bias the selection toward full or partial chains.*

input meshes with sufficient vertex density, the method of Centin and Signoroni [CS18] also preserves the connectivity (they target dense mesh processing). Otherwise, they modify it mainly near the boundaries in a pre-processing step (Figure 7). Their implicit surface heavily depends on the input connectivity, as input vertices are directly fed to the Poisson reconstruction. This not only makes preserving the connectivity harder (especially near sharp edges), but can also change the position of the input vertices.

In contrast, we guarantee that the input connectivity and geometry is preserved. The effects of the sampling step are illustrated in Figure 3.

6.4. Smoothness and sharp edges



To offer high quality output meshes, we designed our method to produce piecewise smooth surfaces, and feature-lines can smoothly blend into the surface (see inset). All methods compared here [LJWH08, CS18] generate smooth surfaces, as they all use smooth implicit surfaces with regularity constraints. However, the overall shape from Poisson reconstruction depends on the gap size, and can exhibit an important curvature variation, as the surface tends to shrink between distant components. The RBFs used by Lin et al. [LJWH08] also have this problem, and often need silhouette constraints for a more faithful surface. We bypass this issue by explicitly minimizing the variation of curvature (Section 5.3).

The aforementioned implicit surfaces do not model sharp features, precisely because of the regularity constraints. Furthermore, Marching Cubes and Marching Triangles would erase them: for Marching Cubes, this known limitation is addressed in Dual Contouring [JLSW02]. Despite the robustness of the mesher of Centin and Signoroni [CS18], it cannot preserve feature lines because of triangle regularity constraints, although this could be handled in their circular-arc bisection, by creating irregular triangles when detecting discontinuities in the normal field of the implicit surface.

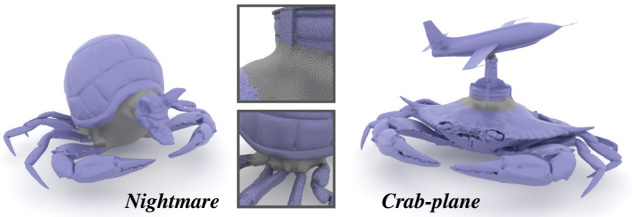


Figure 10: *Composition of complex shapes, the cover mesh of nightmare fills 13 open boundary loops.*

Propagating sharp lines is an ill-posed problem in general, as for the same input there exist multiple solutions depending on the user's goal, e.g. considering 2 half-cubes, one can choose whether to smoothly blend them into the surface (Figure 9). We handle this by specifically propagating sharp edges after the meshing phase, and we avoid coupling the feature lines selection from their geometric reconstruction, allowing for a precise user control.

6.5. Limitations and future work

As first limitation, our operator cannot handle more than one feature line starting from any given point, and we do not handle quad meshes. Also, the Poisson implicit surface can sometimes exhibit tunnels near sharp edges. The tri-laplacian operator can be numerically unstable for very high vertex densities, and is sensitive to mid-frequency noise near boundaries. Our algorithm could be localized, to avoid computing the whole Poisson implicit surface if only a small region needs to be completed. Another future work is to propagate all high-frequency details (geometry, UVs, etc.).

7. Conclusion

We have proposed a novel approach for constructing a single mesh from several parts while exactly preserving the input as well as propagating feature lines when present. Our method is composed of three main steps that decorrelate the generation of the filling connectivity from its actual geometry embedding. Our feature line reconstruction strategy produces high quality results in challenging scenarios e.g. involving CAD-like parts. Most importantly, the input mesh structure is entirely preserved, together with any attributes it may carry e.g., existing UV parameterization, which offers a flexible workflow when modeling by composition.

Acknowledgments

We thank Lin et al. [LJWH08] and Centin and Signoroni [CS18] for their help with the comparisons. We also thank Marie-Paule Cani for her suggestions of test models. Parts of this work were supported by the ERC Starting Grant StG-2017-758800 (EXPROTEA), KAUST OSR Award CRG-2017-3426 and ANR grant 16-LCV2-0009-01 ALLEGORI.

References

[BK73] BRON C., KERBOSCH J.: Algorithm 457: Finding all cliques of an undirected graph. *Commun. ACM 16*, 9 (Sept. 1973), 575–577. 4

[BK04] BOTSCH M., KOBBELT L.: A remeshing approach to multiresolution modeling. In *Proc. SGP* (2004), SGP '04, pp. 185–192. 4

[CPS15] CENTIN M., PEZZOTTI N., SIGNORONI A.: Poisson-driven seamless completion of triangular meshes. *Computer Aided Geometric Design 35-36* (2015), 42 – 55. 2

[CS18] CENTIN M., SIGNORONI A.: Advancing mesh completion for digital modeling and manufacturing. *Computer Aided Geometric Design 62* (2018), 73 – 90. 2, 5, 6, 7

[FKS*04] FUNKHOUSER T., KAZHDAN M., SHILANE P., MIN P., KIEFER W., TAL A., RUSINKIEWICZ S., DOBKIN D.: Modeling by example. In *ACM Transactions on Graphics (Proc. Siggraph)* (New York, NY, USA, 2004), SIGGRAPH '04, pp. 652–663. 1

[HGCO*12] HUANG H., GONG M., COHEN-OR D., OUYANG Y., TAN F., ZHANG H.: Field-guided registration for feature-conforming shape composition. *ACM Transactions on Graphics (Proceedings of SIGGRAPH ASIA 2012) 31* (2012), 171:1–171:11. 2

[HSIW96] HILTON A., STODDART A. J., ILLINGWORTH J., WINDEATT T.: Marching triangles: range image fusion for complex object modelling. In *Proceedings of 3rd IEEE International Conference on Image Processing* (Sep. 1996), vol. 2, pp. 381–384 vol.2. 2

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

[JLSW02] JU T., LOSASSO F., SCHAEFER S., WARREN J.: Dual contouring of hermite data. *ACM Trans. Graph. 21*, 3 (2002). 2, 3, 6

[JLW*06] JIN X., LIN J., WANG C. C., FENG J., SUN H.: Mesh fusion using functional blending on topologically incompatible sections. *The Visual Computer 22*, 4 (2006), 266. 2

[KBH06] KAZHDAN M., BOLITHO M., HOPPE H.: Poisson surface reconstruction. In *Proceedings of the Fourth Eurographics Symposium on Geometry Processing* (Aire-la-Ville, Switzerland, Switzerland, 2006), SGP '06, Eurographics Association, pp. 61–70. 3

[KH13] KAZHDAN M., HOPPE H.: Screened poisson surface reconstruction. *ACM Trans. Graph. 32*, 3 (July 2013), 29:1–29:13. 2, 3

[Lév03] LÉVY B.: Dual domain extrapolation. *ACM Trans. Graph. 22*, 3 (July 2003), 364–369. 2

[LJWH08] LIN J., JIN X., WANG C., HUI K.-C.: Mesh composition on models with arbitrary boundary topology. *IEEE transactions on visualization and computer graphics 14*, 3 (2008), 653–665. 2, 5, 6, 7

[PLS*15] PAN H., LIU Y., SHEFFER A., VINING N., LI C.-J., WANG W.: Flow aligned surfacing of curve networks. *ACM Trans. Graph. 34*, 4 (July 2015), 127:1–127:10. 2

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

[SB16] SCHMIDT R., BROCHU T.: Adaptive mesh booleans. *CoRR abs/1605.01760* (2016). 2

[SBSCO06] SHARF A., BLUMENKRANTS M., SHAMIR A., COHEN-OR D.: Snappaste: an interactive technique for easy mesh composition. *The Visual Computer 22*, 9-11 (2006), 835–844. 2

[SHBSS16] STANKO T., HAHMANN S., BONNEAU G.-P., SAGUIN-SPRYNSKI N.: Surfacing Curve Networks with Normal Control. *Computers and Graphics 60* (Nov. 2016), 1–8. 2

[SJW07] SCHAEFER S., JU T., WARREN J.: Manifold dual contouring. *IEEE Transactions on Visualization and Computer Graphics 13*, 3 (May 2007), 610–619. 2

[SW03] SCHAEFER S., WARREN J.: Dual contouring: "the secret sauce", 2003. 2

[WPM12] WUTTKE S., PERPEET D., MIDDELMANN W.: Quality preserving fusion of 3d triangle meshes. In *Information Fusion (FUSION), 2012 15th International Conference on* (2012), IEEE, pp. 1476–1481. 2

[YZX*04] YU Y., ZHOU K., XU D., SHI X., BAO H., GUO B., SHUM H.-Y.: Mesh editing with poisson-based gradient field manipulation. *ACM Trans. Graph. 23*, 3 (Aug. 2004), 644–651. 2

[ZJC13] ZOU M., JU T., CARR N.: An algorithm for triangulating multiple 3d polygons. In *Proceedings of the Eleventh Eurographics/ACMSIGGRAPH Symposium on Geometry Processing* (Aire-la-Ville, Switzerland, Switzerland, 2013), SGP '13, Eurographics Association, pp. 157–166. 2