Eurographics Symposium on Geometry Processing (2005),
M. Desbrun, H. Pottmann (Editors)

Template-Based Mesh Completion

Vladislav Kraevoy and Alla Sheffer

University of British Columbia, {vlady | sheffa}@cs.ubc.ca

Abstract

Meshes generated by range scanners and other acquisition tools are often incomplete and typically contain multiple connected components with irregular boundaries and complex holes. This paper introduces a robust algorithm for completion of such meshes using a mapping between the incomplete mesh and a template model. The mapping is computed using a novel framework for bijective parameterization of meshes with gaps and holes. We employ this mapping to correctly glue together the components of the input mesh and to close the holes. The template is used to fill in the topological and geometric information missing in the input. The completed models are guaranteed to have the same topology as the template. Furthermore, if no appropriate template exists or if only topologically correct completion is required a standard canonical shape can be used as a template.

As part of our completion method we propose a boundary-mapping technique useful for mesh editing operations such as merging, blending, and detail transfer. We demonstrate that by using this technique we can automatically perform complex editing operations that previously required a large amount of user interaction.

1. Introduction

With the improvement in and the declining costs of scanning technology, modern computer graphics increasingly uses scanned data as a major source for modeling real life objects. Readily available commercial software is used to merge multiple scans. Unfortunately, the resulting merged scans remain incomplete because of occlusions and grazing angle views. The occluded regions can be quite large, for instance, in Figure 1 the scanner failed to capture the interior facing sides of arms and legs. Often, scanners also fail to capture complex model features such as toes or ears (Figure 1). When performing scan completion, the difficulty is to correctly reconstruct the connectivity and the topology as well as to complete the missing geometric details. Even the connectivity reconstruction by itself can be quite challenging. In Figure 1, for instance, each leg should be considered individually, and their respective fronts and backs connected to produce a model with two legs, despite the fact that geometric proximity suggests connecting front to front (and back to back). To correctly repair such models global shape information, such as the one provided by a template, is necessary. Regrettably even when a template is provided, existing completion methods, e.g. [ACP03], lack robustness and are likely to fail on complex models such as this one.

The challenge of mesh completion is not unique to range scanners. Meshes acquired by volumetric techniques (CT, MRI) are by definition closed. However, the acquisition process often generates errors in regions with complex topology. After removing the erroneous surface parts, the meshes need to be completed, raising the same issues. A similar problem also arises in archeological applications, in which models need to be reconstructed from fragments.

Template-based mesh completion can be seen as a special case of the mesh-merging operation, in which the incomplete meshes are merged with completing parts from the template. One of the main difficulties in template-based completion is to map the boundaries of the gaps and holes onto the template in order to correctly define the completing region. This problem also arises when meshes are merged or blended by editing applications. Existing automatic boundary mapping techniques rely on planar parameterization and are thus limited in terms of surface topology and the complexity of the boundaries they can handle. Therefore, users must often manually specify the appropriate boundaries on all the inputs — a tedious and error-prone task when the boundary has a complex shape or multiple loops.

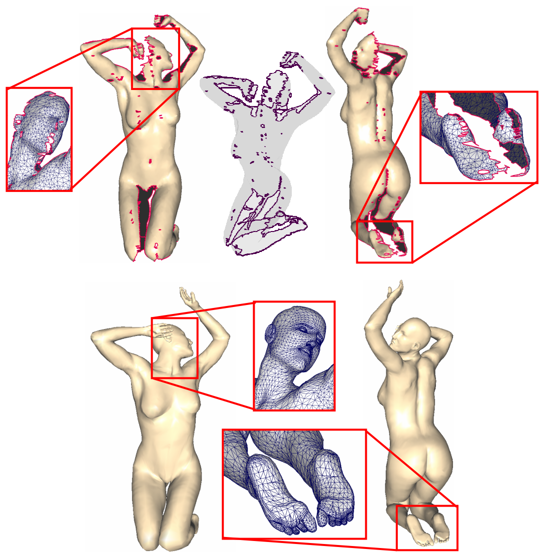

Figure 1: *Template-based mesh completion: (top) incomplete scan input – the semi-transparent grey image shows the multiple complex holes in the input; (bottom) reconstructed model.*

Model completion is not the end-stage of the process, as reconstructed models typically require further processing to meet the requirements of different applications. This processing often includes such tasks as remeshing, texturing, assignment of material properties, and animation. Many of these tasks become simpler if the model is parameterized over a canonical domain. It is particularly useful to parameterize models that belong to the same family of shapes on a common base domain, as this provides a single parameterized space for the entire family of shapes.

Our algorithm facilitates both completion and editing of mesh models by using parameterization tools. It helps the reconstruction of models from incomplete meshes. It also provides tools for mapping boundaries for mesh merging and other mesh-editing operations. As part of the algorithm we compute a low-distortion, bijective parameterization between the completed models and a canonical base domain that may be used during further model processing.

1.1 Previous work

Modern commercial scanners, such as Cyberware [Cyb], are becoming more robust and are capable of merging multiple scans using registration information. However, due to occlusions, the output meshes remain incomplete, containing numerous holes and multiple components. Several methods for closing holes were proposed recently [DMGL02; Lie03; Lev03]. Davis et al. [DMGL02] successfully close holes in large meshes using volumetric techniques, but do not always preserve the topology of the mesh, generating spurious handles. Liepa [Lie03] triangulates the holes using a method with $O(n^3)$ complexity, which can become unpractical for large meshes. Levy [Lev03] uses 2D parameterization to efficiently close holes of any size. The technique requires manual alignment of the components in 2D to close gaps. Since the completion is performed in the plane, the method cannot generate closed models. All three methods close the holes as smoothly as possible, without attempting to reconstruct the missing geometry. Sharf et al. [SACO04], reproduce missing geometry by copying patches from other regions of the same model. However, they cannot generate missing features out of nowhere. Hence their method is inadequate in the common case of scanned meshes missing complex features. None of the above methods uses global shape information. Therefore, they are unlikely to reconstruct correctly the connectivity for complex holes such as the interior side of the legs in Figure 1.

Template-based completion techniques, such as [KHYS02; ACP02; HSC02; ACP03; ASK*05], fill-in the missing information using template geometry. These methods compute a mapping between the incomplete *input* mesh and the template to correctly close the gaps and holes. These techniques were tailored for reconstructing particular families of models. Most methods assume that the templates are geometrically very similar to the input meshes, in terms of shape [KHYS02; ACP02] or pose [HSC02; ACP03] and the gaps and holes in the data are relatively small.

The method of Allen et al. [ACP03] is one of the most robust. It uses user-specified correspondence between a small set of feature vertices, or *markers*, on the input and template meshes to align them with one another. On our request the authors of [ACP03] used their method to map the input mesh in Figure 1 to the corresponding template, given the set of markers we used for completion. The method failed to initialize given this input. The new method of Anguelov et al. [ASK*05] relies on the existence of an underlying skeleton to reconstruct models with major pose variations. As such the method is limited to human or animal models.

Existing parameterization methods focus on two main problems: the planar parameterization of disk-like meshes and the parameterization of closed meshes on canonical base domains (see [FH04] for a review). The standard approach for parameterizing surfaces with holes is to map them to a planar domain. This approach causes significant distortion when the models are far from developable and cannot be used for meshes with genus greater than zero. Schreiner et al. [SPPH04] compute mappings between models of any genus with the same number of simple holes. Their method is not applicable for template-based completion, since our input meshes typically contain many more holes than the templates. None of the methods, reviewed by Floater and Hormann [FH04], addresses parameterization of meshes with multiple components. The challenge in parameterizing such meshes is to maintain the alignment of the components with respect to one another during the parameterization. Sumner and Popovic [SP04] used a variation of the method in [ACP03] to compute mappings between models with significant shape variation. As they noted, this often required specifying a very large number of marker correspondences.

When merging parts of models, most methods require the user to specify the exact boundaries of all the merged regions (e.g., [SCOL*04]). Bierman et al. [BMBZ02] use planar parameterization to automate the mapping of the boundaries from one model to another. This method cannot handle patches with non-disk topology (genus greater than zero or multiple boundary loops). Yu et al. [YZX*04] propose several interactive tools for mapping the boundaries. By using a directional projection tool, they can map boundaries between meshes with different topology. The method fails when the projected boundary self-intersects.

1.2 Overview and contribution

We present an efficient and robust algorithm for template-based mesh completion. The method computes a mapping between the input and template meshes and uses it to facilitate the completion. Like previous template-based techniques, we use a small set of markers to provide a coarse alignment between features on the input and template meshes.

Our method supports both global and local completion. Global completion, used in most of the examples in this paper, maps the entire input mesh onto the template closing all the gaps and holes at once. As such, it takes maximal advantage of the global shape information provided by the template. Alternatively, when the input meshes are very large compared to the size of the holes and gaps, local completion can be used (see Figures 13 and 14). In this case the mapping is restricted to parts of the input and template meshes. Using local completion speeds up the processing and allows mapping between input and template meshes with different genus.

Our method can handle meshes with any number of connected components and boundary loops and any size of gaps and holes. When performing global completion, the reconstructed models are guaranteed to have the same topology as the template. The use of template information provides the global context necessary for establishing correct connectivity for non-planar holes such as the interior sides of the legs and arms (Figure 1) and enables realistic completion of missing geometric data.

Since scanned input meshes often contain a large number of small components, placing a marker on each one would be very tedious. We introduce a mechanism for automatic completion of gaps around marker-less components based purely on geometric adjacency information (Section 5.1, see for example Figure 11).

Templates with a shape similar to the input meshes are not always easily available. We provide two approaches to overcome this difficulty: for many incomplete meshes our method can robustly complete the topology and connectivity using a canonical template and automatically generated markers (Figure 11); alternatively, we can also perform mutual completion from incomplete meshes of similar objects using complimentary information to establish correct cross-hole and cross-gap connectivity and to facilitate geometry completion. In Figure 12 we use the second approach to mutually complete two human scans in different poses.

Our global mapping provides a low-distortion bijective parameterization between the template and the reconstructed model that can be used by subsequent mesh-processing applications. Joining the tasks of mesh completion and parameterization increases the ability to automate the acquisition and processing of large families of models.

The presented algorithm can also be used for mesh-editing applications. Given one mesh with specified boundaries, it is capable of automatically mapping the boundaries onto a second mesh for operations such as merging or blending. It is the first boundary mapping method that can handle both complex boundaries and non-disk topologies. In contrast to previous techniques, it requires no user interaction beyond the specification of marker vertex correspondences.

As part of our algorithm we introduce the first base-mesh parameterization technique that can process surfaces with multiple components and complex holes. It supports positional constraints and can be used to minimize different distortion functions. The resulting parameterizations are globally continuous and bijective for any given input.

The next section provides an overview of our method.

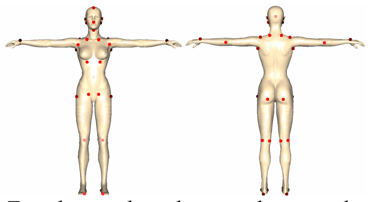

Figure 2: *Template and markers used to complete the female model in Figure 1.*

2. Algorithm

Our algorithm operates on two meshes: input and template. We note that for processing purposes we do not really distinguish between the two. So for instance we can mutually complete two different incomplete meshes, by treating one as an input and the other as a template in turn. For mesh editing we treat the mesh on which the user specified the blending/merging boundary as the input and the second mesh as the template (Color Plates Figure 3). In both cases a set of specified marker vertex correspondences between the input and template meshes is also included (Figure 2). The markers are specified either manually or provided as part of the data [ACP03]. Since a template can be used for

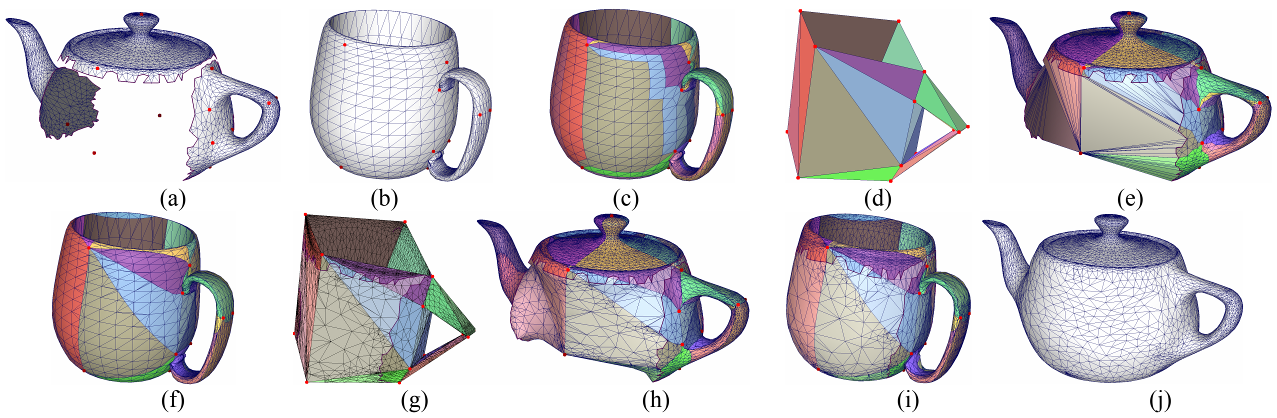

Figure 3: *Algorithm stages. (a,b) Input and template meshes with markers (2 hanging markers on the input mesh are added as described in Section 4.2). (c-e) Segmentation: (c) template, (d) base mesh, (e) input segmentation and virtual triangulation (virtual triangles shown in lighter color). (f-i) Parameterization: (f) template; (g) input mapped to base; (h) closed input mesh; (i) input mesh mapped to template; (j) completed model (using input connectivity and removing Steiner vertices). Note that the shape of the sides and the bottom comes from the mug and hence differs from that of the traditional teapot.*

completing more than one input, it may contain more markers than an individual input (e.g. Figure 3). We found that using those markers (Section 4.2) speeds up, and sometimes improves, the completion. To simplify the subsequent processing, we do not allow vertices on mesh boundaries to be specified as markers.

Our algorithm has four main stages, visualized in Figure 3 on an artificial example:

1. **Pre-processing:** The two meshes are prepared for segmentation and mapping (Section 3).
2. **Segmentation and base-mesh construction:** A base-mesh is constructed and the input and template meshes are consistently segmented into patches using the markers as patch corners. (Section 4, Figure 3 (c,d)).
3. **Base-mesh parameterization:** Both meshes are parameterized on the base (Section 5). At this stage the gaps and holes in each mesh are closed using *virtual* triangles, resulting in a closed mesh with the same topology as the base (Figure 3 (e)). At the end of this stage we have a parameterization between the meshes (Figure 3 (f-i)). This parameterization defines the boundary mapping for mesh-editing operations.
4. **Blending:** For template-based completion, the complete model is constructed (Figure 3 (j)) by blending the template and the input meshes (Section 6).

The stages of the algorithm are described below in detail.

3. Pre-processing

In this stage the algorithm prepares the meshes for mapping by editing mesh components with multiple boundary loops. First, it closes small holes (where the size is based on a user-defined parameter), computing an initial *virtual* triangulation using simple corner cutting. The quality of the initial triangulation is not important as it will be improved by the mapping stage (Section 5.2). Next, the algorithm unites the remaining boundary loops into one. Using the face graph of the mesh, it first computes a shortest-path (Steiner) tree connecting the loops. Then it merges the loops by removing the faces in the Steiner tree from the mesh (Figure 4). The purpose of pre-processing is to avoid ambiguities in gap closure, as it is often difficult to classify which boundary loop on one mesh component corresponds to a boundary loop on another. Generating a single boundary for each component eliminates this ambiguity. The virtual triangulation procedure applied later-on (Section 5) typically reconnects the unstitched triangle paths.

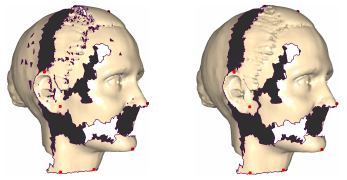
Figure 4: *Results of pre-processing: the small holes are closed and the two large boundary loops on the top-front component are connected into one loop.*

Next, the algorithm segments the pre-processed meshes into patches.

4. Segmentation

A consistent mesh segmentation for closed meshes is typically constructed by incrementally introducing paths between marker vertices [PSS01; SPPH04; KS04]. The main challenge in adapting this approach to our needs is to account for gaps and holes in the input meshes. In particular, since markers may be located on different components, the algorithm must handle paths across gaps. To support paths across gaps or holes, we define three types of *legal paths* on meshes:

- *Cross-gap* paths connect markers on different mesh components. They are formed by pairs of edge paths, each from the marker to the boundary of its respective component. The two boundary vertices are connected by a *virtual* edge. Similar to regular edges, the length of a virtual edge is the Euclidean distance between its end-vertices. Cross-gap paths contain exactly two boundary vertices and are not allowed to contain any boundary edges.
- *Interior* paths are paths between markers within one component that contain only interior edges and vertices.
- *Cross-hole* paths also connect markers within one component but go across boundaries. They are formed by pairs of edge paths, each from a marker to the boundary of the component. The two boundary vertices are connected by a virtual edge. Similar to cross-gap paths, they contain exactly two boundary vertices and are not allowed to contain any boundary edges.

We disallow paths that go along boundaries or cross holes and gaps multiple times, since they can pose difficulties either when segmenting the meshes or later when mapping the patches onto the base mesh.

Our segmentation method first segments the template mesh and derives the base mesh from this segmentation. It then enforces the same segmentation on the input mesh. When performing template-based completion with the same template for reconstructing different input models, the template segmentation and the base mesh can be re-used.

4.1 Base mesh construction

The template mesh is segmented by introducing legal edge paths between marker vertices, one-by-one. At each point, the algorithm selects the shortest path between markers that does not intersect existing paths (Figure 3 (c)). At first, only interior paths are allowed. Once the markers inside each component are triangulated, we allow cross-gap paths to be added. For a template with no gaps this procedure is guaranteed to generate a manifold and orientable segmentation. In all our experiments the method generated valid segmentations for models with multiple components as well, even though in theory it has no orientability guarantees. For models with genus greater than zero, the method may require the user to specify additional markers, if the resulting patches contain handles. Given the segmentation, the algorithm constructs the corresponding base mesh by generating straight edges between the corner vertices of the patches (Figure 3 (d)).

Next, we segment the input mesh enforcing the base mesh connectivity.

4.2 Input mesh segmentation

The input mesh might contain only a subset of the markers, corresponding to base-mesh vertices (Section 2). Therefore, the first stage of the segmentation algorithm introduces the missing markers into the mesh as *hanging* vertices. Each such vertex defines a separate connected component. To position each hanging vertex in 3D, the method measures the distances between the corresponding base-mesh vertex and its neighbors in the base mesh. It positions the hanging marker in 3D at the same distance from the neighboring markers using LLE [RS00] (see Figure 3 (a)). In practice, the exact position has little influence on the result, as it is ignored in the blending stage (Section 6).

To generate the segmentation, the algorithm introduces paths between marker vertices one-by-one, based on the base-mesh connectivity. When introducing each path we must make sure that it does not block future paths [PSS01]. Regrettably, in a multi-component setting path blocking cannot be tested as described in [KS04; SPPH04]. Therefore we specify a particular order of adding paths in which blocking cannot occur, extending the Steiner tree approach for closed meshes [PSS01] to meshes with multiple components and holes. Our method introduces legal paths that correspond to base mesh edges in a specified order.

First, for each connected component, the method constructs a Steiner tree of the markers in this component using only interior paths. Next, it proceeds to generate a Steiner tree of the connected components, by adding cross-gap paths between markers in different components. At the end of this stage, all the markers are connected into a single tree, while the paths constructed so far do not block any future paths between markers inside any component or between markers in different components. At the same time, from now on, new paths will not cause blocking, as all the markers are connected. Finally, the algorithm completes the segmentation by adding paths of all the three legal types (cross-gap, interior, and cross-hole), each time adding the shortest path that corresponds to a base-mesh edge.

The paths are traced on the face graphs of the separate connected components and then converted to edge paths, adding extra, Steiner, vertices as necessary. Using the same arguments as in [PSS01] it is easy to show that the construction algorithm will always result in a segmentation consistent with the base mesh for models of genus zero. While in theory the method can fail for models of higher genus, in practice it performs well (Figures 3, and 14). Occasionally, the segmentation creates so-called *swirls* [PSS01], which are resolved as described in [SPPH04].

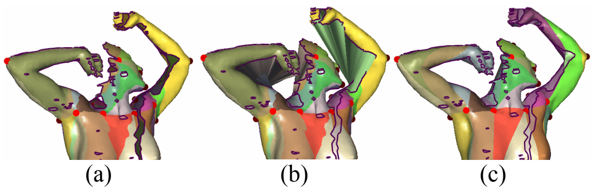
Figure 5: *Segmentation and virtual triangulation: (a) initial segmentation (Section 4.2), (b) initial virtual triangulation (Section 5.1), (c) final segmentation and triangulation (Section 5.2).*

The generated paths segment the mesh components that contain markers into patches, such that the real and virtual edges in each patch form a connected planar graph. The input and template meshes are now parameterized onto the base mesh, using the computed segmentation. Figure 5 (a) shows the initial segmentation of the female model from Figure 1.

5. Base-Mesh Parameterization

This section presents a framework for mapping meshes with multiple components and holes onto a base mesh. The framework enforces the mapping of markers on the input and template meshes to the vertices of the base mesh. To correctly parameterize gaps and holes we use virtual triangulation. In the past, virtual triangulation was used as pre-processing for planar parameterization, to enable parameterization of meshes with holes [SdS00] and to reduce parameterization distortion [LKL02]. Our work further develops this idea using iterative Delaunay triangulation to improve the quality of parameterization and facilitate automatic gap and hole closure.

The mapping is computed in two steps. An initial one-to-one embedding from the mesh to the base domain is computed and the gaps and holes are triangulated, enforcing base-mesh topology (Figure 5 (b)). The mapping and the virtual triangulation are then improved using a combined smoothing and re-triangulation procedure (Figure 5 (c)).

When performing local completion, we simply skip some of the patches during parameterization. In the feline, Figure 14, the patch containing the bulk of the model was specified by the user to be ignored during the parameterization. Clearly, this significantly sped up the completion procedure (Table 1).

5.1 Initial Embedding and Triangulation

The algorithm parameterizes the input mesh onto the base mesh, by mapping each patch onto the corresponding base-mesh face.

1. First, the method applies uniform embedding [Tut60] to parameterize each patch onto the base triangle. Note that a region of a gap or a hole, that is associated with a patch, is by construction bounded by real and virtual edges, thus forming a polygonal face when viewing the patch as a connected planar graph. Uniform embedding is one-to-one for all connected planar graphs. Hence, it is guaranteed to generate a bijective mapping

from the patch to the base triangle. The embedding maps all the gap and hole regions to convex polygons.

2. Next, the algorithm uses the mapping to triangulate the gaps and holes in the patch. The triangulation is performed in 2D using the boundary vertex positions on the base-mesh triangle. The algorithm constructs a Delaunay triangulation of the hole regions using the Triangle software package [She96].

As mentioned in Section 1.2 our algorithm can handle input meshes containing connected components with no markers. To include those in the parameterization, the method classifies each of them as belonging to the nearest patch that contains virtual triangles and embeds them into the triangulation as follows.

For each component, it locates a virtual triangle nearest to it. The component is then embedded into the interior of the triangle using uniform embedding, where the interior domain is formed by subdividing the triangle twice using mid-edges. The surrounding virtual triangulation is updated, adding the boundary vertices of the embedded component to the boundaries of the appropriate hole (Figure 6 (a)). In practice we found that the mapping gives better results if the insertion is performed after a few iterations of the improvement procedure (Section 5.2).

After the triangulation, we obtain an initial bijective base-mesh parameterization. The mesh generated by adding the Delaunay triangles to the input mesh has the same topology as the base mesh. However, the distortion of the initial parameterization depends on the shape of the patches and so can be quite high. In addition, the initial virtual triangulation may connect parts of the input that in reality are very far from each other (Figure 5 (b)). To reduce the distortion and improve the triangulation, a global improvement procedure is applied.

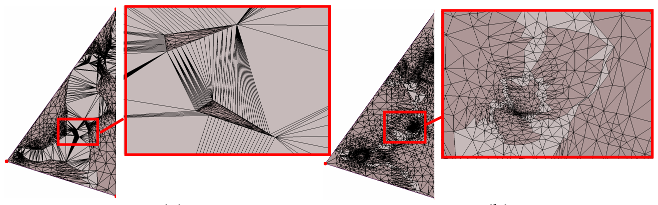
(a) (b)

Figure 6: *(a) Embedding two components with no markers. (b) The components after parameterization improvement.*

5.2 Parameterization Improvement

The goal of the improvement procedure is to generate a globally continuous, low-distortion parameterization of the input meshes over the base-mesh domain. As pointed out by Khodakovsky et al. [KLS03], to achieve this we need a mechanism for relocating the mapped mesh vertices from one base-mesh triangle to another. Existing global parameterization methods [GVSS00; KLS03; THCM04; KS04] operate on closed meshes. The methods map base mesh faces to canonical equilateral base domains in 2D to facilitate the parameterization.

We introduce a technique for bijective base-mesh smoothing that similarly to [GVSS00] uses overlapping quad base-domains. However, instead of using canonical domains as Guskov and others do, we derive the domain geometry from the base mesh, significantly reducing the mapping distortion (Figure 9).

In our setting, each base-domain corresponds to a pair of adjacent faces in the base mesh and is constructed by simply unfolding the two faces in the plane (Figure 7). Since we require a convex domain, it is necessary to scale the triangles, as shown in Figure 8, in the infrequent case when the unfolded quad has a concave corner.

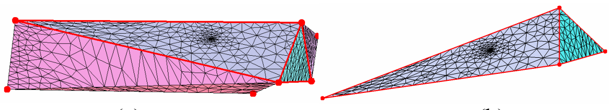
(a) (b)

Figure 7: *(a) Two adjacent base-mesh faces, highlighted in red (with mapped mesh) and (b) their mapping (unfolding) into a planar quadrilateral domain.*

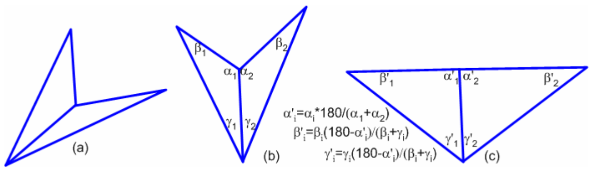
(a)
β_1
β_2
α_1
α_2
γ_1
γ_2
(b)
$\alpha'_i=\alpha_i*180/(\alpha_1+\alpha_2)$
$\beta'_i=\beta_i(180-\alpha'_i)/(\beta_i+\gamma_i)$
$\gamma'_i=\gamma_i(180-\alpha'_i)/(\beta_i+\gamma_i)$
β'_1
α'_1
α'_2
β'_2
γ'_1
γ'_2
(c)

Figure 8: *Unfolding a non-convex quadrilateral: (a) base-mesh faces in 3D; (b) non-convex unfolding; (c) convex unfolding.*

5.2.1 Base-Domain Parameterization and Remeshing

The method improves the virtual triangulation by combining the smoothing with a retriangulation procedure. It iterates over all the pairs of adjacent faces in the base mesh, applying the following steps:

1. **Parameterization:** For each pair of faces, Step 1 constructs the quadrilateral domain and parameterizes the corresponding mesh patches in this domain. First, it maps the exterior bounding paths of the two patches to the edges of the quadrilateral using length-wise parameterization. To parameterize the interior vertices, it uses the parameterization method proposed by Yoshizawa et al. [YBS04]. Thus the algorithm first applies the parameterization improvement framework with the mean-value weights [Flo03] until it converges. It then computes the stretch of the generated parameterization and re-parameterizes the mesh using mean-value weights scaled by a stretch factor. For both types of weights the parameterization on the planar domain is computed by solving a linear system using the conjugate gradient method. This makes our method more efficient than methods such as [KS04; THCM04] in which the mappings of the vertices are computed one by one. Similar to Schreiner et al. [SPPH04], we observed that by introducing a stretch-preserving component into the formulation, we improve the alignment of prominent features when mapping similarly shaped input and template meshes to the same base mesh.

2. **Remeshing and projection to 3D:** This step retriangulates the holes and gaps contained in the two parameterized patches using Delaunay triangulation. Extra vertices are added as necessary to improve the triangu-

lation quality [She96]. To compute the 3D positions for the new virtual vertices, we use mean value embedding [Flo03] in 3D with the weights computed in the plane (Figure 3 (h)). The retriangulation improves both the parameterization and the gap and hole closure. In terms of distortion reduction, combining the parameterization with retriangulation has a similar effect to free-boundary parameterization.

3. **Cleanup:** As explained in Step 4, the parameterization mechanism introduces some Steiner vertices into the mesh. These vertices can be later removed if the removal does not cause triangle flips in the parameterization. The cleanup iterates over all the Steiner vertices in the two parameterized patches and removes the ones which are no longer necessary. Empirically, throughout the iterations the number of Steiner vertices remains about 13% of the original model size.

4. **Introducing Steiner vertices:** After the parameterization and retriangulation, the vertices in the two patches may be mapped to a different base-mesh face from the one they were mapped to before. Hence, the boundary between the patches needs to be recomputed. In Step 1, we assumed that each patch is bounded by paths of edges. Therefore, we need to compute a path of edges that maps to the base-mesh edge shared by the pair of faces. The algorithm generates the path by intersecting the parameterized mesh with the planar domain's diagonal that corresponds to the shared base-mesh edge. This obviously results in numerous Steiner vertices added to the mesh. However, we found that the cleanup step is later able to remove the vast majority of these vertices.

The smoothing and triangulation procedure is repeated until both the parameterization and the triangulation no longer change. By repeatedly smoothing pairs of mesh patches, we allow vertices to move freely all across the base. Figure 3 (g, h) shows the result of applying our algorithm to parameterize and triangulate the broken teapot model. The parameterization preserves the shape of the input mesh and in particular the outline of the boundaries. The gap closure is close to optimal, with the correct components connected to each other.

Our framework can be used, as-is, to optimize different parameterization distortion metrics, by replacing the [YBS04] formulation in the parameterization stage with any other appropriate formulation [FH04]. While our framework is simpler and more efficient than *cross-parameterization* [KS04], we found that it computes parameterizations with less distortion (Figure 9). We mapped David's head to the sphere using 4 markers. For fair comparison, similar to [KS04], we used mean-value [Flo03] parameterization. The angular distortion for the mapping obtained using cross-parameterization was 0.16, using our method it was 0.04. Our mapping took 97 seconds to compute, compared to 238 using [KS04].

After the mapping stage is completed, we have a bijective mapping between the base mesh and the meshes formed by adding virtual triangles to the input and template meshes. Hence, by combining any two mappings, we get a bijective mapping between the closed meshes. Figure 3(i) shows the closed teapot mesh mapped to the mug. For mesh editing, this parameterization is used to map the boundaries from one mesh to the other.

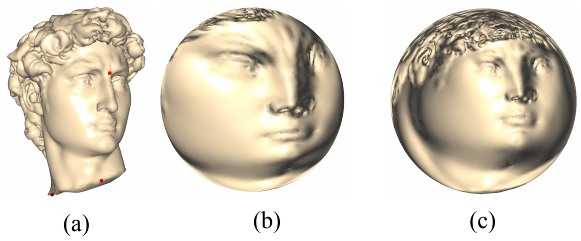

Figure 9: *Parameterization of David's head (a) onto a sphere: (b) using [KS04], (c) using our method.*

6. Blending

For template-based completion we use the mapping computed by the previous stage to construct a complete geometric model by blending the input and template meshes.

To facilitate blending we establish common connectivity for the input meshes using the remeshing approach proposed in [KS04]. We use the connectivity of the closed input mesh (Figure 3) or the template (Figure 1) as the basis for common connectivity. The choice depends on the complexity and resolution of the two meshes. Using the finer, more detailed mesh provides a better approximation. To obtain the same connectivity for all the input geometries, the algorithm simply uses the computed parameterization to map the vertices of one model to the other. Similar to [KS04] the algorithm refines the connectivity if it fails to capture all the details on one of the models.

Since the input and template models often have very different shape, we cannot blend the 3D coordinates directly. To provide intuitive results we blend local shape descriptors instead. In our examples we used the blending scheme of Sheffer and Kraevoy [SK04] to generate the completed models. Other methods, such as [SCOL*04; YZX*04] can also be employed.

The blending uses information from the template in areas where input geometry is missing or unreliable, and input geometry where it is available. The level of reliability can be determined from data confidence values, [TL94] when available. In Figure 1 we marked input data on the hands as unreliable, replacing the fists in the input mesh with open palms from the template. In cases of mutual completion, we can have regions where the geometry is not defined in both models. In these cases the geometry is defined by the projection to 3D of the virtual triangles on the closed template (Section 5.2.1, Step 2). This is, for instance, the case for part of the hair in Figure 12.

Note that once we compute the common connectivity for the meshes, this connectivity provides a straightforward parameterization function between the two. Hence, the base mesh is no longer needed. An important consequence of this is that the Steiner vertices are no longer necessary to maintain the bijectivity of the mapping and can safely be removed (Figure 3 (j)).

7. Results and Applications

Throughout this paper we demonstrate several models constructed using our scheme. The statistics for the models are summarized in Table 1. Global completion took from 20 seconds to 8 minutes for models of 10K to 200K triangles. The times were measured on a 3GHz P4. These times are compatible with those we got from the authors of [ACP03]. However, as noted in the introduction our method is significantly more robust, succeeding on models, such as Figure 1, where [ACP03] fails. Our parameterization algorithm is significantly faster than previous cross-parameterization techniques [KS04;SPPH04].

Model	#Δ input/templ.	#components input/templ.	#markers	time (sec.)
Female	20455/27562	1/1	39	45
Teapot	10739/3450	3/1	16	18
2 females	195660/230831	2/2	37	472
Head	40207/39996	3/1	12	123
Teddy	12596/5120	11/1	4	102
David	35373/5120	1/1	6	140
Feline	13555/1536	1/1	20	33
Hands	45634/5120	2/1	14	150
Escher	1760/39996	1/1	13	25

Table 1: *Model statistics. For the David and feline models the input size is the size of the local completed region.*

The time complexity of the method is O($n\log n$), where n is the number of vertices. The segmentation requires O($kn\log n$) time for tracing the paths, where k is the number of markers – tracing a single path takes O($n\log n$). Since the number of markers does not grow with the model size, it can be viewed as a constant. The smoothing is performed a bounded number of iterations (we set the bound to 10 in all our examples). In each iteration solving the linear systems takes linear time (conjugate gradient) and the expected run-time of Delaunay remeshing is O($n\log n$).

For all of the examples, we required a relatively small number of markers (4 to 39) to establish the mapping and successfully complete the models. For comparison, Allen et al. [ACP03] required 74 markers for template-based completion between scans and templates of humans in the same pose.

Figure 1 demonstrates the results of using our method for completion of large and complex holes. The scanned female model in Figure 1 differs significantly in terms of pose and body proportions from the canonical template that we used [Pos] (Figure 2). Despite these differences the completion algorithm correctly reconstructs the cross-hole connectivity and accurately completes the missing features based on the template.

Figure 3 shows a genus one broken teapot model in which the gaps are larger than the mesh fragments. Despite this,

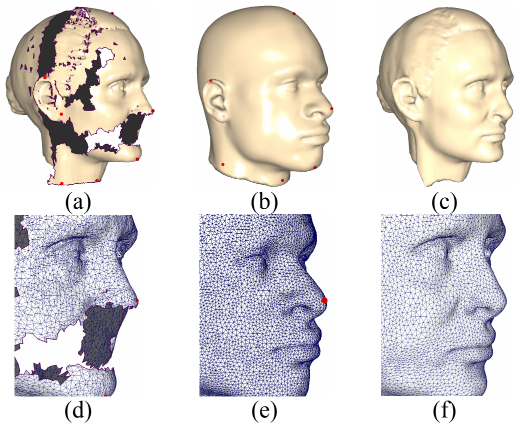

Figure 10: *Completing a head from 3 fragments: (a,d) the input mesh, (b,e) the template, (d,f) the completed model. Features, such as nostrils and mouth, not available in the input mesh were taken from the template.*

the connectivity is reconstructed correctly and the missing geometry is successfully completed. Figure 10 shows the reconstruction of a head from several components. Thanks to the near-perfect alignment achieved by our mapping, the method seamlessly blends partial features from the template and input meshes. In this example the upper half of the nose comes from the input while the lower half comes from the template, yet the two are blended seamlessly and the original geometry of both is clearly preserved. These two examples are typical for archeological applications in which the models need to be completed based on small fragments.

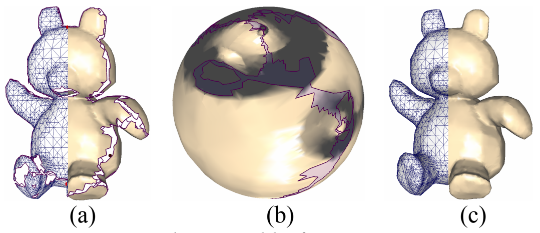

Figure 11: *Completing teddy from 11 components using the sphere as a template and 4 markers: (a) input, (b) teddy mapped to the sphere (normal map), (c) completed model.*

There are many models for which no standard template exists. Such is the case for the teddy bear model generated from merged Z-camera scans (Figure 11). Nevertheless, the model is successfully completed using a simple sphere as a template. The markers are placed automatically at four roughly equidistant (in Euclidean space) points on the model and the sphere. The method robustly parameterizes and completes the model from eleven connected components, most of which have no markers. The spherical parameterization (Figure 11 (b)) is globally continuous and has low distortion.

Figure 12 shows another approach for completing meshes if a template is not readily available. In this example we

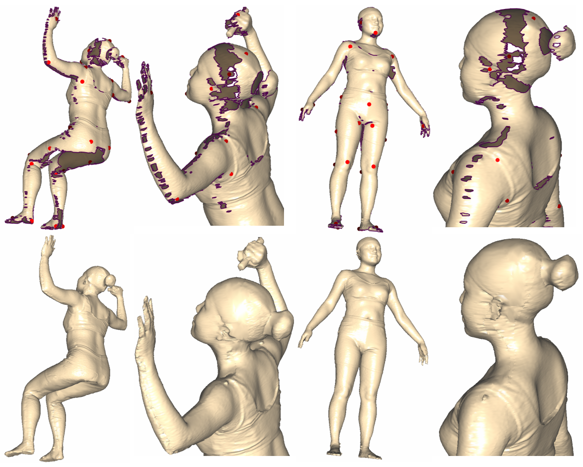
Figure 12: *Mutual completion: (top) two incomplete scans; (bottom) completed models.*

have two incomplete input meshes of females in different poses taken from the CAESAR database [Cae]. To reconstruct the models we compute the mapping between them and then perform blending to complete the sitting model with data from the standing one and vice versa. The algorithm successfully completes not only the medium size holes on the standing model but also the huge hole on the behind of the sitting one. No markers were specified on the hair component of either model. Note that both models have gaps between the hair bun and the head. Therefore the geometry in this region on both models is defined by the virtual triangulation (Section 5.2.1 Step 2).

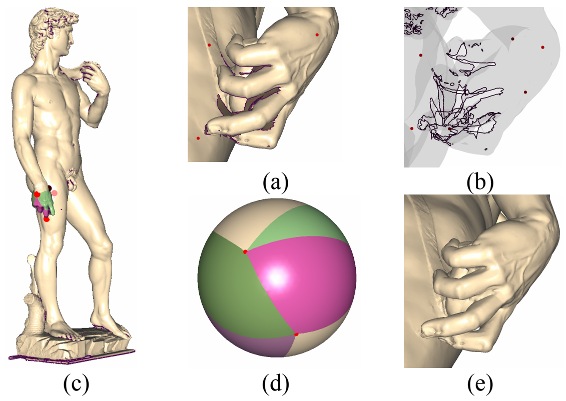

Figure 13: *Local completion of the hand region on the David statue. (a,b) Input mesh as solid and semi- transparent grey images showing the complex hole boundaries on the hand. (c,d) Compatible segmentation of the hand and sphere (The regions ignored by the algorithm are colored in cream, and correspond to two base mesh faces). (e) Completed hand.*

In the examples in Figures 13 and 14 we used local completion to close holes which are relatively small compared to the size of the models. In Figure 13 we show local completion of a bunch of complex holes on the hand of the David model. Due to the high genus of the statue, there are no natural global templates that we can use to complete it. Moreover, since the model contains 1M triangles, computing a global mapping can be quite time-consuming. At the same time, in contrast to the examples in Figures 1 and 12 the holes are relatively small compared to the model size. Hence we chose to perform completion locally. Since there are no complex geometric features missing, we use the standard sphere as a template for the hand region. In this case, the user specifies a sequence of markers to define the boundary of the processed region on the input model (two loops on the top and bottom of David's hand). The template ensures that the model's genus is locally preserved, and no spurious handles are generated. Since the sphere contributes nothing to the model's geometry, we skip the blending stage and use the virtual triangulation of the holes as-is in the completed model.

In Figure 14, we complete the feline model, after breaking-off its tail. This completion is quite tricky as it requires connecting the three boundary loops between themselves forming the necessary handles. Regular hole completion methods are unsuitable for such a task as they typically treat each loop separately. Even manual alignment in this case can be quite challenging. By locally parameterizing the input model onto a figure eight template, we obtain the mapping automatically, achieving a perfect fit.

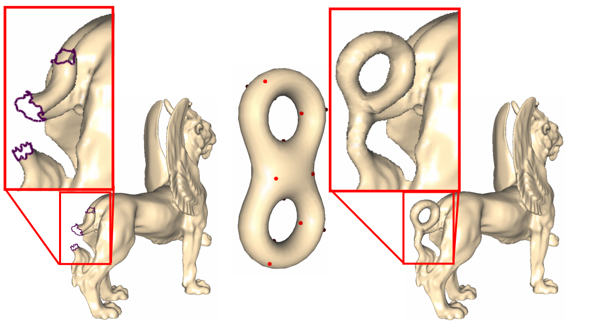
Figure 14: *Local completion operation, incorporating figure eight in place of the feline's tail. The result is practically indistinguishable from the original model.*

Finally, Color Plate Figure 3 demonstrates the use of our method for mesh merging. In the top row two input hand meshes are simultaneously mapped to the template, sphere model. In the bottom row we generate a model inspired by Escher's paintings. This is an example in which the input boundaries cannot be automatically mapped to the template by existing techniques. It is practically impossible to cut a disk-like region of the template head model, containing the mapped spiral, nor is it possible to use axis-aligned projection to map the spiral boundaries.

8. Summary

We presented a robust new method for template-based mesh completion. As demonstrated by the examples, our new method is robust in the face of incomplete input meshes with multiple components and holes. It can successfully complete models on which previous methods are likely to fail. It is very efficient and requires only a small amount of user interaction to specify the marker vertices. Our parameterization mechanism is more generic than previous techniques; it guarantees bijectivity and introduces less distortion (Figure 9). For models with genus greater than zero, the method may require additional markers, if the resulting patches contain handles. These markers can be introduced manually or automatically [SPPH04]. While our algorithm requires fewer markers than most previous techniques, an insufficient number of markers may lead to misalignment of model features.

The computed bijective parameterization between the completed model, the template and the base mesh can be used in a variety of applications. An important application of template-based completion is the construction of parameterized shape spaces [ACP03]. Such spaces are useful for both statistical analysis and the synthesis of new shapes. Our current method requires the user to specify marker vertex correspondences between the input meshes and the template. To parameterize large families of models, such manual selection is impractical. In some cases the markers can be computed from the scan data. In general cases, however, the automation of marker selection requires robust feature-matching techniques. We will explore such matching techniques in our future research.

Acknowledgements

We thank Brett Allen for algorithm comparison. We thank the Stanford Graphics Laboratory for providing the David models and Domi Piturro for the bald head model. This work was supported by NSERC.

References

[ACP02] ALLEN, B., CURLESS, B., POPOVIĆ , Z.: Articulated body deformation from range scan data. In *Proc. of ACM SIGGRAPH 02*, (2002) ACM Press.

[ACP03] ALLEN, B., CURLESS, B., POPOVIĆ, Z.: The space of human body shapes: reconstruction and parameterization from range scans, *ACM Transactions on Graphics (SIGGRAPH 03)*, 22, 3, (2003) 587-594.

[ASK*05] ANGUELOV, D., SRINIVASAN, P., KOLLER, D., THRUN, S., RODGERS, J., DAVIS, J.: SCAPE: Shape Completion and Animation of People, *ACM Transactions on Graphics (SIGGRAPH 05)*, (2005).

[BMBZ02] BIERMANN, H., MARTIN, I., BERNARDINI, F., ZORIN, D.: Cut-and-paste editing of multiresolution surfaces. *ACM Transactions on Graphics*, 21, 3, (2002), 312-321.

[CAE] CAESAR DATABASE, http://www.hec.afrl.af.mil.

[CYB] CYBERWARE INC., http://www.cyberware.com.

[DMGL02] DAVIS, J., MARSCHNER, S. R., GARR, M., LEVOY, M.: Filling holes in complex surfaces using volumetric diffusion. In *Proc. First International Symposium on 3D Data Processing, Visualization, and Transmission* (2002).

[FLO03] FLOATER, M. S.: Mean-value coordinates*.* *Computer Aided Geometric Design*, 20, (2003), 19-27.

[FH04] FLOATER, M. S., HORMANN, K.: Surface parameterization: a tutorial and survey. In *Advances on Multiresolution in Geometric Modelling*, Springer-Verlag, Heidelberg, M. S. F. N. Dodgson and M. Sabin, Eds., (2004).

[GVSS00] GUSKOV, I., VIDIMCE, K., SWELDENS, W., SCHRÖDER, P., Normal meshes, In Proc. *SIGGRAPH* 2000.

[HSC02] HILTON, A., STARCK, J., COLLINS, G.: From 3D shape capture to animated models. *In Proc. First International Symposion on 3D Data Processing, Visualization, and Transmission,* (2002).

[KHYS02] KAHLER, K., HABER, J., YAMAUCHI, H., SEIDEL, H.-P.: Head shop: Generating animated head models with anatomical structure. In *Proc. of ACM SIGGRAPH Symposium on Computer Animation*, (2002).

[KLS03] KHODAKOVSKY, A., LITKE, N., SCHRÖDER, P.: Globally smooth parameterizations with low distortion. *ACM Transactions on Graphics (SIGGRAPH 03)*, 22, 3, (2003), 350-357.

[KS04] KRAEVOY, V., SHEFFER, A.: Cross-parameterization and compatible remeshing of 3D models, *ACM Transactions on Graphics (SIGGRAPH 04),* 23, 3, (2004), 861 – 869.

[LKL02] LEE, Y., KIM, H.S., LEE, S., Mesh parameterization with a virtual boundary, *Computers & Graphics*, 26,5, (2002).

[LEV03] LÉVY, B.: Dual domain extrapolation. *ACM Transactions on Graphics (Siggraph' 03),* 22, 3, (2003).

[LIE03] LIEPA, P.: Filling holes in meshes, In *Proc. of Symposium on Geometry Processing*, (2003),200–205.

[POS] POSER, http://www.curiouslabs.com/

[PSS01] PRAUN, E., SWELDENS, W., SCHRÖDER, P.: Consistent mesh parameterizations. In *Proc. of ACM SIGGRAPH 01*, E. Fiume, Ed.., Computer Graphics Proceedings, Annual Conference Proceedings, (2001), 179-184.

[RS00] ROWEIS, S. SAUL, L.: Nonlinear dimensionality reduction by locally linear embedding. *Science*, 290, 5500, (2000), 2323--2326.

[SPPH04] SCHREINER, J., PRAKASH, A., PRAUN, E., HOPPE, H.: Inter-surface mapping, *ACM Transactions on Graphics (SIGGRAPH 04),* 23, 3, (2004).

[SACO04] SHARF, A., ALEXA, M., COHEN-OR, D.: Context-based surface completion, *ACM Transactions on Graphics (SIGGRAPH 04)*, 23, 3, (2004).

[SDS00] SHEFFER, A., DE STURLER, E.: Surface parameterization for meshing by triangulation flattening. In *Proceedings of the 9th International Meshing Roundtable*, (2000), 161-172.

[SK04] SHEFFER, A., KRAEVOY, V.: Pyramid coordinates for morphing and deformation, In *Proc. of Second International Symposium on 3D Data Processing, Visualization, and Transmission*, (2004), 68-75.

[SHE96] SHEWCHUK, J.R.: Triangle: engineering a 2D quality mesh generator and Delaunay triangulator, In *Proc. Of First Workshop on Applied Computational Geometry*, (1996), 124-133.

[SCOL*04] SORKINE, O., COHEN-OR, D., LIPMAN, Y., ALEXA, M., ROESSL, C., SEIDEL, H.-P.: Laplacian surface editing, In *Proc. of Symposium on Geometry Processing*, (2004).

[SP04] SUMNER, R. W. POPOVIĆ, J.: Deformation transfer for triangle meshes. *ACM Transactions on Graphics (SIGGRAPH 04)* 23, 3, (2004), 397–403.

[THCM04] TARINI, M., HORMANN, K., CIGNONI, P., MONTANI, C.: PolyCube-Maps. *ACM Transactions on Graphics (SIGGRAPH 04),*23, 3, (2004), 850-85

[TL94] TURK, G., LEVOY, M.: Zippered polygon meshes from range images. In *Proc. of ACM SIGGRAPH 94*, *Annual Conference Series*, ACM, 28, (1994), 311–318.

[TUT60] TUTTE, W.: Convex representation of graphs. In *Proc. London Math. Soc.,* 10, (1960).

[YBS04] YOSHIZAWA, S., BELYAEV, A., SEIDEL, H.-P.: A fast and simple stretch-minimizing mesh parameterization. In *Proc. Shape Modeling and Applications*. (2004), 200–208.

[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 Transactions on Graphics (SIGGRAPH 04)*, 23, 3, (2004), 641-648.