Interactive Shape Modeling using a Skeleton-Mesh Co-Representation

J. Andreas Bærentzen*
Technical University of Denmark

Rinat Abdrashitov†
University of Toronto

Karan Singh†
University of Toronto

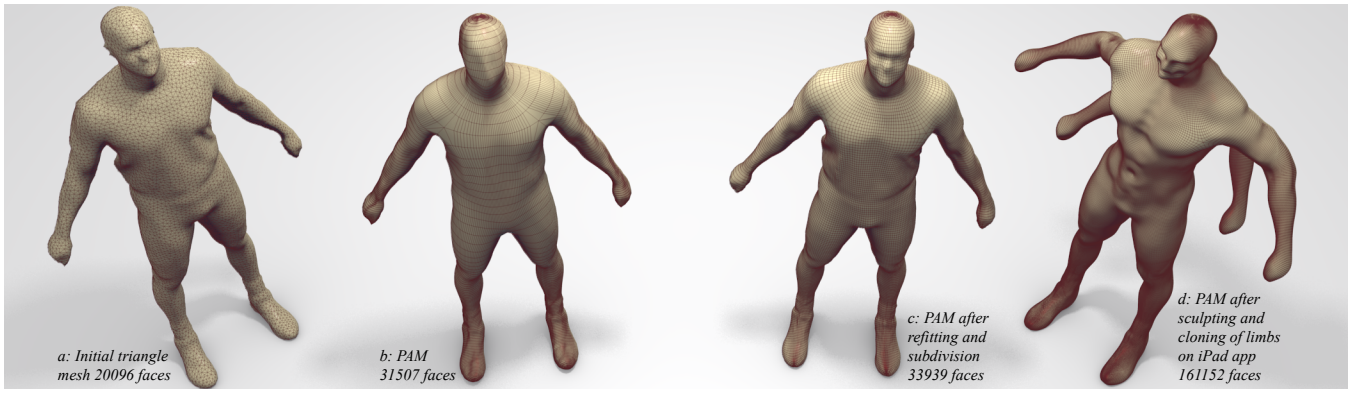

Figure 1: *The input triangle mesh on the left is converted to our polar–annular mesh representation (PAM) (a). It was refitted to the original model and subdivided (b,c) and then edited on an iPad using our PAM based sculpting tool (d).*

Abstract

We introduce the Polar-Annular Mesh representation (PAM). A PAM is a mesh-skeleton co-representation designed for the modeling of 3D organic, articulated shapes. A PAM represents a manifold mesh as a partition of polar (triangle fans) and annular (rings of quads) regions. The skeletal topology of a shape is uniquely embedded in the mesh connectivity of a PAM, enabling both surface and skeletal modeling operations, interchangeably and directly on the mesh itself. We develop an algorithm to convert arbitrary triangle meshes into PAMs as well as techniques to simplify PAMs and a method to convert a PAM to a quad-only mesh. We further present a PAM-based multi-touch sculpting application in order to demonstrate its utility as a shape representation for the interactive modeling of organic, articulated figures as well as for editing and posing of pre-existing models.

CR Categories: I.3.6 [Computer Graphics]: Methodology and Techniques—Graphics data structures and data types, Interaction techniques;

Keywords: shape modeling, polygonal mesh, skeleton

Links: DL PDF WEB

*janba@dtu.dk
†(rinat|karan)@dgp.toronto.edu

1 Introduction

Free-form digital sculpting is steadily replacing traditional CAD workflows for the modeling of organic 3D forms. Animated organic forms adopt anthropomorphic character by articulation and thus have a dominant skeletal structure. In current practice the shape and its skeleton are represented independently and geometrically associated by a process called rigging. Early systems based on a sculpting metaphor recognized the importance of skeletal structures and proposed character construction using implicit functions defined around shape skeletons [Bloomenthal and Wyvill 1997]. Despite numerous advantages, a fundamental limitation of implicit surface models is that the skeleton defines the surface shape and not vice-versa. This surface is usually tesselated into a mesh that can be further manipulated. Adding complex shape details to this mesh will eventually destroy any meaningful connection between the original skeleton and the final 3D shape. Instead, we propose a novel surface representation called a *Polar-Annular Mesh* (PAM), where the surface structure defines the skeleton. As a result the mesh can be refined and detailed using popular sculpting techniques such as found in ZBrush [2010], while preserving skeletal topology.

We are inspired by RigMesh [Borosán et al. 2012], a modeling system that notes the importance of co-modeling, where a set of modeling operations are defined that update both a mesh and a shape skeleton, so they co-exist through the modeling process. We go further to define a co-representation, where the skeletal structure is directly embedded in the connectivity of the PAM mesh. The co-represented skeleton is essential to general shape sculpting: it provides local skeletal reference frames invaluable for coarse, non-linear shape deformations and holds semantic part information useful in selecting meaningful regions of deformation influence.

A PAM represents a surface manifold partitioned into topologically polar (i.e. homeomorphic to a disk) and annular regions. The choice of surface decomposition into poles and annuli is motivated by our focus on organic shapes with articulated limbs, that are readily captured by annuli, capped by poles. This observation is further corroborated by designer surfaced 3D characters that amply use polar-annular patches (Figure 2). A PAM uniquely encodes a skeletal connectivity: every region boundary defines a joint;

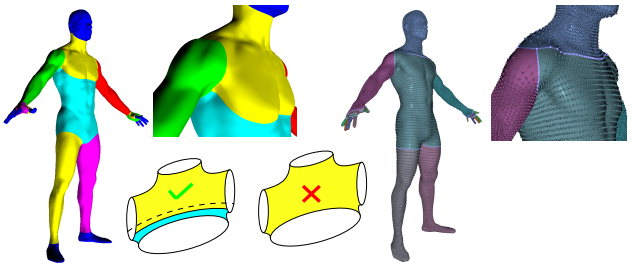

Figure 2: *Designer surfaced articulated models are often PAMs (polar regions shown in blue) at left. The illustration shows that the neck region is an annulus in contrast to a 4-boundary region (not a PAM). Our PAM conversion (right) is structurally similar.*

starting from an arbitrary region boundary, we can enumerate the skeletal joints by recursively visiting the other region boundaries of the regions that are incident on the boundary where we start (Figure 3). The skelatal segments in this construction, may be replaced by chains of joints to better conform to the geometric shape of the PAM. In our implementation, an annular region is represented using connected rings of quads, and a pole by a triangle fan.

The above definition yields a mesh-skeleton co-representation that is compact and operable at varying resolution, making it potentially attractive for interactive 3D modeling on mobile platforms with limited computing power. We demonstrate this potential by addressing two challenges. First, we propose an algorithm that, with minimal user effort, can convert arbitrary 3D polygonal meshes into PAMs with the desired accuracy. This allows PAM based sculpting systems to manipulate existing 3D data as well as interface with general 3D mesh modeling software. We also show how our PAMs may be converted to quad-only meshes by replacing the pole vertices and triangle fans with quads and vertices of lower valence. Second, we develop a novel PAM based mobile (iPad) sculpting application that exploits the affordance of multi-touch for skeletally informed shape modeling. We show that a variety of operations, both those that are skeletally-driven like posing or restructuring limbs, and mesh-driven like smoothing or sculpting fine detail, can be be performed in the same modeling context. We thus validate that PAMs can indeed provide a unified framework for the interactive modeling of articulated characters.

Contribution and Overview Our core technical contribution is the *Polar–Annular Mesh* (PAM) representation. We present a formal definition and analysis of its mathematical properties in Section 3 and techniques for converting general triangle meshes to PAM meshes in Section 4. Section 5 presents auxiliary methods for PAM manipulation, and our multi-touch modeling application is presented in Section 6.

2 Related Work

We broadly classify related work into sculpting interfaces, skeletally driven modeling, Morse theory, and quad-meshing.

Sculpting interfaces: allow users to create and manipulate 3D shapes using the metaphor of traditional sculpting [Pixologic 2010]. Such approaches build up shape detail by smoothly adding or removing material [Galyean and Hughes 1991], by cloning existing 3D geometry [Takayama et al. 2011] and even preserving surface features [Stãnculescu et al. 2013]. Such approaches work well with PAMs (see Section 6) and we are able to adapt the PAM resolution dynamically to capture arbitrarily fine detail. Commercially, the implicit surface primitive based AutoDesk 123DCreature provides

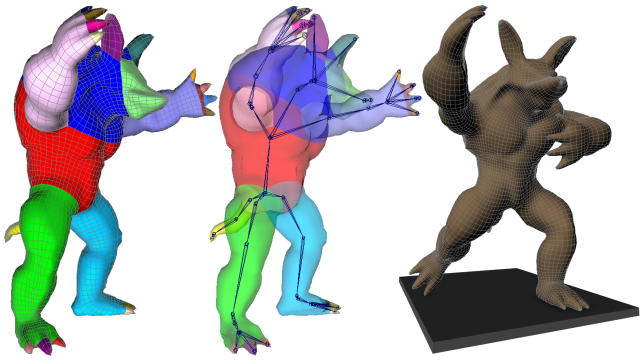

Figure 3: *Armadillo mesh converted to a PAM (left), its simplified skeleton extracted for animation (middle) and posed using the simplified skeleton (right).*

a fluid modeling interface similar to our mobile sculpting application but cannot sculpt arbitrary detail while preserving a meaningful skeleton, and does not exploit the affordances of multi-touch [Sun et al. 2013], specific to articulated shapes. Several sketch and sculpt interfaces like Teddy [Igarashi et al. 2007] infer transient skeletons in 2D/3D for shape modeling operations, suggesting the utility of a persistent mesh-skeleton co-representation.

Skeletally driven modeling: focuses on the creation of articulated figures where the surface shape is dominantly defined by a skeletal structure. A large body of work attempts to construct shape as a combination of a number of skeletal implicit surface primitives, for example [Bloomenthal and Wyvill 1997; Zanni et al. 2013]. These approaches are successful at exploring skeletally defined blobby shapes, and typically hand over a base 3D mesh to mesh sculpting techniques to add surface detail. PAMs support similar skeletal operations and mesh sculpting within the same framework. Character animation has motivated much research on the related problem of rigging or skinning, for example [Vaillant et al. 2013], where a mesh bound to a skeleton deforms to conform to animator controlled skeletal pose. Independent mesh and skeleton inputs can also be registered together and rigged automatically [Baran and Popović 2007]. We are able to pose characters by directly interacting with the skeleton inherent in a PAM and also show how a PAM skeleton can be simplified to an animator's skeleton in Section 5. The presence of an inherent skeleton can provide a user more and persistent shape control, than curve-based free-form deformations [Singh and Fiume 1998], fitted 3D primitives [Andrews et al. 2012], or approaches based on variational shape deformation [Sorkine and Alexa 2007]. Rigmesh [2012] supports the co-modeling of a general 3D mesh and associated skeleton. Such an approach however, is limited by its vocabulary of modeling operations, that must describe how to modify both the mesh and skeleton. Cut and join operations in Rigmesh for example, combine meshes with Booleans and merge skeletons separately. Skeletal approaches to creating base meshes have also been proposed [Srinivasan et al. 2005; Ji et al. 2010], where skeletal segments are converted to cylindrical mesh geometry and stitched across joints. Bærentzen et al. [2012] solve this problem by creating polyhedra for joints of degree greater than 2, that are refined and connected to cylinders. They turn skeletons to base meshes that are co-incidentally polar-annular, but fail to recognize and develop the possibility that PAMs can form the basis of a mesh-skeleton co-representation. A recent trend in character modeling uses part and limb segmentations to construct models by cut and paste of existing parts [Schmidt and Singh 2010; Chaudhuri et al. 2013]. These modeling workflows are complementary to our work, where polar and annular regions created interactively or by our algorithm, are well aligned with semantic part information. Fi-

nally, other authors have investigated how to use, or simply extract, the intrinsic structural information of a triangle mesh [Thiery et al. 2013; Isenburg et al. 2001].

Morse theory: is a tool for analyzing the topology of surfaces through the critical points of a function f defined on the surface [Milnor 1963]. Given a Morse function, f, the Reeb graph is a graph where each connected component of each level set of f maps to a single point on the graph. The connectivity of the graph is inherited from the manifold. Thus, bifurcations occur where level sets change topology. For a survey on Reeb graphs on triangle meshes, the reader is referred to [Biasotti et al. 2008]. Our triangle mesh to PAM conversion can be viewed as an embedding of the Reeb graph connectivity, or to be precise the connectivity of an extended Reeb graph [2000] in the mesh itself.

Quad-meshing: methods attempt to define the flow-lines of shape [Bessmeltsev et al. 2012] and are often based on establishing a vector field guided, periodic parametrization on a triangle mesh. The quad faces are then formed by tracing isoparameter curves in this parametrization, sometimes using an interactive workflow [Krishnamurthy and Levoy 1996; Takayama et al. 2013]. Bommes et al. [2011] proposed a method which removes helical configurations from quad meshes, thereby improving the global structure. However, our work best relates to Dong et al. [2005], who establish a harmonic function on a triangle mesh and independently trace curves in the gradient and its perpendicular direction. Tracing stops when the curve density is too high resulting in t-junctions (optionally triangulated), and no high-level mesh structure. It is unclear if and how their algorithm could be adapted to create PAMs. Our approach grows PAM regions separated by level set curves. While the algorithm we propose can be used to extract a skeleton or quad-mesh from a general mesh, our focus is to define both (prior art does one or the other), as a PAM, within a shape modeling context. For a recent survey of work on quad meshing, see Bommes et al. [2013].

3 Polar Annular Meshes

Definition 1 (PAM) *A polar–annular mesh is a polygonal mesh that consists exclusively of triangles and/or quadrilaterals such that (i) each quadrilateral belongs to a single ring of quadrilaterals denoted an **annular region**, (ii) each triangle belongs to a single fan of triangles denoted a **polar region**, and (iii) the faces incident on a vertex form a region homeomorphic to a disc, and vertices may not be incident on edges (i.e. t-junctions are not allowed).*

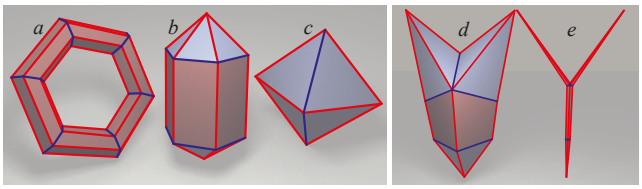

Figure 4: ***Left****: Three PAMs where rib edges are shown blue and spine edges are red.* ***Right****: This PAM contains a junction where three branches meet. Note how the rib edges (blue) form a complex loop at the junction in (d). Contracting the rib edges (almost to the barycenter), we obtain a shape close to the line skeleton (e).*

To facilitate discussion, we need some terminology: A **rib edge** is an edge which bounds a polar or an annular region. A **spine edge** is an edge in the generator direction of a polar or annular region. Finally, a **pole** is the apex of a polar region.

It is easy to demonstrate that PAMs exist. The simplest example with both an annular and a polar region is a ring of quadrilaterals capped with triangle fans at either end (Figure 4b). It is also easy to show that some meshes are ambiguous. For instance, any torus meshed using quads (Figure 4a) can be construed as a PAM in two different ways: the rib edges can form the loops corresponding to the minor radius and the spine edges form the loop corresponding to the major radius (as shown) or vice versa. An octahedron can be construed as a PAM in three different ways due to symmetry. In Figure 4c one case is shown where the poles are the leftmost and rightmost vertex. However, the poles could also be the top-bottom or the front-back pair.

A PAM is constructed by gluing regions together. Since t-junctions are not allowed according to the definition, we need to identify the rib edges of the regions being stitched. However, branching is possible as shown in Figure 4d. Observe that a vertex on the boundary of either a polar region or an annular region is incident on two rib edges and one spine edge from that same region. Once we stitch N regions, we identify pairs of rib edges. Thus, the valency of a non-pole vertex is always $3N - N = 2N$. Moreover, the edges incident on a non-pole are always alternately rib and spine edges. Only spine edges are incident on a pole. Using these two properties, we can label all edges using a graph traversal starting from a single labeled edge. We also observe that since the edge shared by a quad and a triangle must be a rib, the possibility for ambiguity does not arise when a PAM contains both polar and annular regions.

We obtain a *line skeleton* from a PAM by contracting all connected rib edges to a single point (the barycenter of their vertices). Thus, a skeletal node is the set of vertices $\mathcal{V}$ belonging to a connected (possibly complex) loop of rib edges. Two skeletal nodes a and b are connected if there is a spine edge connecting a pair of vertices $v_a \in \mathcal{V}_a$ and $v_b \in \mathcal{V}_b$. It follows that poles correspond to nodes where only the pole belongs to the set of vertices associated with the node. Figure 4e shows a (nearly full) contraction of the PAM in Figure 4d to its line skeleton.

3.1 Primitive Operations

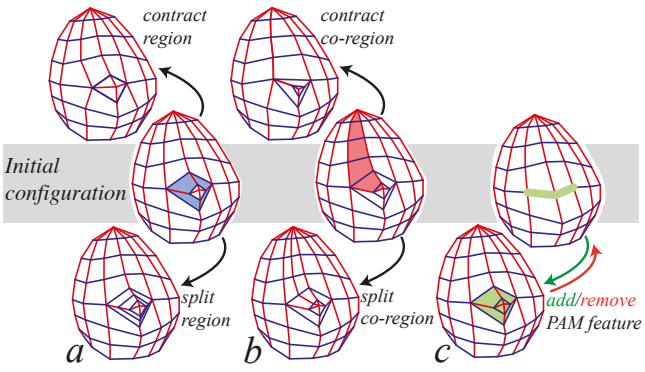

Figure 5: *Primitive PAM operations: the contraction and splitting of a region (light blue) (a), the contraction and splitting of a co-region (light red) (b), PAM features added and removed (c). The ribs being cut to add a feature in (c) are shown using fat light green lines. The added faces are also light green.*

Refinement and simplification. We can refine a PAM by inserting a vertex on all spine edges belonging to a given region (polar or annular). If we then connect these vertices by splitting the faces of the region, we have inserted a new edge loop and thereby created a new annular region as shown in Figure 5a (middle and bottom). A region is a contiguous set of faces bounded by rib loops. We

can also select a contiguous set of faces bounded by spine edges. These face sets do not form regions but end in triangular faces incident on a pole as shown in Figure 5b (middle). We will denote spine-bounded face sets *co-regions*. We can split the rib edges and then the faces of a co-region thereby introducing a new co-region as shown in Figure 5b (bottom).

These two operations can be inverted by removing the introduced edge loops (or sequences). In practice, we prefer to perform contraction of regions since the averaging introduces smoothing as a part of simplification. Thus, we remove a region or co-region by contracting all its spine or rib edges, respectively. See Figure 5a-b (middle and top). In Section 5, we discuss how both refinement and simplification can be used in practice.

Adding and removing PAM features. The refinement and simplification operations described above do not change the skeletal structure. This, however can be done by adding and removing PAM regions. To do so, we need to cut the PAM in order to open a loop of rib edges, creating a boundary in the model. This can be done in two ways: we can create an incision by splitting a sequence of rib edges as shown in Figure 5c (middle). In the figure, we cut along two edges, creating a boundary loop consisting of four rib ribs edges. More generally, cutting along n edges creates a loop of $2n$ edges. We can add a PAM by identifying the boundary rib edges with a loop of boundary rib edges from another model that has been cut open. In the example, we add a PAM feature consisting of an annular and a polar region, thereby adding a small branch as seen in Figure 5c (middle and bottom). Of course, we can also cut open an entire rib loop, dividing the model in two. We would then typically discard the smaller of the two pieces (often just a single polar region) and add a new PAM feature to the larger. This is the primitive operation used e.g. for branch extension (cf. Section 6).

Note that when we *open* a PAM by creating a boundary loop of rib edges, it is technically not a PAM before we create a closed model by adding a new PAM feature. Adding a feature requires that the number of boundary rib edges in the cut is the same as the number of boundary rib edges in the part that we attach. Since cutting along n edges produces a loop of $2n$ rib edges, we may also need to refine or coarsen the PAM feature that we attach if it has an uneven number of edges in the boundary rib loop.

4 Converting Triangle Meshes to PAMs

It is desirable to be able to convert existing triangle meshes to PAMs so that they can be used for further modeling. Our approach is based on the observation that the PAM regions are simply closed loops of polygons that cover the surface. If we embed a system of non-intersecting, closed curves in a manifold, this system can be seen as being dual to the structure of a PAM: each closed curve will, in the end, be transformed into a polar or an annular region. Given a function f on a manifold, we can obtain such a system of non-intersecting closed curves as level set curves of f – assuming that we choose level set curves that do not contain critical points (where ∇f=0). On a triangle mesh, the critical points of a Morse function occur only at vertices [Banchoff 1970]. Thus, by avoiding vertices, it is possible to ensure simple curves. Of course, we must also require that f is designed such that it respects the geometric structure of the shape: the gradient field should flow along important features and the level set curves flow around these features.

Overview: Our initial step is to embed a system of level set curves in the input triangle mesh $\mathcal{T}$. This step is similar to the algorithm by Attene et al. [2003] for computing the extended Reeb graph (ERG) [Biasotti et al. 2000] by slicing the mesh along level sets. Attene et al. are interested in the type and connectivity of critical points, which can be found by analysis of the boundary curves of the region bounded by a number of slices. Our algorithm on the other hand proceeds by removing the vertices which lie between slices. We then use a marching front approach to change the valencies of all vertices to either four (for loops corresponding to annular regions) or three (loops corresponding to polar regions). When all vertices are valence three or four, we can obtain a PAM simply by computing the dual mesh. In the following, we describe each step in detail.

Embedding Level Set Curves. Initially, the user specifies into how many levels the shape should be divided. For level, c, we visit all edges in $\mathcal{T}$. An edge is cut by introducing a vertex if $a < c \leq b$ where c is the level value and $a < b$ are the values at the end point. The half open interval $]a, b]$ ensures that, if a vertex v lies precisely on the level set, i.e. $f[v] = c$, we only cut edges that connect to vertices with smaller values of f. Effectively, this is a perturbation of the value at one end point which ensures that no vertex of $\mathcal{T}$ lies on the level set curve. Since the critical points, saddle points and extrema, are at vertices, the LS curve does not contain critical points and is, therefore, a simple (or Jordan) curve (see Figure 6).

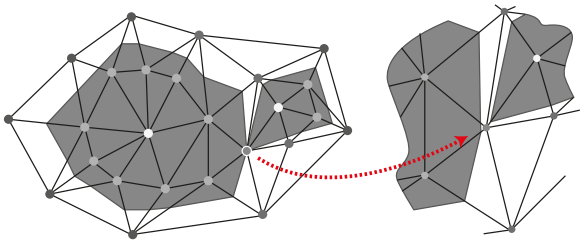

Figure 6: *The dark region above is inside the level set curve corresponding to value c. The close up illustrates that even when a vertex has value $f[v] = c$, the LS curves do not contain it.*

Removing old vertices. We remove all vertices that do not lie on the level set curves. When removing a vertex, we simply merge all incident faces and triangulate the resulting polygon by greedily connecting the closest vertices of the polygon which are not connected by an edge and do not lie on the *same* connected component of a level set curve. Vertices that lie on different components of the same LS curve may be connected, however, and these connections represent saddle regions. Removing vertices from a region bounded by only one LS curve (i.e. a region containing an extremum) leads to a situation where we cannot triangulate the resulting polygon, since all vertices belong to the same LS curve. This polygon is then left untriangulated.

The next step is to smooth the individual LS curves. This is done by parametrizing the curves and then smoothing in the parameter domain to avoid shrinkage. Finally, we optimize the mesh by edge flipping to minimize the edge length. The result of this step is a polygonal mesh where all vertices belong to an LS curve and are evenly distributed along these.

Separating Vertices and Collapsing Edges. In order to arrive at a mesh which is the dual of a PAM, we now visit all level set curves starting from the positive and negative valued curves adjacent to the 0-level curve (a choice that is obviously arbitrary). For each level set curve, we first separate the vertices. This is done by creating a duplicate curve directly under the level set curve and cutting all intersecting edges and connecting the resulting vertices

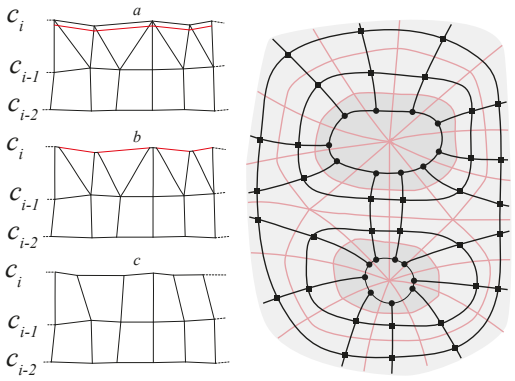

Figure 7: ***Left****: visiting the curve at level c_i (a), we initially separate the vertices with more than one edge connecting to c_{i-1}. This leads to (b) . We collapse all edges on c_i that form triangles with a vertex on c_{i-1}, resulting in (c).* ***Right****: a given level set curve (black) connects to the curves immediately above and below it (in terms of the level set value). Thus, every vertex on an LS curve has valence four unless the LS curve encloses an extremum of the function f since in that case there is no curve above (or below if it is a minimum) and every vertex has valence three.*

(see Figure 7a-b) while removing the old level set curve. We then collapse all edges which lie on the c_i level curve and form a triangle with the curve on the c_{i-1} level curve. This produces the result in Figure 7c. We again improve the distribution of vertices on the level set curve.

As we expand the front, visited level set curves contain vertices only of valence four as shown for the cup model in Figure 8b. There are two exceptions to this statement. When we reach a level set curve enclosing an extremum, there are no curves inside, so its vertices only connect to previous LS curves. This requires no special handling. The other exception is when a curve of level c_i consists of two connected components. In this case, there may be an edge from both of the components to a curve of level c_{i-1} (and likewise for $-c_i$ and $-c_{i-1}$). These cases are handled in a post process. We remove edges from remaining vertices with valence > 4 until they do have valence four. In some cases, the removed edges connect to vertices which are thereby reduced to valence three. These vertices are also removed along with the incident edge that connects to another level set which causes a new valence three vertex. Thus, the procedure becomes recursive.

Obtaining a Polar Annular Mesh. The described algorithm produces a mesh where all vertices belong to edge loops. For a single edge loop, all vertices either have valence four or valence three. In the latter case, the loop forms a polygon. Both cases are shown in Figure 7 (right). Taking the dual results in a mesh which consists exclusively of quadrilaterals or triangles and these form either rings (of quads) or triangle fans. Consequently, we arrive at a mesh which fulfills the requirements in Definition 1 and is, therefore, a Polar Annular Mesh. The final result of this conversion process for a cup model is shown in Figure 8c.

Scalar Field Generation The function f which is used for triangle mesh to PAM conversion could be designed a number of ways. However, an obvious and effective method is to generate a harmonic function from user specified vertex constraints. We allow the user to interactively select a set of vertices and set a scalar boundary value at each. The program then solves $\Delta f = 0$, where Δf is

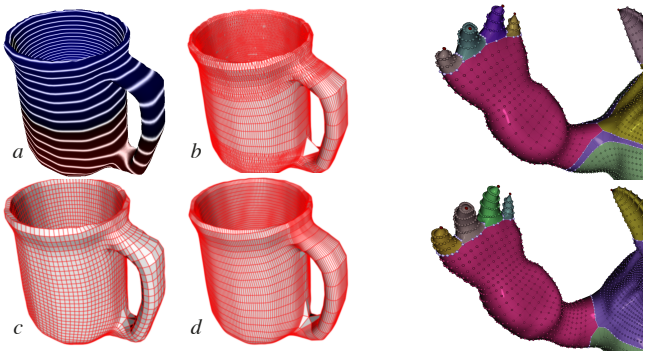

Figure 8: ***Left****: The process of triangle mesh to PAM conversion. The cup is shown with its harmonic function f (a), halfway through the marching front conversion (b), as a converted model (c), and finally after optimization (d).* ***Right****: A segmented PAM (top) and a segmented PAM where an additional ring of constraints has been used to better define the shoulder region (bottom).*

the discrete Laplace operator. Like Ni et al. [2004], we use mean value weights [Floater 2003] and not cotangent weights [Pinkall and Polthier 1993] since the latter can become negative. The cup in Figure 8 (left) is created from just two constraints (on the bottom of the cup inside and out) a positive and a negative constraint. In general, setting these constraints is simple: they are placed on the tips of protrusions. For objects without a clear branching structure such as the human head in Figure 12 it is generally important to obey symmetry. Constraining all vertices in an edge loop to the same (non-extremal) value can be used to guide PAM generation near salient features, e.g. the shoulder of the Armadillo model as shown in Figure 8 (right) where the mesh has been segmented: the color of a given face indicates to which skeletal branch it belongs.

5 Auxiliary Methods

Clearly, any practical use of Polar–Annular Meshes for content production entails the need for more tools. In the following, we present the most important.

Refitting smooths the geometry of the PAM in the manifold of the input triangle mesh $\mathcal{T}$ with the dual objective of improving the vertex distribution while refitting the PAM geometry to $\mathcal{T}$ (as shown in Figure 9 (top)). We initially project a PAM vertex, v onto $\mathcal{T}$ by finding its closest face and the barycentric coordinates of v in this face. Subsequently, for each PAM vertex, v we create a local uv map using the geodesic polar coordinates method of [Melvær and Reimers 2012] and compute the positions of the neighbors of v. We smooth the position of v in uv coordinates and update its face id and barycentric coordinates to reflect the new position. When a sufficient number of smoothing iterations have been performed, we convert the vertex positions back to 3D. A similar smoothing operation is also discussed in [Melvær and Reimers 2012]. The main difference is that we do not convert vertex positions back into 3D between iterations. This *geodesic smoothing* is very effective at redistributing the vertices without introducing shrinkage. In conjunction with PAM simplification (see the following) it can be used to optimize PAMs to fit the geometry while achieving the desired complexity. This procedure was used to create Figure 8d. As an implementation detail, it is worth noting that geodesic smoothing can cause shrinkage at very high curvature areas since it may simply pull the vertices away from these regions. To avoid that, we keep the poles (where high curvature is often concentrated) fixed during smoothing.

Subdivision. Firstly, iteratively adding rib and spine edges is clearly possible. This can be used to refine the mesh. However, polar subdivision and in particular the C2PS scheme introduced by Myles and Peters [2009] has the nice property that while it works like Catmull-Clark [Catmull and Clark 1978] for non-polar regions, it also preserves the pole structure (doubling the valence). Like Bærentzen et al. [2012], we have implemented a simplified scheme which uses only one ring information and works as an extension to *factored* Catmull-Clark subdivision [Warren and Schaefer 2004]. It is important to note that PAMs are closed under subdivision: the line skeleton of the subdivided PAM is a subdivided version of the skeleton of the original PAM.

While PAM conversion is obviously lossy, Figure 9 (top) shows that with the combination of subdivision and refinement of the initial PAM, we can recapture almost all the details of the original armadillo model.

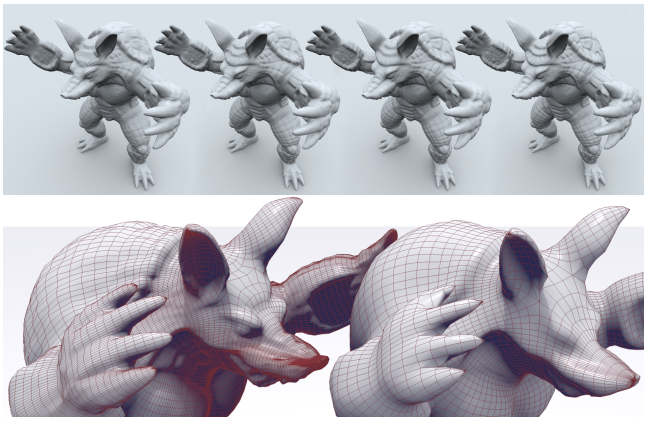

Figure 9: ***Top****: From left to right this figure shows the Armadillo mesh converted to a PAM (57k faces), PAM after one iteration of subdivision and refitting (227k faces), PAM after two iterations of subdivision and refitting (908k faces), and (far right) the original triangle mesh (346k faces).* ***Bottom****: The Armadillo as a highly detailed PAM (left) and after simplification has been performed, removing 600 strips from the mesh (right).*

Simplification. Methods for triangle mesh simplification or quad mesh simplification do not apply directly to PAMs since a PAM is a tri-quad mesh with certain constraints. We use a single operation which preserves the invariants of a PAM and is related to the polychord removal introduced by [Daniels et al. 2008]. In Section 3 we introduced the primitive simplification operations of region contraction. Recall that a region is a contiguous set of faces separated by spine edges and that a co-region is a contiguous set of faces separated by rib edges. We prioritize region (or co-region) removal by always picking the region (or co-region) whose longest spine (or rib) edge is shortest amongst all regions and co-regions. Removing 600 regions and co-regions in this way from the Armadillo model in Figure 9 (bottom, left), we arrive at the mesh on the right.

Pole Quadrangulation. When exporting the PAM mesh to another application, the high valency poles may be unsuitable for the downstream application. The procedure for pole removal initially selects a pair of edges that divide the triangle fan into two clusters of edges. The dividing pair of edges is not in these clusters which are shown red in Figure 10a. For each cluster, we introduce a new vertex on all edges, and connect these new vertices (Figure 10b). These connecting edges are immediately collapsed producing two

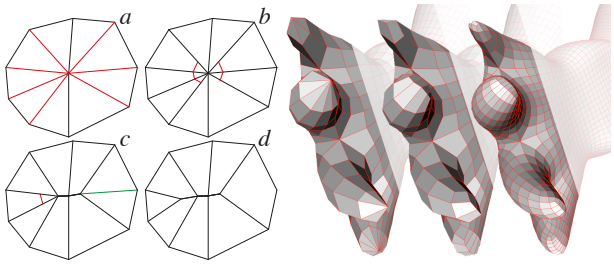

Figure 10: ***Left****: pole removal. The edges are divided into two clusters (shown red in a) which are connected by edges (red in b) that are collapsed (see c). A new cluster of two edges is selected (shown red in c) and the process is repeated. The remaining edge (green in c) is simply removed. Final result shown in d.* ***Right****: the hand of the Armadillo model. From left to right: a simplified polar mesh, the mesh after the poles have been quadrangulated, and after one step of Catmull-Clark subdivision.*

pairs of quadrilaterals around the original pole (Figure 10c). From each cluster, we remove the two outermost edges and repeat the process until each cluster holds less than two edges. If the count is zero, we are left with single triangle. If the count is one, we simply remove the remaining edge, producing a single quad. Both cases are illustrated in Figure 10c-d. To finally obtain a triangle free mesh, we perform one step of Catmull-Clark subdivision.

Skeleton Abstraction and Skinning. In our iPad application, we typically use properties of the skeleton without explicitly extracting it, such as for branch deformations (see Section 6). However, for use in third party tools, it is important to be able to extract a skeleton. Since each rib loop can be contracted to a point on the skeleton, we can consider each rib loop to be a node (or joint) of our PAM skeleton S. We manually pick a vertex on the PAM, whose associated rib loop defines the root joint of the hierarchy. The skeletal hierarchy S is then built using a simple breadth first traversal out from the root rib loop, along adjacent rib loops. Animation skeletons typically have far fewer joints that the contracted rib loop skeleton S. We create such an animation skeleton A from S, by iteratively collapsing some joints to their parent joints. A joint with one child may be collapsed if the angle between its incident skeletal edges is straighter than a threshold angle (we use angle $> 164^\circ$). While this approach works well in practice, a more global algorithm could further optimize lengths of joints and the overall curvature of a simplified chain of joints. For joints with more children, we allow a fixed number of collapses. The PAM skeleton shares the property of Reeb graphs that branch points are often on or close to the surface [Gebal et al. 2009]. Removing branch nodes has the benefit of pulling the branches away from the surface.

The PAM vertices are naturally rigged to the skeleton S by their associated rib loops. PAM vertices can be rigged to the animation skeleton A in various ways, the simplest of which is linear blend skinning, where each mesh vertex is controlled by a weighted combination of skeletal joints. Traditionally, skin weights are based on the shortest Euclidean distance from a mesh vertex to the skeleton A. Instead, we compute skin weights for a vertex based on the distance (along the skeleton S) between its associated rib loop centroid and the joints of A. This distance is easily found by traversing the chain of spine edges connecting the vertex to the rib loop associated with the joint. Our skin weights can optionally be exported along with the animation skeleton A for use in 3rd party tools. See Figure 3 (right). The visual difference between Euclidean skin weights and our weights is minimal, though our weights tend to better preserve rigidity of rib loops that are not orthogonal to A.

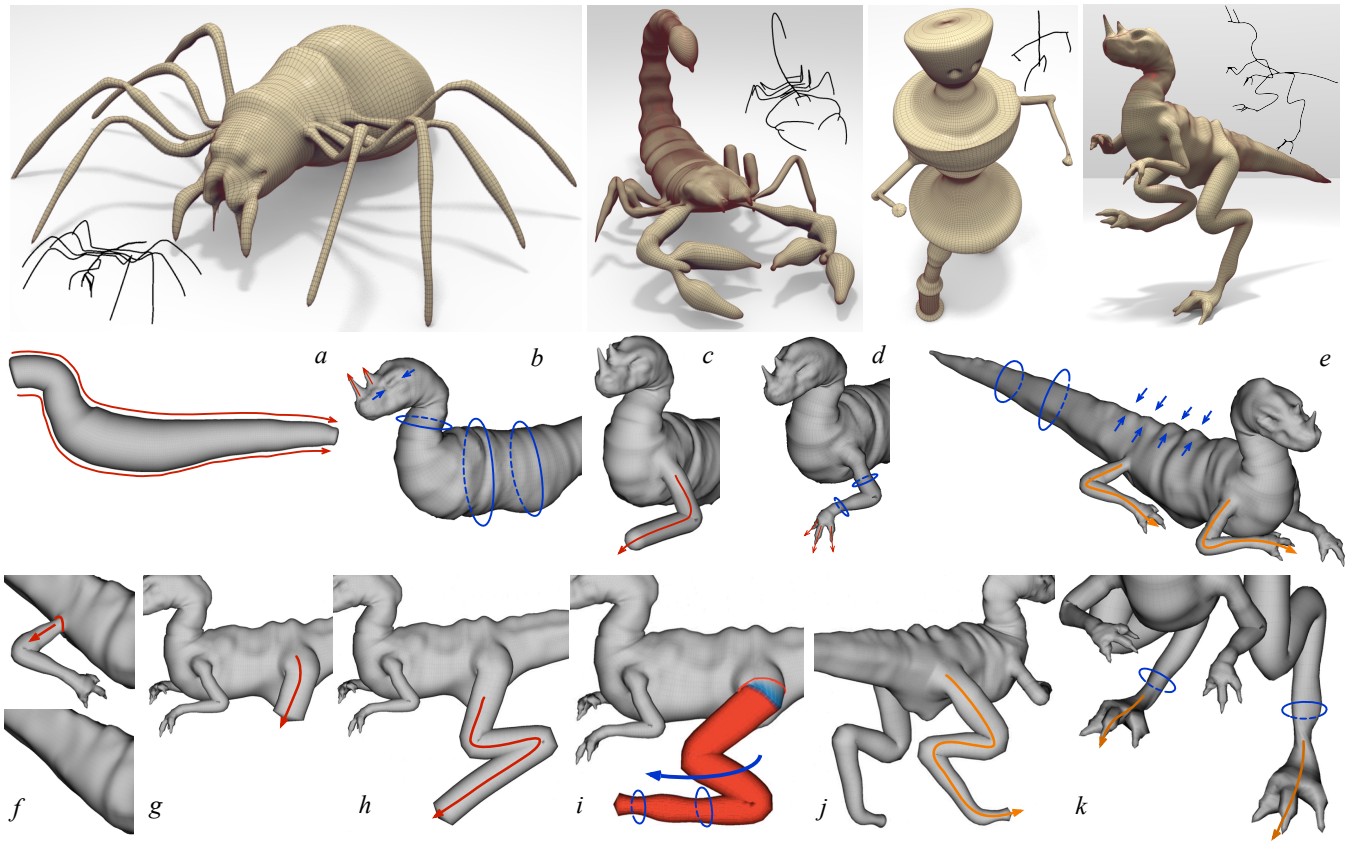

Figure 11: ***Top****: The spider, scorpion, robot, and dinosaur were all sculpted from scratch by a 3D artist using our iPad app. The inset black curves show the skeleton of each model.* ***Bottom****: The tools used in creating this dinosaur model are: initial shape creation (a), branch creation and sculpting (b), branch creation and some sculpting (c,d), sculpting and cloning of branches (e), branch deletion (f), branch creation (g), branch extension (h), branch deformation (i), and branch cloning (j,k).*

6 Multi-touch sculpting using PAMs

We demonstrate the effectiveness of the PAM representation on an Apple iPad. While a gestural multi-touch interface is well suited to interactive modeling of articulated shapes, there are presently few such prototypes on mobile devices, in part due to the computational complexity of working with a general mesh representation. The structure of a PAM reduces the memory and computation necessary for common mesh operations like dynamic refinement, remeshing and selection, enabling interactivity on small mobile devices.

In implementing the modeling tools provided by the application, we extensively use the structural and skeletal information in the PAM representation to select regions (subtrees), interpolate along the skeleton, and find edge loops where the mesh can be cut, to add or remove branches. We use the word branch to refer to an entire subtree, that forms part of the model on one side of a user indicated point. Our modeling operations are illustrated as snapshots from a typical modeling session in Figure 11 (bottom) and detailed below. Below we present timings. These are worst case numbers: timings have been measured for more geometry than we usually manipulate. The model used in all cases is a PAM representation of the Armadillo model at 37604 vertices.

Navigation: Similar to ILoveSketch [Bae et al. 2008] which uses mode switching to disambiguate gesture strokes from draw strokes, we use a two finger tap to switch between navigation and modeling. In navigation mode, typical 3D navigation gestures are employed. *Navigation runs at 58-60 fps.*

Initial shape creation: A two-finger stroke is used for the initial shape creation. We extract a skeletal curve in the view plane by averaging and smoothing the two finger strokes. Polar regions are created at the first and last skeleton point. Annular regions are created for all other skeletal points. The diameter is determined by the distance between the corresponding touch points. *30000 vertices are created within 1 sec.*

Branch creation or extension: A one finger stroke is used to create a new branch on a PAM. The first touch point is mapped to the closest projected vertex v (not a pole) of the PAM. The branch width w is a fraction of the perimeter of the rib loop containing v, and proportional to the speed of the stroke. We gather rib edges around v until their total length is approximately w, and cut them open, creating a boundary loop along the selected rib edges. Next, the user drawn stroke generates a 3D skeletal curve emanating from vertex v. Empirically, we found users to consistently imagine this curve as emanating normal to the surface of the PAM at v, but conforming to the stroke as drawn in the view plane soon after. We capture this by first generating the stroke on the view plane positioned in 3D at v. The rotation to align the curve tangent at v with the PAM surface normal at v is then applied with a smooth fall-off within a fraction (say 1/3) of the arc length of the curve. A polar region is created at the last skeleton point. Annular regions are

created for all other skeleton points with a radius of $w/2$, to produce a PAM with a boundary loop containing the same number of rib edges as the loop previously cut open. Finally the two boundary loops are trivially stitched together. If the first touch point is mapped to a pole, we extend the corresponding branch. First, the pole is removed to create a boundary loop. The new branch is then created and stitched to this boundary loop as previously described. *30000 vertices are created within 2 sec.*

Branch deformation: The user performs a long press to place a *pin* point. Next, by placing two fingers on the screen, a *pivot* point is defined and an Affine transform initiated. The transformation itself is defined by typical two-finger gestures: a circular twisting action for rotation around the view vector, spreading or gathering to scale and a pan gesture to translate. *Deformations are performed at 56-60 fps for 20000 vertices.* The rib loops containing the *pin* and *pivot* points, segment the PAM to define deformation extent. The transformation, as defined above, is applied to the region beyond the *pivot* point. The transformation smoothly falls-off over rib loops in the transition region between the *pivot* and *pin* points. The region of the PAM on the other side of the *pin* point is unaffected. In an alternative, simpler workflow, the user sets the pivot point, and the app automatically infers the placement of the pin point. *Segment computation takes 0.1 sec. for 20000 vertices.* We counter shearing introduced by rotation or translation in the transition region by rotating rib loops. For each rib loop we first compute an estimated tangent along its skeletal curve, as the line connecting the barycenters of its two adjacent rib loops. The rib loop is then rotated such that its best fitting plane is perpendicular to this skeletal tangent. *Rib rotation takes 0.01 sec. for 15000 vertices in the transition area.*

Branch cut-copy-and-paste: The user sets a *pin* point to indicate where to cut the branch. A long swipe across the branch then deletes it. If the *pin* point lies at a rib junction, we close the hole by stitching boundary edges together. Otherwise, a polar region is used to close the rib loop. A short swipe across a branch cuts it by deleting a sequence of spine edges along the rib loop containing the *pin* point. While in this detached state, the user can rotate the branch around the vector parallel to the first skeletal segment of the detached branch, using a rotate gesture. A one finger tap allows the user to reposition the branch to another place on the model. The same procedure as for branch creation is used to attach the branch. A subsequent swipe pastes the branch either at the same place or a new one (in which case the introduced hole is capped). *For 20000 vertices, detaching takes 0.7 sec. and deletion takes 0.6 sec. Repositioning takes 1.4 sec. for 5000 vertices.* Instead of a cut or delete gesture, an initial three-finger swipe across the branch copies it. A one-finger tap then creates a clone at the place indicated. The cloned branch can be rotated in the same way as a detached branch and pasted using a subsequent three-finger swipe. The latter gestures can be used to multiply clone a copied branch. *Copying of 5000 vertices takes 1.4 sec. and cloning them takes 1.2 sec.*

Sculpting and Smoothing: If both fingers are inside the model, two-finger spreading and gathering gestures sculpt the shape, creating dents or bumps, respectively. If the gesture starts with both fingers touching background, we smoothly contour the model along both sides of the ribs. If the gesture starts with one finger touching model and another touching background, we contour only along one side. The PAM structure makes it easy to find the rib loops that are closest to the touch points and affected by the contouring. Contouring can be isotropic or anisotropic. A patting gesture is used for smoothing. *5000 vertices affected by sculpting are determined within 0.02 sec. Deformations are carried on at 58-60 fps.*

7 Implementation and Results

The methods presented in this paper were implemented primarily in C++ and Obj-C (for UI elements) and based on a half-edge mesh data structure [Kettner 1998] which is the foundation for our implementation of the PAM representation on both of the platforms that we use: iOS 7/iPad and OSX 10.9/Mac. The Mac software was run on a 2.6GHz Intel Core i7 Retina MacBook Pro 10,1. The iPad application was tested on iPads of generation 3, 4, and 5. The fourth generation iPad was used for sculpting the models shown in this paper. The fifth generation iPad was used for timings in Section 6.

We tested the performance of the harmonic function generation and the triangle mesh to PAM conversion. The results are summarized in Table 1 which also states the model source. Note that this non-interactive code path is not efficient. Especially the generation of the harmonic function (with Jacobi iterations) could be much improved. The Man is shown in Figure 1. The original mesh (a) is

Model (source)	tri in	slices	faces out	f	PAM
Man (Princeton)	20096	300	31507	8.66 s	14.97 s
Armadillo (Stanford)	25944	150	21283	13.7 s	13.84 s
Bunny (Stanford)	6966	100	11874	4.23 s	2.18 s
Children (AIM@Shape)	20002	150	67443	8.32 s	17.22 s
Head (authors)	20016	150	37868	7.17 s	9.45 s
Fighter (Thingiverse)	12490	48	7587	N/A	2.63 s

Table 1: *This table shows the number of input triangles, level set slices, output faces (PAM) and the time it takes to create the harmonic function f and the actual triangle mesh to PAM conversion.*

shown after default PAM conversion (b); after simplification (to remove 420 regions or co-regions), refitting and subdivision (c); and after further interactive modeling (d). Other triangle meshes converted to PAMs are shown in Figure 12. The Dancing Children, Bunny and Armadillo models were converted using a harmonic function f created from constraints placed at obvious extrema of the model. The head model was more challenging. Constraints placed on eyeballs, in nostrils and on the ears produce a desirable mesh, albeit with a less meaningful skeleton (for example, there is no bone to control the lower jaw). A coordinate function was used as f, for the fighter. While the mesh along the fuselage is reasonable, there are no poles on the wings to capture it skeletally. We also see a loss of detail near the edges of the wings, where we have no way to constrain the spine edges to conform precisely to wing edges (we can manually constrain rib edges, as in Figure 8(right)).

A 3D artist was asked to create a set of models using our application. The results are shown in Figure 11 in the order in which the models were created. The artist spent an hour and a half on the spider, one hour and 12 minutes on the scorpion, 27 minutes on the robot and an hour on the dinosaur. We can also use the app to modify existing 3D models. In Figure 1d the arms of the model were cloned and the face and body sculpted. In Figure 13, the Armadillo was posed using the application while the cup and the hand were created from scratch. Snapshots from a typical modeling session shown on the dinosaur model in Figure 11 (bottom), highlight the interoperable nature of articulated shape modeling, where skeletal branching and posing operations are freely interspersed with sculpting and smoothing of geometric detail.

8 Discussion

Stripped to its essence, a PAM is a light-weight representation that enables the direct manipulation of a polygonal mesh representation while retaining its implicit skeletal structure. This skeletal structure and its induced shape segmentation is what allows us to pose, ani-

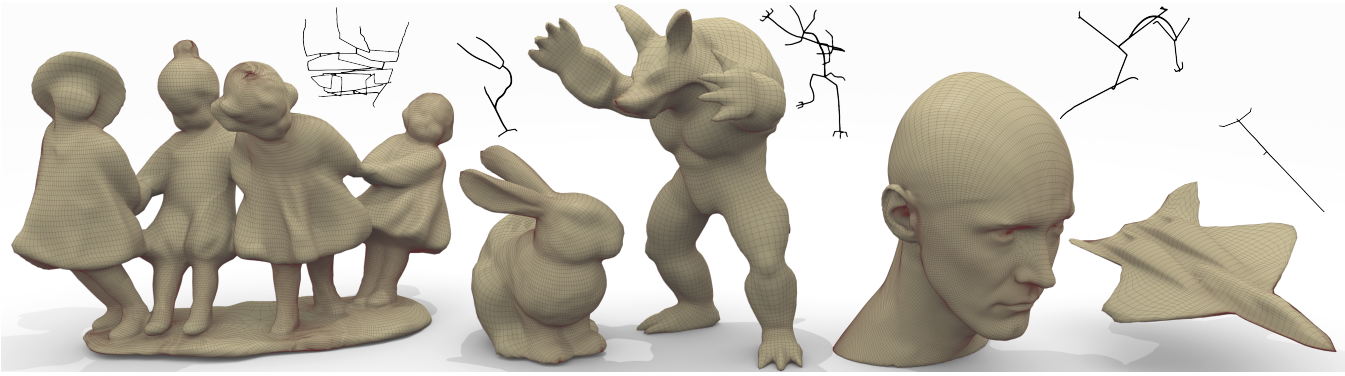

Figure 12: *These models are all the results of triangle mesh to PAM conversion. The Dancing Children model and the airplane have not been processed further. The Stanford Bunny has been refitted and the Armadillo and head model have been simplified and refitted. The inset black curves show the skeletons of the adjacent models.*

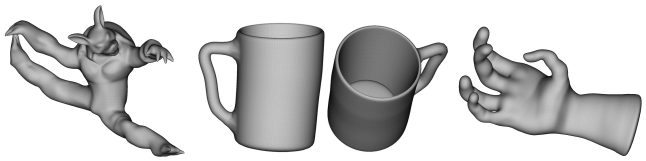

Figure 13: *The armadillo was posed using the iPad app. The cup and the hand were created using the app.*

mate, clone, delete, and transform a 3D model without recourse to dynamic parametrization, segmentation algorithms or background remeshing. We showcase the expressivity of PAMs using a simple multi-touch based iPad app for shape modeling, or editing of existing shapes converted to PAMs.

Our triangle mesh to PAM conversion process currently requires users to specify extreme points of the harmonic function, to guide the conversion. While such input also provides a user with precise control over the placement of poles, an automatic solution would be of interest. We considered using the auto diffusion function (ADF) [Gebal et al. 2009]. Unfortunately, the level sets of the ADF increase in frequency near the poles, and the ADF yields a spurious pole at its minimum, near the middle of the model. Therefore, we leave automatic generation of f to future work. Also pertaining to PAM conversion, the zero level curve (which may consist of several components) has special significance: this is where the front propagation (which changes all vertices to have valence four or three) starts off. Thus, the number of vertices on this initial curve determines the mesh sampling. We could adaptively control the sampling of this curve to avoid loss of detail in the mesh conversion, but this is not presently implemented. Finally, our front propagation algorithm is not sensitive to the topology of the input mesh, and tiny handles which lie entirely between two discretely sampled level set curves, are not presently handled.

Our iPad application and its tools were mimally designed to illustrate the benefits of a persistent mesh-skelton co-representation. We hope to augment it in the future with tools that exploit skeletal symmetry and textured brushes for sculpting detail. In summary, we argue that a mesh structure with an embedded skeletal segmentation can positively impact interactive shape modeling. A PAM is one such mesh-skeleton co-representation with various restrictions, such as the lack of t-junctions to arrest the global propagation of mesh detail along spine edges, or the high valence of polar triangle fans. We hope these limitations will fuel research in PAM generalizations and other novel mesh-skeleton co-representations.

Acknowledgements

We thank the reviewers for their insightful comments, Chris De Paoli for user feedback and sculpting, and Daniele Brazzolotto whose MSc project provided early inspiration. Models are from the Princeton Segmentation Database, the Stanford 3D Scanning Repository, the AIM@Shape Repository, and Thingiverse. This work has been funded in part by NSERC, GRAND and the Otto Mønsted Foundation through grant # 13-70-0688.

References

ANDREWS, J., JIN, H., AND SÉQUIN, C. 2012. Interactive inverse 3d modeling. *Computer-Aided Design and Applications 9*, 6, 881–900.

ATTENE, M., BIASOTTI, S., AND SPAGNUOLO, M. 2003. Shape understanding by contour-driven retiling. *The Visual Computer 19*, 2, 127–138.

BAE, S.-H., BALAKRISHNAN, R., AND SINGH, K. 2008. Ilovesketch: as-natural-as-possible sketching system for creating 3d curve models. In *Proceedings of the 21st annual ACM symposium on User interface software and technology*, ACM, 151–160.

BÆRENTZEN, J. A., MISZTAL, M. K., AND WEŁNICKA, K. 2012. Converting skeletal structures to quad dominant meshes. *Computers & Graphics 36*, 5, 555–561.

BANCHOFF, T. F. 1970. Critical points and curvature for embedded polyhedral surfaces. *The American Mathematical Monthly 77*, 5, 475–485.

BARAN, I., AND POPOVIĆ, J. 2007. Automatic rigging and animation of 3d characters. *ACM Transactions on Graphics 26*, 3, 72:1–72:8.

BESSMELTSEV, M., WANG, C., SHEFFER, A., AND SINGH, K. 2012. Design-driven quadrangulation of closed 3d curves. *ACM Trans. Graph. 31*, 6 (Nov.), 178:1–178:11.

BIASOTTI, S., FALCIDIENO, B., AND SPAGNUOLO, M. 2000. Extended reeb graphs for surface understanding and description. In *Discrete Geometry for Computer Imagery*, G. Borgefors, I. Nyström, and G. Baja, Eds., vol. 1953 of *Lecture Notes in Computer Science*. Springer Berlin Heidelberg, 185–197.

BIASOTTI, S., GIORGI, D., SPAGNUOLO, M., AND FALCIDIENO, B. 2008. Reeb graphs for shape analysis and applications. *Theoretical Computer Science 392*, 1, 5–22.

BLOOMENTHAL, J., AND WYVILL, B., Eds. 1997. *Introduction to Implicit Surfaces*. Morgan Kaufmann Publishers Inc., San Francisco, CA, USA.

BOMMES, D., LEMPFER, T., AND KOBBELT, L. 2011. Global structure optimization of quadrilateral meshes. *Computer Graphics Forum 30*, 2, 375–384.

BOMMES, D., LÉVY, B., PIETRONI, N., PUPPO, E., SILVA, C., AND ZORIN, D. 2013. Quad-mesh generation and processing: A survey. *Computer Graphics Forum 32*, 6, 51–76.

BOROSÁN, P., JIN, M., DECARLO, D., GINGOLD, Y., AND NEALEN, A. 2012. Rigmesh: automatic rigging for part-based shape modeling and deformation. *ACM Transactions on Graphics (TOG) 31*, 6, 198:1–198:9.

CATMULL, E., AND CLARK, J. 1978. Recursively generated b-spline surfaces on arbitrary topological meshes. *Computer-aided design 10*, 6, 350–355.

CHAUDHURI, S., KALOGERAKIS, E., GIGUERE, S., AND FUNKHOUSER, T. 2013. AttribIt: Content creation with semantic attributes. In *Proc. UIST*, ACM.

DANIELS, J., SILVA, C. T., SHEPHERD, J., AND COHEN, E. 2008. Quadrilateral mesh simplification. *ACM Transactions on Graphics 27*, 5, 148:1–148:9.

DONG, S., KIRCHER, S., AND GARLAND, M. 2005. Harmonic functions for quadrilateral remeshing of arbitrary manifolds. *Computer aided geometric design 22*, 5, 392–423.

FLOATER, M. S. 2003. Mean value coordinates. *Computer Aided Geometric Design 20*, 1, 19–27.

GALYEAN, T. A., AND HUGHES, J. F. 1991. Sculpting: An interactive volumetric modeling technique. *Computer Graphics 25*, 4 (jul), 267–274.

GEBAL, K., BÆRENTZEN, J. A., AANÆS, H., AND LARSEN, R. 2009. Shape analysis using the auto diffusion function. *Computer Graphics Forum 28*, 5, 1405–1413.

IGARASHI, T., MATSUOKA, S., AND TANAKA, H. 2007. Teddy: a sketching interface for 3d freeform design. In *ACM SIGGRAPH 2007 courses*, ACM, 21.

ISENBURG, M., GUMHOLD, S., AND GOTSMAN, C. 2001. Connectivity shapes. In *Proceedings of the conference on Visualization'01*, IEEE Computer Society, 135–142.

JI, Z., LIU, L., AND WANG, Y. 2010. B-mesh: A modeling system for base meshes of 3d articulated shapes. *Computer Graphics Forum 29*, 7, 2169–2177.

KETTNER, L. 1998. Designing a data structure for polyhedral surfaces. In *Proceedings 14th Annual Symposium on Computational Geometry*, ACM, 146–154.

KRISHNAMURTHY, V., AND LEVOY, M. 1996. Fitting smooth surfaces to dense polygon meshes. In *Proceedings of the 23rd Annual Conference on Computer Graphics and Interactive Techniques*, ACM, New York, NY, USA, SIGGRAPH '96, ACM, 313–324.

MELVÆR, E. L., AND REIMERS, M. 2012. Geodesic polar coordinates on polygonal meshes. *Computer Graphics Forum 31*, 8, 2423–2435.

MILNOR, J. W. 1963. *Morse Theory*. Princeton university press.

MYLES, A., AND PETERS, J. 2009. Bi-3 c 2 polar subdivision. *ACM Transactions on Graphics 28*, 3, 1–12.

NI, X., GARLAND, M., AND HART, J. C. 2004. Fair morse functions for extracting the topological structure of a surface mesh. *ACM Transactions on Graphics 23*, 3, 613–622.

PINKALL, U., AND POLTHIER, K. 1993. Computing discrete minimal surfaces and their conjugates. *Experimental mathematics 2*, 1, 15–36.

PIXOLOGIC, 2010. Zbrush.

SCHMIDT, R., AND SINGH, K. 2010. Meshmixer: An interface for rapid mesh composition. In *ACM SIGGRAPH 2010 Talks*.

SINGH, K., AND FIUME, E. 1998. Wires: A geometric deformation technique. In *Proceedings of the 25th Annual Conference on Computer Graphics and Interactive Techniques (SIGGRAPH '98)*, ACM, 405–414.

SORKINE, O., AND ALEXA, M. 2007. As-rigid-as-possible surface modeling. In *Proceedings of the Fifth Eurographics Symposium on Geometry Processing*, Eurographics Association, Aire-la-Ville, Switzerland, Switzerland, SGP '07, Eurographics, 109–116.

SRINIVASAN, V., MANDAL, E., AND AKLEMAN, E. 2005. Solidifying wireframes. In *Proceedings of the 2004 bridges conference on mathematical connections in art, music, and science. Banf, Alberta, Canada*.

STÃNCULESCU, L., CHAINE, R., CANI, M.-P., AND SINGH, K. 2013. Sculpting multi-dimensional nested structures. *Computers & Graphics 37*, 6, 753 – 763.

SUN, Q., LIN, J., FU, C.-W., KAIJIMA, S., AND HE, Y. 2013. A multi-touch interface for fast architectural sketching and massing. In *Proceedings of the SIGCHI Conference on Human Factors in Computing Systems*, ACM, 247–256.

TAKAYAMA, K., SCHMIDT, R., SINGH, K., IGARASHI, T., BOUBEKEUR, T., AND SORKINE, O. 2011. Geobrush: Interactive mesh geometry cloning. *Computer Graphics Forum (proceedings of Eurographics) 30*, 2, 613–622.

TAKAYAMA, K., PANOZZO, D., SORKINE-HORNUNG, A., AND SORKINE-HORNUNG, O. 2013. Sketch-based generation and editing of quad meshes. *ACM Trans. Graph. 32*, 4 (July), 97:1–97:8.

THIERY, J.-M., GUY, E., AND BOUBEKEUR, T. 2013. Sphere-meshes: Shape approximation using spherical quadric error metrics. *ACM Transaction on Graphics (Proc. SIGGRAPH Asia 2013) 32*, 6, 178:1–178:12.

VAILLANT, R., BARTHE, L., GUENNEBAUD, G., CANI, M.-P., ROHMER, D., WYVILL, B., GOURMEL, O., AND PAULIN, M. 2013. Implicit skinning: Real-time skin deformation with contact modeling. *ACM Trans. Graph. 32*, 4 (July), 125:1–125:12.

WARREN, J., AND SCHAEFER, S. 2004. A factored approach to subdivision surfaces. *Computer Graphics and Applications, IEEE 24*, 3, 74–81.

ZANNI, C., BERNHARDT, A., QUIBLIER, M., AND CANI, M.-P. 2013. Scale-invariant integral surfaces. *Computer Graphics Forum 32*, 8, 219–232.